Genesys®

Framework 8.1

# Management Layer

# User’s Guide

## About Genesys

Genesys is the world's leading provider of customer service and contact center software - with more than 4,000 customers in 80 countries. Drawing on its more than 20 years of customer service innovation and experience, Genesys is uniquely positioned to help companies bring their people, insights and customer channels together to effectively drive today's customer conversation. Genesys software directs more than 100 million interactions every day, maximizing the value of customer engagement and differentiating the experience by driving personalization and multi-channel customer service - and extending customer service across the enterprise to optimize processes and the performance of customer-facing employees. Go to 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 DVD, 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

## 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. Before contacting technical support, please refer to the ***Genesys Care Program 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 Licensing Guide.***

## Released by

Genesys Telecommunications Laboratories, Inc. www.genesyslab.com

**Document Version:** 81fr_us-ml_06-2013_v8.1.301.00

# Table of Contents

**Chapter 4**

# List of Procedures

# Preface

Welcome to the *Framework 8.1 Management Layer User's Guide.* This document introduces you to the concepts, terminology, and procedures relevant to this layer of the Genesys Framework.

This document is valid only for the 8.1 release(s) of this product.

**Note:** For versions of this document created for other releases of this product, visit the Genesys Technical Support website, or request the Documentation Library DVD, which you can order by e-mail from Genesys Order Management at orderman@genesyslab.com.

This preface contains the following sections:

For information about related resources and about the conventions that are used in this document, see the supplementary material starting on page 249.

## About Management Layer

The Management Layer is the Genesys software that provides numerous functions to monitor and control your Genesys installation.

## Intended Audience

This document is primarily intended for system architects and system administrators. It has been written with the assumption that you have a basic understanding of:

- Computer-telephony integration (CTI) concepts, processes, terminology, and applications

- Network design and operation
- Your own network configurations
- Genesys Framework architecture and functions
- Configuration Manager interface, terminology, and object-managing operations
- Genesys Administrator interface, terminology, and object-managing operations, if applicable
- Architecture and functions of the Genesys solutions that you are using

You must read the *Framework 8.1 Deployment Guide* before using this guide. The *Framework 8.1 Deployment Guide* contains information about the Genesys software you must deploy before deploying the Management Layer and instructions for deploying this Framework layer. In addition, it contains a sample worksheet that helps you successfully configure and install the Management Layer components.

# Making Comments on This Document

If you especially like or dislike anything about this document, feel free to e-mail your comments to `Techpubs.webadmin@genesyslab.com`.

You can comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this document. Please limit your comments to the scope of this document only and to the way in which the information is presented. Contact your Genesys Account Representative or Genesys Technical Support if you have suggestions about the product itself.

When you send us comments, you grant Genesys a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

# Contacting Genesys Technical Support

If you have purchased support directly from Genesys, please contact Genesys Technical Support.

Before contacting technical support, please refer to the *Genesys Care Program Guide* for complete contact information and procedures.

# Changes in This Document

## Version 8.1.301.00

This document has been updated as appropriate for features that are new in this release of Management Framework, including:

- Clarified use of startup and shutdown timeouts for third-party applications; see Chapter 8 on page 127.
- Removed troubleshooting advice for obsolete error "Too Many Log Events in Folder".
- Updated parameters for the mlcmd utility; see Appendix A on page 173.

## Version 8.1.201.00

This document has been updated as appropriate for features that are new in this release of Management Framework, including:

- Added information about monitoring NTP services; see "NTP Service Monitoring" on page 27.
- Moved procedures for creating and using startup scripts from Chapter 4 on page 47 to Appendix F.
- Updated instructions and parameters for the mlcmd utility; see Appendix A on page 173.
- Added information about the Solution Control Server utility logmsg; see Appendix B on page 181.

In addition, all references to Genesys 6.x or earlier products have been removed, as they have reached End of Support.

## Version 8.1.101.00

This is the first release of the *Framework 8.1 Management Layer User's Guide*. In the future, this section will list topics that are new or have changed significantly since the first release of this document.

Genesys

Chapter

# 1 Overview

This chapter briefly introduces Management Layer 8.1.

This chapter contains the following sections:

## Overview of Management Layer Functions

The Management Layer of Genesys Framework 8.1 provides the following functions:

- **Solution and application control and monitoring**—you control and monitor all solutions and applications from a single point. The Management Layer displays the real-time status of every configured `Solution` object, and you can activate and deactivate solutions and single applications from this layer. This control and monitoring also includes user-defined solutions.
- **Centralized logging**—applications log maintenance events in the unified log format and the events are recorded in one central location. That format enables easy selection of required log records and centralized log storage for convenient access and solution-level troubleshooting. With centralized logging, you can also track individual interactions, audit activities in your contact center, and store alarm history.
- **Alarm signaling**—flexible alarm signaling triggers alarms based on application maintenance events, thresholds for system performance variables, or Simple Network Management Protocol (SNMP) variables. Solution Control Server communicates alarms to Solution Control Interface and can write alarms to system logs. You can configure the system to convert alarms into SNMP traps and send them as e-mails to a specified e-mail address. (The latter automatically enables paging

notifications.) The Management Layer automatically associates alarms with the solutions they affect and stores alarms as active conditions in the system until either they are removed by another maintenance event or you clear them. Alarms are visible only if you have access to the application that generated the alarms.

- **Application failure management**—fault-management functions consist of detection, isolation, and correction of application failures. For nonredundant configurations, the Management Layer automatically restarts applications that fail. For redundant configurations, this layer supports a switchover to the standby applications and also automatically restarts failed applications.
- **Built-in SNMP functionality**—extended SNMP support enables both alarm processing and SNMP data exchange with an SNMP-compliant network management system (NMS). This means you can integrate a third-party NMS with a Genesys system to serve as an end-user interface for system control and monitoring and for alarm signaling.

**Note:** The SNMP functionality of the Management Layer is controlled by the Genesys licensing system. Refer to the *Genesys Licensing Guide* for information about ordering licenses that activate this functionality.

- **Individual host monitoring**—host parameters are monitored, including records of CPU and memory usage and information about currently running processes and services.
- **Support for geographically distributed environments**—a special mode in Genesys Solution Control Server simplifies the operation of a geographically distributed installation that uses a single Configuration Database.

**Note:** The Management Layer support for geographically distributed environments is controlled by the Genesys licensing system. Refer to the *Genesys Licensing Guide* for information about ordering licenses that activate this functionality.

See “Architecture” on page 21 for information about the Management Layer components and “Management Layer Functionality” on page 25 for more detailed information about the Management Layer functions.

Refer to the *Framework 8.1 Deployment Guide* for configuration and installation instructions.

# New in This Release

The following new features are introduced in the 8.1 release of the Management Layer:

- **Ability to enable and disable log filtering at the application level:** Log filtering can be enabled and disabled for individual applications. For more information, refer to the *Genesys Security Deployment Guide*.
- **Alarm scripts made environment independent:** Alarm Detection and Alarm Reaction scripts now use the name of the affected configuration object by default, instead of the database identifier (DBID). This ensures seamless XML import and export of Alarm Detection and Alarm Reaction script definitions.
- **New alarm detection events:** Existing and new log events are now used to indicate when Message Server was unable to insert log messages into the Log Database. Users can use these log events to create Alarm Conditions that trigger alarms when this situation occurs. For more information, see the Framework 8.1 Combined Log Events Help file.
- **Enhanced audit logs:** Host and Tenant attributes have been added to audit logs.
- **Enhanced monitoring of the state of NTP Services:** Local Control Agent can now monitor the state of NTP services. New logs report when an NTP service ceases to be available and when it becomes available. In addition, users can now configure the signature of an NTP service/daemon.
- **Ability to install Solution Control Server utilities separately:** Users can now install the Solution Control Server utilities without installing Solution Control Server. Previously, the utilities were only installed automatically with Solution Control Server. See the section “Installing Solution Control Server Utilities Separately” in the *Framework 8.1 Deployment Guide*.
- **Enhancements to the mlcmd command-line interface:** Access to mlcmd is now restricted to authorized users. In addition, the parameters now use a name or a DBID, and new parameters enable users to:
  - Clear all active alarms raised by an application or on the basis of a specific Alarm Condition.
  - Report CPU usage for each thread of a given process of a given application, and to store the results in an XML file.

  Refer to Appendix A on page 173 for more information.

# Component Compatibility

The following tables highlight the Management Layer functionality available in various releases of the Genesys configuration environment:

- Table 1, “Functionality Supported in 7.0 to 7.6 Environments,” on page 18

- Table 2, “Functionality Supported in 8.0 to 8.1 Environments,” on page 19

Consult these tables to verify how Management Layer components of a particular release would operate in the environment of the same or a different release.

To operate correctly, all Management Layer components must be of the same major release.

### Table 1: Functionality Supported in 7.0 to 7.6 Environments

| Management Layer Release | 7.0 Environment | 7.1 Environment | 7.5 Environment | 7.6 Environment |
|---|---|---|---|---|
| **Management Layer Components Release 7.0** | **Full compatibility:**<br>• **Full 7.0 functionality** | Forward compatibility:<br>• Full 7.0 functionality<br>• No 7.1 features | Forward compatibility:<br>• Full 7.0 functionality<br>• No 7.x features | Forward compatibility:<br>• Full 7.0 functionality<br>• No 7x features |
| **Management Layer Components Release 7.1** | Backward compatibility:<br>• Full 7.0 functionality<br>• No 7.1 features | **Full compatibility:**<br>• **Full 7.1 functionality** | Forward compatibility:<br>• Full 7.1 functionality<br>• No 7.5 features | Forward compatibility:<br>• Full 7.1 functionality<br>• No 7.x features |
| **Management Layer Components Release 7.5** | Backward compatibility:<br>• Full 7.0 functionality<br>• No 7.x features | Backward compatibility:<br>• Full 7.1 functionality<br>• No 7.5 features | **Full compatibility:**<br>• **Full 7.5 functionality** | Forward compatibility:<br>• Full 7.5 functionality<br>• No 7.6 features |
| **Management Layer Components Release 7.6** | Backward compatibility:<br>• Full 7.0 functionality<br>• No 7.x features | Backward compatibility:<br>• Full 7.1 functionality<br>• No 7.x features | Backward compatibility:<br>• Full 7.5 functionality<br>• No 7.6 features | **Full compatibility:**<br>• **Full 7.6 functionality** |
| **Management Layer Components Release 8.0** | Backward compatibility:<br>• Full 7.0 functionality<br>• No 7.x features | Backward compatibility:<br>• Full 7.1 functionality<br>• No 7.x features | Backward compatibility:<br>• Full 7.5 functionality<br>• No 7.x features | Backward compatibility:<br>• Full 7.6 functionality<br>• No 8.0 features |

**Table 1: Functionality Supported in 7.0 to 7.6 Environments (Continued)**

| Management Layer Release | 7.0 Environment | 7.1 Environment | 7.5 Environment | 7.6 Environment |
|---|---|---|---|---|
| **Management Layer Components Release 8.1** | Backward compatibility:<br>• Full 7.0 functionality<br>• No 7.x or 8.x features | Backward compatibility:<br>• Full 7.1 functionality<br>• No 7.x or 8.x features | Backward compatibility:<br>• Full 7.5 functionality<br>• No 7.6 or 8.x features | Backward compatibility:<br>• Full 7.6 functionality<br>• No 8x features |

**Table 2: Functionality Supported in 8.0 to 8.1 Environments**

| Management Layer Release | 8.0 Environment | 8.1 Environment |
|---|---|---|
| **Management Layer Components Release 7.0** | Forward compatibility:<br>• Full 7.0 functionality<br>• No 7.x or 8.x features | Forward compatibility:<br>• Full 7.0 functionality<br>• No 7.x or 8.x features |
| **Management Layer Components Release 7.1** | Forward compatibility:<br>• Full 7.1 functionality<br>• No 7.x or 8.x features | Forward compatibility:<br>• Full 7.1 functionality<br>• No 7.x or 8.x features |
| **Management Layer Components Release 7.5** | Forward compatibility:<br>• Full 7.5 functionality<br>• No 7.x or 8.x features | Forward compatibility:<br>• Full 7.5 functionality<br>• No 7.x or 8.x features |
| **Management Layer Components Release 7.6** | Forward compatibility:<br>• Full 7.6 functionality<br>• No 8.x features | Forward compatibility:<br>• Full 7.6 functionality<br>• No 8.x features |
| **Management Layer Components Release 8.0** | **Full compatibility:**<br>• **Full 8.0 functionality** | Forward compatibility:<br>• Full 8.0 functionality<br>• No 8.1 features |
| **Management Layer Components Release 8.1** | Backward compatibility:<br>• Full 8.0 functionality<br>• No 8.1 features | **Full compatibility:**<br>• **Full 8.1 functionality** |

Chapter

# 2 Architecture

This chapter discusses the Management Layer architecture and describes each component. It also provides component recommendations you can use while planning your overall Genesys installation.

This chapter contains the following sections:

## Management Layer Architecture

Figure 1 on page 22 shows interactions between Management Layer components and an application. To enable the Management Layer's solution-control and fault-management capabilities, you must install Local Control Agent on each host on which a monitored application (whether a Genesys server or a third-party application) is running. To enable the Management Layer's centralized-logging and alarm-signaling capabilities, you must configure each Genesys server application with a connection to Message Server.

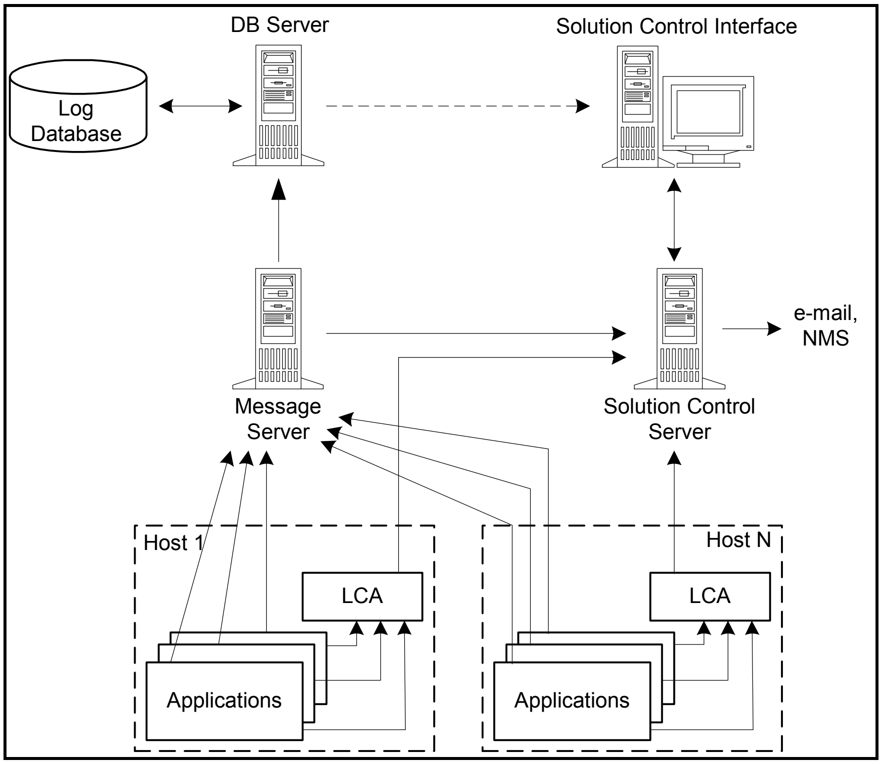


**Figure 1: Management Layer Architecture**

# Components Description

## Local Control Agent

Local Control Agent (LCA) is a daemon component that monitors, starts, and stops Genesys server applications as well as third-party server applications that you have configured in the Genesys configuration environment. In addition, LCA detects failures of Genesys servers and communicates their roles in redundancy context.

## Message Server

Message Server provides centralized processing and storage of all maintenance events that Genesys server applications generate. Events are stored as log records in the Centralized Log Database (also referred to as *Log Database*) where they are available for further centralized processing. Message Server also checks for log events configured to trigger alarms. If it detects a match, it sends the alarm to Solution Control Server (SCS) for immediate processing.

For usability reasons, Genesys does not recommend that you configure multiple Message Servers for one Log Database, with each Message Server assigned to handle logs for different applications. To view logs processed by a particular Message Server and therefore generated by a given application, you still have to filter the logs based on the application that generated them.

**Note:** Some solution components may also exchange messages via Message Server. You can find more details on this Message Server function in solution-specific documentation.

## Solution Control Server

Solution Control Server (SCS) is the processing center of the Management Layer. It uses Local Control Agents to start solution components in the proper order, monitor their status, and provide a restart or switchover in case of application failure. SCS also processes user-specified alarms.

## Solution Control Interface

Solution Control Interface (SCI) provides a user-friendly interface for managing Genesys solutions. SCI displays the status and configuration of all installed Genesys solutions and information about detected alarms and maintenance logs. You can start and stop solutions or single-server applications, including third-party applications, through this interface. SCI also allows advanced handling of maintenance logs and advanced viewing of host processes.

**Note:** Starting in release 8.0, the management functions performed through SCI can also be performed through the Genesys Administrator web-based user interface. For more information, see the *Framework 8.1 Genesys Administrator Deployment Guide.*

SCI also incorporates the forced re-login feature, which protects data by minimizing all open dialog boxes and forcing a logged-in user to log back into the system after an extended period of inactivity. For further information about using this feature, refer to the *Framework 8.1 Deployment Guide* and to the *Genesys 8.1 Security Deployment Guide*.

## Log DB Server

Log DB Server interfaces the Management Layer components with the DBMS (database management system) in which the Log Database is installed.

## Log Database

The Log Database stores all log records, including interaction-related records, alarm history records, and audit records.

## SNMP Master Agent

SNMP Master Agent is an interface between the Management Layer and an SNMP-compliant network management system (NMS). It distributes:

- SNMP traps, which are converted from alarms, from Solution Control Server to NMS.
- SNMP commands, which a user enters from an NMS interface, back to SCS for further processing.

For more information, refer to Chapter 7 on page 93.

Genesys

Chapter

# 3 Management Layer Functionality

This chapter describes the Management Layer capabilities that help you optimally manage the Genesys software serving your contact center.

This chapter contains the following sections:

**Notes:**

- You can also manage Management Layer components such as DB Server for the Log Database, Message Server, Solution Control Server, and Genesys SNMP Master Agent via the Management Layer as discussed in this chapter.
- Starting in release 8.0, the functions described in this chapter as being performed via the Solution Control Interface (SCI) can also be performed via the Genesys Administrator web-based interface. See *the Framework 8.1 Genesys Administrator Deployment Guide* for more information and detailed instructions.

## Solution Control and Monitoring Functions

Use the Management Layer's control and monitoring functions to:

- Start single applications or entire solutions through a single control operation from Genesys Administrator or Solution Control Interface (SCI).
- Shut down single applications or entire solutions in the same manner.

- Start all or a set of configured solutions.
- View the current runtime status of applications and entire solutions via Genesys Administrator or SCI.
- View all processes currently running on any host via SCI.
- View CPU and memory usage data for any host via SCI.
- View CPU usage data for each thread of a given process of an application.
- Manually switch operations from a primary server to its backup.
- Monitor the health of the NTP service, and change the signature of the NTP service or daemon.

For efficient solution maintenance, you should also use the monitoring functions for both solutions and applications. However, under normal circumstances, Genesys recommends that you always perform control functions over entire solutions as opposed to single applications. This ensures the correct logical order of application startup and shutdown and eliminates unnecessary error log events. In redundant configurations, the solution-level control operations also take into account the configured backup servers and start them up or shut them down automatically along with the primary servers.

Regardless of what level of control—applications or entire solutions—you choose to implement, the same architecture provides the control and monitoring functions. It consists of:

- Solution Control Server (SCS).
- As many instances of Local Control Agent (LCA) as there are computers with Genesys servers and/or with Management Layer–controlled third-party servers.
- Genesys Administrator or any number of instances of SCI, through which the control and monitoring capabilities are available directly to a user.

**Note:** A single SCS should be assigned to handle no more than 250 hosts and their applications. In this environment, SCS can handle the simultaneous failure of up to 50 hosts, while reflecting the actual runtime statuses of the hosts and affected applications in 15 seconds or less. If an SCS has to handle more than 250 hosts, the time to respond to host status changes might increase, especially if more than 50 of its hosts experience failures simultaneously.

If you need SCS to handle more than 250 hosts, Genesys recommends that you configure multiple Solution Control Servers in distributed mode to limit the load on each SCS. Refer to the *Framework Deployment Guide* for information about setting up a distributed SCS environment.

## Manual Switchover

The Management Layer provides an additional control function, a *manual switchover* of an application's operations to its backup application. Use this function, for example, for test purposes, during application upgrades, or during some hardware maintenance procedures. You can perform a manual switchover for any pair of redundant applications on which both primary and backup are running. During the switchover, the Management Layer changes the mode of the selected backup application to primary and the mode of the primary application to backup. The switchover mechanism is described in detail in "Fault Management Functions" on page 39.

You *cannot* manually switch over applications of these types:

- Configuration Server
- Database Access Point
- Solution Control Server

## NTP Service Monitoring

Starting in release 8.1, the Management Layer can monitor the health of the NTP services on supported platforms.

Local Control Agent (LCA) collects information about the state of the daemons at startup, and updates it periodically thereafter, when checking the state of third-party applications. Log events `00-08008` and `00-08009` report that the NTP service is available and is not available, respectively. Refer to the *Framework Combined Log Events Help* file for a description of these log events.

You can also configure the signature of an NTP service/daemon. Use the `signature` option in the `ntp-service-control` section of the `Host` object for which you want to configure the signature. Refer to the *Framework Configuration Options Reference Manual* for detailed information about this option.

## Permissions

To use Genesys Administrator or SCI to monitor solutions and applications, the user must have `Read` permission for the corresponding objects in the Configuration Database. To perform any control operations with respect to solutions and applications, the user must have `Execute` permission with respect to the corresponding objects. To receive alarm reactions related to applications and hosts, the user must have `Read` permission for the corresponding objects in the Configuration Database. For more information about security settings, see the *Framework 8.1 Deployment Guide.*

Remember that the Management Layer is a multiclient environment that makes solution-control functions available to an unlimited number of users simultaneously. The proper and responsible use of these functions is crucial for normal solution operations. Consider using the security capabilities of the Configuration Layer to limit access to these control functions to the trained personnel directly responsible for the contact center environment. Furthermore, schedule all control operations to occur during off-peak hours, preferably when the contact center is not processing any interactions, to ensure the availability of the customer interaction functions.

## Controlling Third-Party Applications

You can apply the Management Layer control and monitoring functions to third-party applications that meet the prerequisites listed on page 127.

These applications include, but are not limited to:

- SQL servers.
- CRM services.
- ERP services.

**Warning!** Windows users, please note that the Management Layer attempts to start an application without analyzing whether the application can run on an unattended computer (for instance, on a Windows computer with no user currently logged in) or whether the application can operate without a console window. Because the LCA that starts applications is always installed as a Windows Service, all processes start without a console window.

# Logging Functions

The Management Layer collects Genesys application logs of defined levels and stores them in a centralized database.

## Log Levels

Genesys applications can report log events at five levels of detail: *Alarm, Standard, Interaction, Trace,* and *Debug.* Only the first four are intended for on-site analysis by a user. Log events of the Alarm, Standard, Interaction, and Trace levels feature the same unified log record format and can be stored in the Centralized Log Database.

### Logging During Normal Operation

For complete specifications of log events reported at the Alarm, Standard, Interaction, and Trace levels, see *Framework 8.1 Combined Log Events Help.* To access this Help file, refer to Appendix C on page 185.

### Alarm Log Level

The Alarm-level logs contain only log records of the Alarm level. SCS generates Alarm log events on behalf of other applications when receiving from them log events configured as Detection Events in Alarm Conditions. Using this level, Solution Control Server reports the occurrence and removal of all alarms to the Centralized Log Database.

This level contains the Management Layer translations of log events of other levels into Alarms.

### Standard Log Level

Permanently enable *only* the Standard level of logging during solution operation in regular production mode. It contains high-level events that report both major problems and normal operations of in-service solutions.

An event is reported at the Standard level if it satisfies one of these criteria:

- Indicates that an attempt to perform any external operation has failed
- Indicates that the latest attempt to perform an external operation that previously failed has succeeded
- Indicates detection of a condition that has a negative impact on operations, actual or projected
- Indicates that a previously detected condition, which had a negative impact on operations, no longer exists
- Indicates a security violation of any kind
- Indicates a high-level data exchange that cannot be recognized or does not follow the expected logical sequence
- Indicates inability to process an external request
- Indicates successful completion of a logical step in an initialization process
- Indicates a transition of an application from one operational mode to another
- Indicates that the value of a parameter associated with a configurable threshold has exceeded that threshold
- Indicates that the value of a parameter associated with a configurable threshold that earlier exceeded the threshold has returned to its normal range

### Interaction Log Level

The Interaction-level logs report the details of an interaction processed by solution components that handle interactions. The log contains information about the processing steps for each interaction by each solution component.

An event is reported at the Interaction level if it:

- Is a recognizable high-level data exchange with another application about an interaction.

- Indicates a change in real-time state of an interaction handled by the application (unless such a change is visible from the high-level data exchange).

The specific criteria depend on a particular component and its role in interaction processing.

Use the Interaction-level log records to analyze and troubleshoot new interaction-processing scenarios, especially when you introduce new solutions or enable new functions within existing solutions. Note that Interaction-level records contain the interaction attributes, such as Interaction ID, that you can use to search for log events related to the same interaction but generated by different applications.

**Warning!** Using the Interaction level generates a higher number of logging events on the network and that may adversely affect the performance of the DBMS, Message Servers, and interaction-processing components.

### Trace Log Level

The Trace-level logs report the details of communications between the various solution components. The log contains information about the processing steps for each interaction by each solution component.

An event is reported at the Trace level if it satisfies one of these criteria:

- It is a recognizable high-level data exchange with another application.
- It is a recognizable high-level data exchange with an external system.
- It indicates a change in real-time state of user-level objects that the application handles (unless such a change can be seen from the high-level data exchange).

Use the Trace-level log records to analyze and troubleshoot new interaction-processing scenarios, especially when you introduce new solutions or enable new functions within existing solutions.

**Warning!** Using the Trace level generates a higher number of logging events on the network and that may adversely affect performance of the DBMS, Message Servers, and interaction-processing components.

## Logging During Irregular Operation

Standard-level, Interaction-level, and Trace-level log events do not contain all the details you or someone else may need to analyze and troubleshoot solutions malfunctions. That’s why Genesys Technical Support might ask you to provide relevant Debug-level logs when you request their assistance.

Because the Debug-level log events do not have a unified format, are not documented, and can only be stored in a text file, they are only useful to Genesys Technical Support. Logging at this level is likely to adversely affect application performance. Enable this log level only when a Genesys representative requests it. Keep in mind that running Genesys servers with the Debug level of logging is highly resource-intensive and, as such, is not recommended when you are in production mode.

Before you set a logging level more detailed than the Standard level, carefully consider whether a situation (such as the initial deployment or first signs of technical problems) really calls for it. Then, test for how more-detailed logging affects the network loads in a lab or controlled environment.

Note that changing the log level of a running application does not interrupt solution operations.

In addition to asking for Debug-level log records when you report a problem to them, Genesys Technical Support might also request that you reproduce the problem because:

- During regular operations, many contact-center systems, such as DBMSs, IVRs, and switches, do not employ logging at the level of detail required to diagnose serious technical issues.
- Reasons other than application failure can contribute to interaction-handling errors. For example, a call can be misrouted (delivered to a wrong DN) despite the fact that applications are functioning properly.

## Centralized Logging

The centralized logging function provides a number of advantages over the more traditional logging to a text file:

- Keeping log records of all applications in one place and presenting them in the unified log record format provides for a comprehensive view of the solutions' operations history.
- Using a relational DBMS such as the central log storage enables quick access to the required records and allows for advanced record selections, which you can base on a variety of search criteria.
- Viewing, via Genesys Administrator or SCI, the logs stored in a Centralized Log Database gives you an integrated view of the solutions' maintenance history and complements the solution-control and alarming capabilities.
- Deleting, via a wizard in SCI, the obsolete logs or logs of a particular solution, host, or application makes the Log Database management more convenient.

Given these advantages, Genesys recommends using the centralized logging as the primary method for storing Standard-level log events of all applications. When enabling the log output of the Interaction and Trace level (as directed in the section “Log Levels” on page 28), store log events of both levels in the Centralized Log Database in addition to log events of the Standard level. Genesys does not recommend the simultaneous use of both local and centralized logging options except for some special, temporary purpose.

The centralized logging system consists of:

- One or more Message Servers that collect log events from applications.
- One or more Log Databases.
- One or more DB Servers, which interface Message Server with the DBMS in which the Log Database is set up.
- Genesys Administrator or one or more Solution Control Interfaces.

Provided that the Standard level of log output is routinely used under normal production conditions, always limit the centralized logging system to one Message Server and one Log Database for all but large and geographically distributed interaction management networks.

If any part of the centralized logging system becomes unavailable, the log outputs of the affected applications are temporarily redirected to local binary files. Upon restoration of normal functioning, the applications automatically resume logging to the Centralized Log Database. The log records accumulated in the local binary files are automatically transferred to the Log Database.

If the connection between the Message Server and the Log DB Server is unavailable, messages are stored in a queue. When the connection is restored, the messages in the queue are written to the Log Database. See the “Message Server” section of the *Framework 8.1 Configuration Options Reference Manual* for more information.

**Note:** The format of records kept in the Log Database is specified in Chapter 9 on page 137.

## Viewing Log Database Entries

Although you can use any general-purpose DBMS client to make advanced selections from the Log Database, Genesys Administrator and SCI’s log-viewing capabilities may actually meet your needs just as well. In either interface, you can view an entire log. They also provide a number of predefined selections from the Log Database, which are based on the most typical maintenance-selection criteria:

- Records generated by components of a selected solution.
- Records generated by applications located on a selected host.
- Records of a specified output level.
- Records containing a specified combination of symbols in text.
- Records generated within a specified time interval.
- Records containing specified values of certain extended attributes.

You can use these predefined selection criteria in any combination.

You can also save selected log records in a regular text file or an XML (Extensible Markup Language) file. Use the instructions in *Framework 8.1 Genesys Administrator Help* or *Framework 8.1 Solution Control Interface Help.*

To delete obsolete log records, you can use Genesys Administrator or use the wizard in SCI.

## Logging and Application Performance

Follow these recommendations to increase an application's performance while enabling the application's logging:

- Always enable buffering of the log output when sending logs to a log file. (Refer to the "Common Log Options" chapter in the *Framework 8.1 Configuration Options Reference Manual.*)
- Store log files on the local disk of the computer running the application rather than storing them using network file systems. Such systems may not perform very well and the added network traffic for this storage can affect application performance.
- Configure only log events of the Standard level to be sent to the Log Database. For log events of other levels, consider using the `memory` output as the safest output in terms of application performance.
- Directing log output to the console (by using the `stdout` or `stderr` settings) can affect application performance. Avoid using these log output settings in a production environment.

## Alarms

The Management Layer uses the Centralized Log Database to store detailed and structured information about Alarm activation and clearance. (See "Alarm-Signaling Functions" on page 35 for more information about how alarms are generated.) Solution Control Server generates alarm-related information as log events of the Alarm level for each Alarm activation and clearance event. Solution Control Server attaches a set of extended attributes to each Alarm log event; in particular, the `ID` attribute uniquely identifies each Alarm.

For complete specifications of Alarm log events that SCS and Message Server reports and for information about extended attributes for each log event, see *Framework 8.1 Combined Log Events Help.* To access this Help file, refer to Appendix C on page 185.

SCI provides a special view in which to display Alarm History records from the Centralized Log Database. This view displays a summary of all information available for each recorded Alarm. For detailed information about viewing Alarm-History records, see Chapter 4 on page 47 and *Framework 8.1 Solution Control Interface Help*.

## Audit Trail

The Management Layer also uses the Centralized Log Database to store Audit-Trail records (from here on referred to as *Audit records*) that Framework components (in particular, Configuration Server and SCS) generate for configuration changes and control actions performed over processes, solutions, and alarms. Starting in release 8.1, the Audit records also include the name of the client application and details about the host on which the client application resides. This is to ensure compliance to Payment Card Industry Data Security Standard (PCI-DSS) 10.3.

Framework components generate Audit records as log events and, if available, attach extended attributes to Audit log events.

For information about setting up an audit trail and viewing the Audit logs, see "How to Set Up an Audit Trail and View Audit Logs" on page 67.

For complete specifications of Audit log events that Framework components report and for information about extended attributes for each log event, see *Framework 8.1 Combined Log Events Help*. To access this Help file, refer to Appendix C on page 185.

## Interaction Tracing

You can configure Framework components to send Interaction-level log events to the Centralized Log Database. You can later retrieve from the database all records related to a certain interaction, enabling you to trace its progress in the contact center.

**Note:** Storing Interaction-level log events in the Log Database might affect application performance, so Genesys does not recommend it in production environments.

Framework components might attach a set of extended attributes to each Interaction log event. In particular, each such event contains a unique identifier of the contact center interaction in the `IID` extended attribute.

**Note:** The set of extended attributes for Interaction-level log events may vary depending on a particular interaction's properties and the component that generates the log event.

For complete specifications of Interaction-level log events that Framework components report and for information about extended attributes for each log event, see *Framework 8.1 Combined Log Events Help*. To access this Help file, refer to Appendix C on page 185.

Use Genesys Administrator or SCI to display all Interaction-level records from the Centralized Log Database. Use predefined selection criteria to search for all records with a particular Interaction ID.

For detailed information about viewing Interaction-level log records with SCI, see Chapter 4 on page 47, and *Framework 8.1 Solution Control Interface Help*. For information about viewing Audit records with Genesys Administrator, see *Framework 8.1 Genesys Administrator Help*.

# Alarm-Signaling Functions

Maintenance events that the user may want to become aware of and react to immediately are communicated as Standard-level log events that Genesys applications generate. Each log event is assigned a unique number, which identifies the situation being reported. Thus, the alarm-signaling function of the Management Layer is based on the capability to detect the log events that have been preconfigured to trigger alarms and to send them to an alarm-processing center. In addition, the Management Layer monitors certain system and SNMP variables, which you can also use for alarm signaling.

## Alarm Detection

The Management Layer detects alarms by matching the following against the alarm conditions you have configured:

- Log events coming from all applications
- The thresholds of the system performance variables (such as CPU or memory usage) and of local or remote SNMP variables. (The SNMP threshold monitoring is available only when you have enabled SNMP functionality.)

SCS provides the same alarm-reaction processing for both Alarm-detection mechanisms.

### Using Log Events for Alarm Detection

To use a Standard-level log event to trigger an alarm, you must configure an object of the `Alarm Condition` type and associate it with the log event ID in the Configuration Layer. In configuring this object, you would:

- Specify the log event that should trigger this alarm at runtime.
- Assign an alarm category.
- Define the source of the alarm.
- Set conditions for automatic alarm clearance.

You can define a source of an alarm as a specific application, all applications of a particular type, or all applications of the interaction management network. In each case, the resulting alarm message contains the application name. So, you can always analyze the content of the alarm message and know the alarm's exact source.

**Note:** You can also use log events of the Interaction and Trace levels to trigger an alarm message.

**Warning!** When you create Alarm Condition objects for log events of any level, you must also configure applications to send log events of this level to Message Server. Otherwise, the Management Layer won't be able to detect them.

Once configured, an alarm condition automatically triggers an alarm in response to an occurrence of the log event on which the alarm condition is based. If the same log event occurs subsequently while the alarm is active, the clearance timeout is reset.

As previously noted, the alarm detection takes place in Message Server, so you must connect the potential sources of alarms to Message Server for alarm signaling to operate. If you are planning to use the recommended centralized logging function (see "Centralized Logging" on page 31), your applications should already be connected to Message Server. Otherwise, you need to set up Message Servers and configure your applications to connect to them specifically for alarm-detection purposes.

The easiest way to configure a new alarm condition is by using Genesys Administrator or the Alarm Condition Wizard in Solution Control Interface. For more information, see *Framework 8.1 Genesys Administrator Help* or *Framework 8.1 Solution Control Interface Help, respectively.*

Note that the Configuration Layer provides a number of preconfigured alarm conditions based on the events that cause service degradation in any environment. Before configuring your own alarm conditions, see if they may have been predefined in the Configuration Layer. For more information about predefined alarm conditions, see Chapter 10 on page 149.

### Using System Parameters and SNMP Thresholds for Alarm Detection

The alarm-detection mechanism for thresholds for system performance variables or SNMP variables is similar in many respects to the log-event-based mechanism. In particular, you must configure certain alarm conditions in the Configuration Database that indicate the values for the Management Layer to monitor.

When you are using thresholds for system performance variables, the Management Layer detects an alarm by periodically comparing the current value of the specified performance variable with the specified limits. If a change in the variable's value exceeds the specified limit, the Management Layer triggers an alarm.

When you are using SNMP variables as thresholds, the Management Layer detects an alarm by periodically comparing the current value of the specified SNMP variable, as identified by OID, with the specified limits. If a change in the variable's value exceeds the maximum limit of the specified minimum to maximum value range, an alarm is triggered. When the variable's value falls below the minimum limit of the specified range, the active alarm is cleared.

The Rising Threshold, which triggers an alarm when crossed *only if the value is rising*, must be a higher number than the Falling Threshold, which clears the alarm when crossed *only if the value is falling*. For example, if the Rising Threshold is 300 then the Falling Threshold must be less than 300.

This mechanism provides alarm signaling with both local SNMP variables—that is, variables from the Genesys MIB file, implemented locally in SCS—and with remote SNMP variables—that is, variables provided by third-party SNMP agents.

To monitor a variable of either type, use Genesys Administrator or the Alarm Detection Wizard in SCI to create:

- An Alarm Detection Script.
- A new Alarm Condition based on the Alarm Detection Script.

For more information, see *Framework 8.1 Genesys Administrator Help* or *Framework 8.1 Solution Control Interface Help.*

## Default Alarm Processing

Once it detects an alarm, Message Server sends it to Solution Control Server for processing. SCS processes the detected alarm in this way:

1. Stores the alarm in the system as `active` until it is removed manually, expires based on the configurable timeout, or is cleared by another log event, which you can optionally define in the Alarm Condition object as an automatic removal condition.
2. Generates log messages about every alarm detection and its removal.
3. Passes the alarm information and a list of all the running solutions that the alarm may affect to Solution Control Interface to display them for the user. SCS only passes alarm information about objects (such as applications or hosts) that the user currently logged in to Genesys Administrator or SCI has permissions to view. If necessary, the user can then take the appropriate action.

Note that for alarm processing to take place, you must connect SCS to the Message Servers that detect the alarms.

Since SCS maintains active alarms as runtime states, and if you are using SCI to view alarms, you do not have to permanently connect Solution Control Interface to SCS to display alarm information. Whenever you start Genesys Administrator or any instance of SCI, it automatically displays all active alarms currently registered in SCS as long as you have permissions to view the objects associated with the active alarms.

For more information, see *Framework 8.1 Genesys Administrator Help* or *Framework 8.1 Solution Control Interface Help.*

## Customized Alarm Processing

In addition to relying on the default alarm-processing actions, you can configure other actions (called *alarm reactions*) that the Management Layer is to take when it detects a specified alarm, including:

- Shutdown a specified application.
- Start up a specified application.
- Restart the application that reported the alarm.
- Start up a specified solution.
- Send an e-mail message with detailed information about the alarm to specified Internet addresses.
- Switchover operations from the application that reported the alarm to its backup application, for applications running in primary, backup, or either mode.
- Send an SNMP trap with detailed information about the alarm to a general-purpose network management system.
- Execute an operating system command.
- Change the value of a configuration option of a specified application, including the application that reported the alarm. (If the proposed change to an option is for a section or option that does not exist, the system creates both.)

Most of these reactions do not require any special arrangements. However, the switchover reaction type requires that the application in question have a backup application configured and running. The application restart and switchover mechanisms are described in detail in “Fault Management Functions” on page 39.

If you wish to use SNMP trap capabilities, you must install an SNMP master agent and configure your Solution Control Server to connect to it. You can use Genesys SNMP Master Agent or a third-party SNMP master agent you already have within you network management system—as long as it is compliant with the AgentX protocol. For instructions on these procedures and for detailed specification of the SNMP trap to which the Management Layer converts the alarms, see Chapter 7 on page 93.

Though the Management Layer itself does not provide paging notifications, you can arrange these through the supported e-mail or SNMP interfaces using your e-mail server or network management system, respectively.

An alarm reaction is configured in the Configuration Database as a Script object of the `Alarm Reaction` type. For runtime execution of a particular alarm, you must associate the alarm reaction with the corresponding Alarm Condition object. You can configure any combination of supported reactions with respect to any alarm condition. The easiest way to do this is by using Genesys Administrator or the Alarm Condition Wizard available in Solution Control Interface.

Starting with the initial 7.0 release, you can configure alarms in the Management Layer that execute alarm reactions not only at alarm activation, but also at alarm clearance. To achieve this, add the alarm reaction Scripts that should be executed when the alarm is cleared to the `Clearance Scripts` list of the corresponding Alarm Condition object. You can also use Genesys Administrator or the Alarm Condition Wizard to accomplish this.

You can also use the mlcmd utility to clear all active alarms raised by an application or on the basis of a specified Alarm Condition. Refer to Appendix A on page 173 for more information.

# Fault Management Functions

*Faults*—accidental and unplanned events causing a system to fail—present the biggest challenge to solution availability. The functions that detect, isolate, and correct various types of faults are partly incorporated into every Genesys component and partly implemented in the Management Layer of the Genesys Framework. The role of the Management Layer in application failure management is described in detail in the following subsections.

## Application Failures

A complete application failure may be a result of either an internal defect (for example, an infinite loop) or an external event (for example, a power failure). It may manifest as either a process nonresponse or termination. Typically, if a solution component stops working, the solution is no longer available to process customer interactions.

Since the application that fails cannot perform any functions, you must employ an external mechanism for both detection and correction of faults of this type. The Management Layer serves as such a mechanism. To detect an application failure, the Management Layer employs a simple monitoring component called Local Control Agent (LCA), which continuously maintains a connection with the application, confirming both its existence and ability to communicate. To make sure an application failure is never confused with a connection failure, the LCA that monitors a specific application always resides on the computer on which the application itself is running.

LCA is installed on a one-per-host basis and can connect to all Genesys applications located on the host. When a connection is broken, LCA generates a message to Solution Control Server, where an appropriate recovery action is chosen and executed according to the system configuration. SCS uses the

Advanced Disconnect Detection Protocol (ADDP) to recognize a loss of connection with LCA. A loss of connection is interpreted as a failure of the host (that is, as failures of all Genesys components running on that host).

**Note:** ADDP is, by default, enabled for the connection between SCS and LCA, with the ADDP timeout set to 9 seconds. With the default settings, SCS can detect and handle application failures in 20 seconds or less.

However, if there is a particular risk of network delays, Genesys recommends setting ADDP timeouts to values equal to or greater than 10 seconds, rather than relying on default values to avoid false detection of disconnection. You can modify ADDP parameters for the connection between SCS and LCA in the `Host` object of the computer that runs LCA. For more information about these settings, refer to the *Framework 8.1 Configuration Options Reference Manual*. For more information about ADDP, refer to the *Framework 8.1 Deployment Guide*.

If you have not configured a backup application for the failed component, the correction procedure normally consists of attempts to restart the failed process, if so configured. The same LCA component that detects application failures starts any Genesys application located on the host upon a command from SCS. If the application in question is a server, the clients automatically reconnect to this server once it is restarted and initialized.

Genesys recommends that you configure an automatic application restart procedure for all daemon applications.

**Warning!** Stopping an application via the Management Layer is not considered an application failure. Therefore, the Management Layer does not restart applications that it has stopped unless you have configured an appropriate alarm condition and alarm reaction for them.

If a backup application is configured and started, the Management Layer automatically switches operations over to that application, given that you have a *high-availability* license. If the application is a server, the clients automatically connect to the backup server.

The Management Layer currently supports `warm standby` between redundant components within the layer. It also supports switchovers between redundant client applications, regardless of the redundancy type specified by those applications. Starting with Management Layer 7.0.1, you must have a high-availability (HA) license to support either type of redundancy.

**Warning!** The Management Layer does not support `cold standby` redundancy type.

Starting with release 7.0, the Management Layer provides more robust switchover capabilities, and, in particular, allows detection of situations when a running application is unable to provide service and treats this situation as an application failure. The `Service Unavailable` application status serves this purpose.

When an application reports that its status has changed to `Service Unavailable,` and a backup server for this application is configured and started, the Management Layer automatically switches operations over to the backup server. Respectively, when both primary and backup applications are running with the `Service Unavailable` status, the backup application may report that it can now provide the service (that is, the backup application status changes to `Started`). In this case, the Management Layer automatically switches operations over to the backup application. As with a switchover resulting from an application failure, you must have an HA license to perform a switchover related to service unavailability.

**Note:** While some applications support the `Service Unavailable` status and report it under appropriate circumstances, others do not. (For instance, when T-Server loses its connection to the CTI Link, T-Server changes its status to `Service Unavailable`). The Management Layer operates based on the information supplied by an application and cannot automatically detect an application's inability to provide service. Refer to the application-specific documentation to determine if the `Service Unavailable` status is supported on the application side.

## Warm Standby Redundancy Type

Genesys uses the term *warm standby* to describe the redundancy type with which a backup server application remains initialized and ready to take over the operations of the primary server. Inability to process interactions that may have originated during the time it took to detect the failure is reduced to a minimum. `Warm standby` redundancy type also eliminates the need to bring a standby server online, thereby increasing solution availability.

The standby server recognizes its role as a backup and does not process client requests until its role is changed to primary by the Management Layer. When a connection is broken between the primary server and the LCA running on the same host, a failure of the primary process is reported. As a result, the Management Layer instructs the standby process to change its role from standby to primary, and the former standby starts processing all new requests for service.

**Note:** To switch to `Primary` mode, the backup Configuration Server must have an active connection to the Configuration Database during the failure of the primary Configuration Server.

While normal operations are restored as soon as the standby process takes over, the fault management effort continues. It consists of repeated attempts to restart the process that failed. Once successfully restarted, the process is assigned the standby role.

If Solution Control Server detects a loss of connection with the LCA of a host, SCS performs switchover for all applications located on the host, if backup applications are configured. There are two exceptions to this:

- A Configuration Server in backup mode ignores the switchover command if it detects another Configuration Server in primary mode. In other words, if the LCA residing on a host with a Configuration Server in primary mode goes down, the SCS requests that a Configuration Server in backup mode, on another host with an available LCA, switch over to primary mode. When the request is received, this Configuration Server checks whether the Configuration Server in primary mode is down, as indicated by a lost connection between the two Configuration Servers. The Configuration Server in backup mode switches over to primary mode only if this connection is down. If the connection still exists, no switchover occurs.
- An SCS in backup mode does not try to switch itself over if it can still detect the SCS that is in primary mode. In other words, if an SCS in backup mode loses its connection to an LCA residing on a remote host with an SCS in primary mode—either because the LCA went down or a network timeout caused the SCS to drop its connection—the SCS in backup mode checks whether it is still connected to the remote SCS in primary mode. If that connection is also lost, the SCS switches over and runs in primary mode.

## Hot Standby Redundancy Type

Genesys uses the term *hot standby* to describe the redundancy type with which a backup server application remains initialized, clients connect to both the primary and the backup servers at startup, and the backup server data is synchronized from the primary server. Data synchronization and existing client connections to the backup guarantee higher availability of a component. Configuration Layer and Management Layer components do not support `hot standby` between pairs of redundant components. They do support switchover between client applications configured with this type.

Hot standby redundancy type with client connection support is only implemented in T-Servers for most types of switches and is not implemented in applications of other types. For a complete description of the hot standby redundancy type, refer to the latest version of the deployment guide for your specific T-Server.

## Hang-up Detection

Starting in release 8.0, LCA can now use hang-up detection to detect unresponsive Genesys applications supporting this functionality. Users can then configure appropriate actions, including alarms if required.

To enable hang-up detection, use the configuration options `heartbeat-period` and `heartbeat-period-thread-class-<n>` to set the time interval in which a heartbeat message must be received before the application itself, or a thread of the application, is considered to be unresponsive for each application. A third option, `hangup-restart`, can be used to set the action that LCA takes when it deems the application to be non-responsive, either automatically restarting the application or just generating a notification of the situation.

For more information about these options, refer to the *Framework 8.1 Configuration Options Reference Manual*.

**Warning!** Use this functionality with great care, and only with those applications for which support of this functionality has been announced. Failure to use it properly could result in unexpected behavior, from ignoring the options to an unexpected restart of the application.

To determine if your application supports hang-up detection, refer to application-specific documentation. Support by Management Framework components is indicated in Table 3.

**Table 3: Management Framework Support for Hang-up Detection**

| Component | Application Level | Thread Level | Hangup Thread | |
|---|---|---|---|---|
| | | | Class | Name |
| Configuration Server | Yes | Yes | 1 | auth thread |
| Solution Control Server | Yes | Yes | 1 | mailer thread |
| DB Server | Yes | No | N/A | N/A |
| Message Server | Yes | No | N/A | N/A |
| SNMP Master Agent | Yes | No | N/A | N/A |

**Notes:**

- Configuration Server and Solution Control Server both have additional threads that can be monitored for hang-up detection. Configuration Server has an optional authentication thread; Solution Control Server has an optional mailer thread.
- The option `hangup-restart` does not apply to Solution Control Server.
- The option `heartbeat-period-thread-class-<n>` does not apply to DB Server, Message Server, and SNMP Master Agent.
- Solution Control Server must be running if the `hangup-restart` option is used.

## Remote Site Failures with Distributed Solution Control Servers

Starting in release 8.0, any Solution Control Server in the distributed environment can also detect the failure of a remote site controlled by another Solution Control Server in the environment and generate an appropriate log message.

Solution Control Server considers a remote site to have failed if it stops receiving polling messages from the Solution Control Server (or primary and backup Solution Control Servers, if configured) controlling the remote site within a specified time period. The time period is specified by the configuration option `alive_timeout`, configured on each Solution Control Server. Refer to the *Framework Configuration Options Reference Manual* for a detailed description of this option. In this case, the primary Solution Control Server generates log event `43-20600`.

If the remote site recovers and the Solution Control Server (or primary and backup Solution Control Servers, if configured) controlling that site starts to send polling messages, the primary Solution Control Server generates log event `43-20601` to indicate that the remote site is back in service.

For additional information, refer to the following documents:

- *Framework Deployment Guide* for more information about distributed Solution Control Servers, and for detailed instructions for configuring them
- *Framework Configuration Options Reference Manual* for a detailed description of the option `alive_timeout`.
- *Framework 8.1 Combined Log Events Help* for a full description of Solution Control Server log events 43-20600 and 43-20601. To access this Help file, refer to Appendix C on page 185.

# Built-in SNMP Support Functions

The Management Layer provides you, as a network administrator, with a way to monitor and control Genesys applications when using an SNMP-compliant third-party network management systems (NMS) user interface. Built-in support for an SNMP-compliant NMS means that Solution Control Server not only converts Genesys alarms into SNMP traps, but also processes various NMS commands and generates SNMP traps based on changes in the current status of an individual application.

The following requirements apply to the components that integrate Genesys 7 or later with an SNMP-compliant third-party NMS:

- Solution Control Server must contain built-in SNMP functionality, which was the case for all 7.0 releases.
- An SNMP master agent application must be compliant with the AgentX protocol. Use either Genesys SNMP Master Agent or one provided by a third-party. The Genesys SNMP Master Agent is available on the latest Management Framework 8.1 product CD.
- The license file must contain licenses that enable the SNMP functionality of the Management Layer. Refer to the *Genesys Licensing Guide* for information about how to order licenses and set up the licensing system.

Depending on the type of NMS you are using, you may also need to modify it to ensure a successful integration. For example:

- Make sure that your NMS knows the communication port number of the SNMP master agent.
- If you use several SNMP master agent applications, make sure their communication port numbers are unique and are known to the NMS.

In addition, check documentation for your NMS to find out if:

- You must load a copy of the Genesys MIB file into NMS so that your NMS can monitor and control Genesys applications.
- You must modify your NMS as needed so that it can display and process SNMP traps that an SNMP master agent generates on behalf of Genesys applications, including SCS.

Genesys

Chapter

# 4 Using the Management Layer

As described in Chapter 2, the Management Layer consists of a number of components. In addition, using various functions of the Management Layer often involves Configuration Layer components. As a result of this complex architecture, some users may find it difficult to put a Management Layer function to work. This chapter gives hands-on tips on how to use these various functions, including information about involved applications and reference documents.

This chapter contains the following sections:

- How to Manage Environments That Use Configuration Server Proxy, page 79

**Warning!** The Management Layer is designed to operate with one Configuration Server (one primary-backup pair of Configuration Servers). That is, you must connect the Management Layer components and all applications it controls to the same Configuration Server.

# How to Monitor Solutions, Applications, and Hosts

The monitoring function of the Management Layer allows a user to view the current runtime status of configured hosts, daemon applications, and entire solutions. As mentioned in "Solution Control and Monitoring Functions" on page 25, the monitoring function requires the installation of:

- Solution Control Server (SCS).
- An instance of Local Control Agent (LCA) for each Genesys host computer.
- Any combination of the following:
  - Genesys Administrator
  - An instance of Solution Control Interface (SCI) for each user who performs the monitoring.

**Note:** For SCS to monitor an application, you must specify the name of the executable file of the application in the properties of the `Application` object.

**Genesys Administrator**

Starting with release 8.0, you can monitor solutions, applications, and hosts through Genesys Administrator, a centralized Web-based user interface. The `Dashboard`, located on the `Monitoring` tab under `Environment`, displays the count and status of all applications and hosts that currently are configured. You can view the status of individual `Solution`, `Application`, and `Host` objects by selecting the appropriate category under `Provisioning > Environment`. Refer to *Framework 8.1 Genesys Administrator Help* for more information.

**Solution Control Interface**

You can also monitor solutions, applications, and hosts through SCI. You can categorize the views displayed in SCI by subject and by level of detail. For example, a subject-based List View (such as the Solution List View, the Application List View, or the Host List View) lists all configured solutions, applications, or hosts, and their current statuses. In addition, you can use List View, the Details pane, and the Details window to display each subject (such as a solution, an application, or a host) with a lesser or greater level of detail. In any view, the current subject status is both written in words and signified by the icon color. Refer to *Framework 8.1 Solution Control Interface Help for more information.*

**mlcmd** Starting with release 8.0, you can also use the mlcmd command-line utility to view the status of hosts, application, and solutions. For detailed information about this utility, and detailed instructions for using it, see Appendix A on page 173.

# How to View System Performance

The monitoring function of the Management Layer also allows a user to view the performance characteristics of configured hosts. The system performance viewing requires the installation of the same components as for host monitoring (see page 48). You can monitor a host computer that runs any Genesys-supported operating system as long as Local Control Agent is running on that computer.

SCI displays system characteristics that include a list of all processes that currently are running, with their process identification numbers and the command lines used for starting the applications. You can view memory and CPU usage for each process and the host as a whole in tabular or graphical format. In addition, for hosts that run Windows operating systems, you'll be able to view a list of Windows Services and their status. To display this information, open the Host Details window.

*Framework 8.1 Solution Control Interface Help* provides detailed instructions on how to display the Host Details window and describes the tabs that present the system performance characteristics.

# How to Start and Stop Applications and Solutions

With the control function of the Management Layer, you can start and stop individual applications with a single control function through Genesys Administrator or SCI. You can also use the control function to start and stop solutions, with one exception: you must use Genesys Administrator to start and stop a solution of type `Default Solution Type` or `Framework`.

Starting with release 8.0, you can also stop applications and solutions gracefully, also called `Graceful Stop` or graceful shutdown. See "Graceful Stop" on page 55 for more information.

As mentioned in "Solution Control and Monitoring Functions" on page 25, the control function requires the installation of:

- Solution Control Server (SCS)

- An instance of Local Control Agent (LCA) for each Genesys host computer.
- Any combination of the following:
  - Genesys Administrator
  - An instance of Solution Control Interface (SCI) for each user who starts and stops applications and solutions.

The start and stop commands are accessible in Genesys Administrator through the:

- `Start`, `Stop`, and `Graceful Stop` buttons on the toolbar
- `Start`, `Stop`, and `Graceful Stop` commands in the `Tasks` panel

The start and stop commands are accessible in SCI through the:

- `Actions` menu.
- `Start`, `Stop`, and `Graceful Stop` buttons on the main toolbar.
- `Start`, `Stop`, and `Graceful Stop` buttons on the Details pane or Details window of the subject.
- `File` menu on the Details window of the subject.
- Shortcut menus for selected objects.

**Note:** Genesys recommends that you start and stop entire solutions as opposed to starting and stopping single applications.

*Framework 8.1 Genesys Administrator Help* and *Framework 8.1 Solution Control Interface Help* provides detailed instructions on how to start and stop solutions and applications.

**mlcmd** Starting with release 8.0, you can also use the mlcmd command-line utility to start and stop applications and solutions. For detailed information about this utility, and detailed instructions for using it, see Appendix A on page 173.

## Processing the Start Command for Applications

When the Management Layer receives a request to start a particular application, SCS generates a command line and passes it to LCA, which executes the command. The command line contains:

- The working directory of the application as specified in the `Application` object’s `Properties`.
- The name of the executable or startup file of the application as specified in the `Application` object’s `Properties`.
- The host name of Configuration Server currently running in Primary mode.
- The port number of Configuration Server currently running in Primary mode.

**Note:** If you are using Configuration Server Proxy, SCS substitutes the host and port parameters of Configuration Server Proxy, where appropriate. For more information, see "How to Manage Environments That Use Configuration Server Proxy" on page 79.

- The application name as specified in the `Application` object's `Properties`.

Be sure that the working directory, executable (or startup) file name, application name, and startup timeout parameters are specified correctly in the `Application` object's `Properties`; otherwise, the Management Layer will be unable to start the application.

Unless an application is explicitly configured with a connection to Configuration Server Proxy, SCS starts the application against the Configuration Server to which SCS is currently connected. The port for connection to Configuration Server which SCS provides to the application is determined as follows:

- If the application is configured with a connection to Configuration Server, SCS starts the application using the same `PortID` as that is configured between the application and Configuration Server.
- If the application is not configured with a connection to Configuration Server, SCS starts the application using `PortID = default`. This provides backward compatibility for pre-release 7.5 applications.

## Processing the Start Command for Solutions

When the Management Layer receives a request to start a particular solution, SCS via LCA starts all applications included in the solution, in the order specified in the `Solution` object. The solution is considered Started when all mandatory applications that belong to it or their backup applications start successfully. When a mandatory solution component does not start, SCS determines if the solution configuration contains a backup server configured for this application. Then, one of the following happens:

- If the solution configuration does not contain a backup server, SCS interrupts the solution startup procedure and the solution status remains `Stopped`.
- If the solution configuration contains a backup server, SCS attempts to start the backup application:
  - If successful, the solution startup procedure continues through completion. After which, the Management Layer applies the restart mechanism to applications that could not start.
  - If unsuccessful, SCS interrupts the solution startup procedure and the solution status remains Stopped.

Note that after starting a mandatory application, Solution Control Server attempts to start a backup server configured for this application. This only happens when the backup server is included in the same solution. If the backup server application does not start while the primary server application is running, SCS proceeds by starting the next mandatory component.

## Starting Applications Automatically

Starting with release 7.0.1, Solution Control Server starts applications without a user command (that is, automatically) when:

- SCS starts.
- The computer running the applications is restarted.

If you do not modify an Application object as described in this section, SCS automatically starts the application only at the application's host restart, given that the application has been running prior to the host restart.

To enable this, you must configure the autorestart option for each application, as described in the following procedures.

### Procedure: Genesys Administrator—Configuring an application to be started automatically by Solution Control Server

**Purpose:** To enable SCS to start an application automatically every time SCS establishes a connection with LCA and the latter does not report the Started status for the application.

#### Prerequisites

- The Configuration Layer has been installed and configured, and is running.
- Either the Application object that you want to configure already exists, or you are performing this procedure while you are configuring it.
- You are logged in to Genesys Administrator.

#### Start of procedure

1. In Genesys Administrator, go to Provisioning > Environment > Applications, and double-click the application to open its Configuration tab.
2. Click the Options tab, and select Advanced View (Annex) from the View drop-down list.
3. If there is a section called sml, select it.
4. Click New.
5. In the New Option dialog box, specify the following:
   a. In the Section field, type sml, unless it is already displayed.
   b. In the Name field, type autostart.

  c. In the `Value` field, type `true`.

6. Click OK.
7. Click `Save and Close`.

**End of procedure**

## Procedure: Configuration Manager—Configuring an application to be started automatically by Solution Control Server

**Purpose:** To enable SCS to start an application automatically every time SCS establishes a connection with LCA and the latter does not report the `Started` status for the application.

### Prerequisites

- The Configuration Layer has been installed and configured, and is running.
- Either the Application object that you want to configure already exists, or you are performing this procedure while you are configuring it.
- You are logged in to Configuration Manager.

### Start of procedure

1. Open the `Application` object's `Properties` dialog box in Configuration Manager.
2. Click the `Annex` tab.

**Note:** If the `Annex` tab is not visible, go to `View > Options` and select `Show Annex tab in object properties.`

3. Create a new configuration section called `sml` or open the section with this name if it already exists.
4. Within this section, create a new configuration option named `autostart` and set its value to `true.`

### End of procedure

To disable an automatic application startup in scenarios in which SCS starts or in which an application was not running prior to its computer restart, either delete the `autostart` option or set its value to `false.`

## At SCS Startup

When SCS starts, it:

1. Establishes connections with all LCAs in the system and receives current statuses of all configured applications.
2. Checks the applications' configurations in the Configuration Database to determine whether you have enabled the `autostart` option.
3. For applications that have `Stopped` status and have the `autostart` option enabled, SCS:
   a. Waits for the interval specified in the `Startup Timeout` property of a particular `Application` object.
   b. Checks whether the application's status changes after the timeout has expired. If not, SCS starts the application as described in "Processing the Start Command for Applications" on page 50.

## At Computer Restart

When a computer restarts on which Management Layer–controlled applications are installed, SCS:

1. Establishes a connection with the LCA running on that computer.
2. For applications that were running (that is, had `Started` or equivalent status) prior to the computer restart, SCS:
   a. Waits for the interval specified in the `Startup Timeout` property of a particular `Application` object.
   b. Checks whether the application's status changes after the timeout has expired. If not, SCS starts the application as described in "Processing the Start Command for Applications" on page 50.
3. Identifies applications that were not running (that is, had `Stopped` status) prior to the computer restart.
4. Checks the applications' configurations in the Configuration Database to determine whether you have enabled the `autostart` option (see the procedure "Configuration Manager—Configuring an application to be started automatically by Solution Control Server" on page 53).
5. For applications that have `Stopped` status and have the `autostart` option enabled, SCS:
   a. Waits for the interval specified in the `Startup Timeout` property of a particular `Application` object.
   b. Checks whether the application's status has changed after the timeout expires. If not, SCS starts the application as described in "Processing the Start Command for Applications" on page 50.

As a result, both the applications that were running prior to a computer restart and the applications that were not running but whose configuration contained the `autostart` option set to `true` are started automatically after you restart a computer.

### Starting Third-Party Applications Automatically

Management Layer supports the automatic start-up and restart of third-party applications as described above. However, control and monitoring of third-party applications is performed in a different manner than that for Genesys applications. For details about how third-party applications are controlled, refer to Chapter 8 on page 127.

## Graceful Stop

When you stop an application or a solution, it shuts down, ceasing all processing immediately. This can have a detrimental effect on the rest of the system. Starting with release 8.0, you can stop an application or a solution gracefully, in a manner known as a *graceful stop,* or *graceful shutdown.*

**Note:** You must use Genesys Administrator to stop solutions of type `Framework` or `Default Solution Type`.

Applications that are being stopped gracefully refuse any new requests, but continue to process their current requests. Applications are stopped only after they have finished processing all of their requests.

The graceful stop command can be issued to any application. Applications that support this functionality process the command as described in the preceding paragraph. Applications that do not support the graceful stop functionality are stopped ungracefully.

For each application, you can specify a timeout with the `suspending-wait-timeout` configuration option in the `Application` object's `Annex`. If the status of the application does not change to `Suspending` after the graceful stop command but before the timeout expires, the application is considered not to support graceful shutdown, and is stopped ungracefully.

**Note:** The Shutdown Timeout does not apply to third-party applications. See “Stopping Third-Party Applications” on page 134.

For a solution to stop gracefully, the graceful stop command is issued to each of its composite applications. How each composite application handles the command depends on whether the application supports the graceful-stop functionality.

**Note:** Because a number of solutions can share the same applications, some solution components may continue to have a status of `Started` after you stop the solution.

For more information about graceful stop, refer to *Framework 8.1 Genesys Administrator Help* or *Framework 8.1 Solution Control Interface Help*. For details about the `suspending-wait-timeout` configuration option, refer to the *Framework 8.1 Configuration Options Reference Manual.*

# How to Use Startup Files

Some Genesys applications require special scripts to start and stop the application. Refer to the deployment procedures of your specific product to determine if separate start and stop files must be configured.

For Genesys applications that require both start and stop scripts to function properly, you must configure these scripts instead of single batch files as discussed in this chapter. For `Application` objects that have the commands `start_stop.start_command` and `start_stop.stop`_command specified in their `Annex`, the specified values would be used to start and stop the application on the target host. If these parameters are not specified, by default, system termination signals would be sent to the running process when the stop is issued. But if the `stop_command` is specified, an external script will be invoked instead for those applications that require special handling for termination.

**Note:** For Genesys applications for which the command line, command line argument, and `start_command` are all specified, the value in the `start_command` would take precedence.

If your application requires startup files, use the procedures in Appendix F on page 245.

# How to Manage Third-Party Applications

You can use the control and monitoring function of the Management Layer to manage third-party applications configured in the Genesys Configuration Database. You can use the Management Layer to start and stop third-party applications. Even if you did not use the Management Layer to start a particular application, the application’s runtime status is displayed. For managing third-party applications, install the same components as for monitoring and starting Genesys application (see page 48 and page 49).

The monitoring views and control commands are available through Genesys Administrator and SCI, just as they are when managing Genesys applications.

For more information about how the Management Layer handles third-party applications, see Chapter 8.

# How to Configure Logging

**Notes:**
- Message Server can only write logs using a PostgreSQL DBMS if the Log DB Server version supports PostgreSQL.
- Genesys strongly recommends that log records be stored securely.

As mentioned in “Logging Functions” on page 28, the logging function requires the installation of:

- One or more Message Servers that collect log events from applications.
- One or more Log Databases, and one or more DB Servers, which connect Message Server with the DBMS in which you have set up the Log Database.

Log configuration options are described in the “Common Configuration Options” section of the *Framework 8.1 Configuration Options Reference Manual*.

**Genesys Administrator and Configuration Manager**

You can configure log options for a single application through Genesys Administrator and Configuration Manager on the `Options` tab of the respective `Application` object. For more information, refer to *Framework 8.1 Genesys Administrator Help* or *Framework 8.1 Configuration Manager Help*.

**Solution Control Interface**

In Solution Control Interface, you can use the Log Wizard to configure logging functions for single applications and solutions. You can also use the Wizard to configure logging functions for all applications installed on a single host, for a set of applications you select, or for entire solutions. This functionality is possible because log configuration options are unified for all Genesys server applications.

To configure logging through SCI, launch the Log Wizard accessible in SCI. You can launch this Wizard from the main SCI toolbar using the `Run a Wizard` button or from shortcut menus of the objects displayed in SCI. When started for an `Application` object, the Log Wizard sets up log configuration options for the corresponding application. When started for a set of applications, the Log Wizard sets up log configuration options for all applications you select. When started for a `Solution` object, the Log Wizard sets up log configuration options for all applications included in the corresponding solution. When started for a `Host` object, the Log Wizard sets up log configuration options for all applications installed on the corresponding host.

Refer to *Framework 8.1 Solution Control Interface Help* for more information about the Log Wizard.

**Log Levels**

Log levels are defined in “Logging Functions” on page 28. Note that changing the log level of a running application does not interrupt solution operations.

The exceptions to this rule are Configuration Server and DB Server for the Configuration Database:

- You must configure options for these two applications in their configuration files.
- You have to restart these two applications for the new values to take effect.
- You cannot use the Log Wizard for these two applications.

For complete specifications of log events reported at the Alarm, Standard, Interaction, and Trace levels, see *Framework 8.1 Combined Log Events Help.* To access this Help file, refer to Appendix C on page 185.

## LCA Logging

Unlike with other server applications, you do not configure an `Application` object for LCA. However, starting with release 7.0, you can change the default settings for common log options for LCA.

To do so, create a configuration file called `lca.cfg` or save the sample configuration file, `lca.cfg.sample` which the installation creates, as `lca.cfg`. In this file, specify new values for appropriate options. The configuration file only contains the `log` section. You must locate this file in the same directory as the LCA executable file.

**Note:** You can also specify a custom name for the configuration file. To start LCA with a custom name use the following format:
*`<executable name> <port>`* `-c` *`<configuration file name>`*
For example (UNIX): `lca 7117 -c` *`lca_custom.cfg`*
Where *`lca_custom.cfg`* is the user defined configuration file.

The LCA configuration file must have the following format:

```
[log]
<log option name>=<log option value>
<log option name>=<log option value>
```

For more information about common log options and the LCA configuration file, refer to the *Framework 8.1 Configuration Options Reference Manual.*

# How to Customize Log Events

Log levels are defined in “Logging Functions” on page 28. Each log event has a default log level. Starting with release 7.6, you can customize log events for any application in release 7.6 or later by changing the default log level of an event to a more appropriate level, or by disabling the event completely.

You can toggle the customizations on and off, without deleting them. This enables you to specify the customized log levels at any time, but only use them when appropriate. Note that this option enables or disables all of the customizations; it cannot be applied to specific ones.

Customizations are unique to the application in which they are created. For example, application A customizes a set of log events. application B does not customize any log events. If the feature is activated, the log events will have the customized properties of the application which generated them—if generated by application A, the log events will be customized as specified by A; if generated by application B, the log events will not be customized. If application B was to customize the same set of log events, but with different custom definitions, the log events generated by application B would be customized as specified by B.

**Warning!** Use caution when making these changes in a production environment.

Depending on the log configuration, changing the log level to a higher priority may cause the log event to be logged more often or to a greater number of outputs. This could affect system performance.

Likewise, changing the log level to a lower priority may cause the log event to be not logged at all, or not logged to specific outputs, thereby losing important information. The same applies to any alarms associated with that log event.

In addition to the precautionary message above, take note of the following:

- When the log level of a log event is changed to any level except none, it is subject to the other settings in the `log` section at its new level. If set to none, it is not logged and therefore not subject to any log configuration.
- Changing the log level of a log using this feature changes only its priority; it does not change how that log is treated by the system. For example, increasing the priority of a log to Alarm level does not mean that an alarm will be associated with it.
- Each application in an HA pair can define its own unique set of log customizations, but the two sets are not synchronized with each other. This can result in different log behavior depending on which application is currently in primary mode.
- This feature is not the same as a similar feature in Universal Routing Server, version 7.2 or later. In this Framework feature, the priority of log events are customized. In the URS feature, the priority of debug messages only are customized. Refer to the *URS 7.5 Reference Manual* for more information about the URS feature.
- You cannot customize any log event that is not in the unified log record format. Log events of the Alarm, Standard, Interaction, and Trace levels feature the same unified log record format.

**Note:** Predefined log events of the Debug level are also in the unified log record format, and therefore can be assigned to another log level. However, any application can generate a raw text stream and call it a debug log event. Such non-unified log messages cannot be reassigned.

Refer to *Framework 8.1 Genesys Administrator Help* or *Framework 8.1 Solution Control Interface Help,* as appropriate, and the *Framework 8.1 Configuration Options Reference Manual* for instructions for customizing logs, and detailed examples.

# How to View Centralized Logs

With the Management Layer logging function, you can view the log records stored in Centralized Log Database, filter log records by their level, and search for records meeting the specified criteria. The log-viewing function requires the installation of:

- One or more Message Servers that collect log events from applications.

  **Note:** For usability reasons, Genesys does not recommend that you configure multiple Message Servers for one Log Database, with each Message Server assigned to handle logs for different applications. To view logs processed by a particular Message Server and therefore generated by a given application, you still have to filter the logs based on the application that generated them.

- One or more Log Databases.
- One or more DB Servers, which connect Message Server with the DBMS on which you have set up the Log Database.
- Any combination of the following:
  - Genesys Administrator
  - An instance of SCI for each user who views logs. For SCI to display log records, you must connect it to the Database Access Point configured for the DB Server that provides access to the Centralized Log Database.

**Note:** Log filtering can be enabled and disabled for individual applications. Refer to the Genesys Security Deployment Guide for more information.

## Procedure: Genesys Administrator—Viewing Centralized Log records

### Prerequisites

- Management Layer components are installed and running.
- Centralized Logging is enabled.
- You are logged in to Genesys Administrator.

### Start of procedure

1. Do one of the following, as appropriate, to display a list of log records in the Details panel:
   - To view all log records stored in the Centralized Log Database, go to `Monitoring > Environment > Centralized Log`.
   - To view all log records for a specific `Application` object, go to `Provisioning > Environment > Applications` and select the Application.
   - To view all log records for all applications running on a specific `Host` object, go to `Provisioning > Environment > Hosts` and select the Host.
2. To view an individual record or a subset of records, do one or both of the following:
   - Define one or more filtering criteria using the query builder fields above the list of log records.
   - Select a log record and, with one button, define a filter based on the value of certain fields in that record.

Refer to *Framework 8.1 Genesys Administrator Help* for more information about viewing the Centralized Log Database.

### End of procedure

## Procedure: Solution Control Interface—Viewing Centralized Log records

### Prerequisites

- Management Layer components are installed and running.
- Centralized Logging is enabled.
- You are logged in to Solution Control Interface.

### Start of procedure

1. In the Shortcuts bar of SCI, select View > Centralized Log. The list of log records in the Centralized Log Database appears in the List pane.
2. To view an individual record or a subset of the records, filter the list by doing any of the following:
   - Use the Find command to retrieve particular database records. The Find command, which is accessible from the Action menu when the Log View is displayed, enables you to search for a particular log record by log level, host name, application name, or record text.
   - Use the Log Advanced Search Wizard. The Wizard, accessible by the Run a Wizard button on the main SCI toolbar, enables a search based on a custom SQL statement or on user-specified criteria. Criteria for a Wizard search include log level, log generation time, log source, or extended attributes. Log source can be a particular application, applications that belong to a particular solution, or applications that run on a particular host computer.
   - Select a View type to view log records related to a selected object. For example, an Application Details window displays log records generated by the application; a Host Details window displays log records generated by all applications installed on the host computer; and a Solution Details window displays log records generated by all applications included in the solution.
3. To save the selected log records in a regular text file or an XML file, right-click in the List pane and select the appropriate format from the context menu. For information about the formats of these files, see Chapter 9 on page 137.

*Refer to Framework 8.1 Solution Control Interface Help* for a complete description of the log-viewing functionality.

### End of procedure

**Log Event Specifications**

For complete specifications of log events reported at the Alarm, Standard, Interaction, and Trace levels, see *Framework 8.1 Combined Log Events Help*. To access this Help file, refer to Appendix C on page 185.

# How to View Real-Time Logs

With the Management Layer logging function, you can enable and disable the real-time viewing of log messages that applications send to the network log output. Log messages differ from log records in that real-time log messages are displayed in SCI immediately when applications send them. Log messages are only displayed as text in SCI. Because Message Server collects log messages from applications and passes them directly to SCI, the real-time log-viewing function requires the installation of one or more Message Servers and an instance of SCI for each person who performs the log viewing.

To view real-time logs, enable the application real-time log with the `Start Real-Time Log` command, which is accessible from the `Log` menu of the `Application Details` window in SCI. Once activated, real-time logs display on the `Real-Time Log` tab of that window.

**Note:** You can start real-time logs only in Solution Control Interface—not in Genesys Administrator.

For SCI to display an application's real-time logs:

- The selected application must be started.
- The selected application must have the `network` log output configured.
- The selected application must have a configured connection to the Message Server, and this Message Server must be started.

Refer to *Framework 8.1 Solution Control Interface Help* for detailed instructions for enabling, disabling, and viewing real-time logs.

# How to Manage Log Records

You can manage records in the Log Database by:

- Using Genesys Administrator. Refer to *Framework 8.1 Genesys Adminsitrator Help* for instructions.
- Using the Log Database Maintenance Wizard in SCI.
- Creating your own scripts for automated database purging.

**Note:** Log records also contain alarm history. You should be careful not to delete current alarm history records when you remove log records from the Log Database.

The log-managing function requires the installation of the same components as for the log-viewing function (see "How to View Centralized Logs" on page 60).

## Using the Log Database Maintenance Wizard in SCI

You can use the Management Layer logging function to manage log records stored in the Centralized Log Database. With the Log Database Maintenance Wizard, available in SCI, you can specify criteria—through a custom SQL statement or individual selections—for the search and removal of log records from the database. Available criteria categories include log level, log generation time, log source, or extended attributes. Log source can be a particular application, applications that belong to a particular solution, or applications that run on a particular host computer.

Launch the Log Database Maintenance Wizard with the `Run a Wizard` button on the SCI toolbar. *Framework 8.1 Solution Control Interface Help* contains detailed instructions on how to start the Wizard.

# Automating the Purging Functionality

This section describes how to automate the removal of obsolete log records from the Log Database. Database purging involves the periodic, automated execution of appropriate SQL statements within your SQL server. To enable automated purging:

1. Prepare SQL statements that remove log records.
2. Schedule automated execution of the SQL statements.

## Preparing SQL Statements

As you create SQL statements that delete records from the Log Database tables, keep in mind that these SQL statements must contain one or more criteria for selecting the log records you want to remove. You can base the selection criteria on the values of the log record fields, such as log record generation time, application name, host name, and so forth. For example, you might remove older log messages from the `G_LOG_MESSAGES` table and their corresponding attributes, if any, from the `G_LOG_ATTRS` table with the following SQL statements (in the order specified):

```
DELETE FROM G_LOG_ATTRS
WHERE LRID IN (SELECT G_LOG_MESSAGES.ID
   FROM G_LOG_MESSAGES
   WHERE (TIMEGENERATED > <start datetime>)
   AND (TIMEGENERATED < <end datetime>)
)
DELETE FROM G_LOG_MESSAGES
   WHERE (TIMEGENERATED > <start datetime>)
   AND (TIMEGENERATED < <end datetime>)
```

Refer to the Log Database structure description in “Database Format” on page 139 for more information about Log Database tables and fields. Combine the selection criteria to achieve the level of purging that suits your environment.

Check the Log Database Maintenance Wizard, available in SCI, for examples of the records-removal SQL statements that the Wizard prepares. The Wizard provides the graphical interface through which you specify various log-records selection criteria, and it displays the resulting SQL statements.

### Scheduling the Execution of SQL Statements

To enable automated purging of log records, schedule the periodic, automatic execution of the SQL statements you have prepared (for example, once a week). The simplest way to do this is to use either SQL server utilities or operating system services.

#### Using SQL Server Utilities

If you decide to use SQL Server utilities, refer to your SQL Server documentation to determine whether that server provides tools for automatic execution of SQL statements.

#### Using OS Services

If you decide to use scheduling tools available in your operating system, you should:

1. Prepare a command (either an executable file or a batch/shell file) that executes your SQL statement(s).
2. Use an operating system tool that enables you to schedule the specified command for execution.

To prepare a command that executes your SQL statement(s), use either a batch file or shell script. A command like this usually calls an SQL Server–specific tool to execute command-line SQL statements and passes an SQL statement to this tool as a parameter. For example, you can use the following tools to execute command-line SQL statements:

- `isql.exe` (a Microsoft SQL tool)
- `sqlplus` (an Oracle tool)

To schedule a specified command for execution with the required frequency, consider using these tools:

- `cron` on UNIX platforms.
- `at` on Windows platforms.

## How to Trace Interactions

You can trace interactions by using Interaction-level log records. You can view these logs using Genesys Administrator or SCI.

**Note:** The installation requirements for enabling the `Interaction view` are the same as for the Centralized Log. (See “How to View Centralized Logs” on page 60.)

## Procedure: Genesys Administrator—Tracing interactions

**Purpose:** To trace interactions by using Interaction-level log records.

### Prerequisites

- Management Layer components are installed and running.
- Centralized Logging is enabled.
- You are logged in to Genesys Administrator.

### Start of procedure

1. Select `Monitoring > Environment > Centralized Log`, and select the `Interaction` tab.
2. To view all Interaction-level log records for a specific application or host, do one of the following:
   - Filter these records by entering the name of the application or host in the `Application` or `Host` field, respectively.
   - Select `Provisioning > Environment > Applications or Hosts> <name of Application or Host object>`, and select the `Interaction` tab.

   For more information, refer to *Framework 8.1 Genesys Administrator Help*.

### End of procedure

## Procedure: Solution Control Interface—Tracing interactions

**Purpose:** To trace interactions by using Interaction-level log records.

### Prerequisites

- Management Layer components are installed and running.
- Centralized Logging is enabled.
- You are logged in to Solution Control Interface.

### Start of procedure

1. In SCI, select `Log` in the shortcuts bar, or expand `Centralized Log` in the Items tree.
2. Click `Interaction` to open a list of Interaction logs.

3. In this list, search the log records by their `Interaction ID` to display all records related to the same interaction, regardless of from which application each log event came.

   The `Details` pane in this view displays general information for the Interaction-level log record you select in the `List` view. It also displays extended attributes attached to the record, which might give additional details about a particular interaction.

**End of procedure**

# How to Set Up an Audit Trail and View Audit Logs

You can use Management Layer's centralized logging functionality to set up and view an audit trail.

## Setting Up the Audit Trail

Audit logs are log messages of type Standard or Trace that are marked as audit-related, and are logged in response to some action or event that requires an audit. To determine which logs are actually Audit logs, refer to *Framework 8.1 Combined Log Events Help*. This help file identifies Audit logs for each component.

To set up the Audit trail, use the log configuration option `verbose`, and set the output to `network` to ensure that the logs will be stored in the Log Database, ready for viewing.

### Standard-Level Audit Logs

To set up an Audit trail using only Standard-level Audit logs, configure the following options:

- In the `Application` objects representing the components that have Audit logs and for which you want to set up an audit trail:

  ```
  [log]
  verbose=standard
  standard=network
  ```

  and optional, but recommended:

  ```
  print-attributes=true
  ```

- In Message Server:

  ```
  [messages]
  db-storage=true
  ```

This will ensure that log events of Standard level will be stored in the Log Database, ready for viewing. Use the *Framework 8.1 Log Events Help* to identify which of the logs are Audit logs.

### Standard- and Trace-Level Audit Logs

**Warning!** Trace-level logging generates a significantly greater number of logs than Standard-level logging, and may affect the performance of your system.

To set up an audit trail with Standard- and Trace-level Audit logs, configure the following options:

- In the `Application` objects representing the components that have Audit logs and for which you want to set up an audit trail:

  ```
  [log]
  verbose=trace
  trace=network
  ```

  and optional, but recommended:

  ```
  print-attributes=true
  ```

- In Message Server:

  ```
  [messages]
  db-storage=true
  ```

This will ensure that log events of Standard, Interaction, and Trace level will be stored in the Log Database, ready for viewing. Use the *Framework 8.1 Log Events Help* to identify which of the Standard- and Trace-level logs are Audit logs.

For more information about the options used in setting up an audit trail, refer to the *Framework 8.1 Configuration Options Reference Manual*. For more information about log levels, refer to “Log Levels” on page 28.

## Viewing Audit Logs

You can view Audit logs using Genesys Administrator or SCI.

### Procedure: Genesys Administrator—Viewing Audit logs

**Start of procedure**

1. Select `Monitoring > Environment > Centralized Log`, and select the `Audit` tab.

2. To view all Audit log records for a specific application or host, do one of the following:
    - Filter these records by entering the name of the application or host in the `Application` or `Host` field, respectively.
    - Select `Provisioning > Environment > Applications or Hosts> <name of Application or Host object>,` and select the `Audit` tab.

    For more information, refer to *Framework 8.1 Genesys Administrator Help.*

**End of procedure**

### Procedure: Solution Control Interface—Viewing Audit logs

**Start of procedure**

1. In SCI, select `Audit` in the `View` shortcuts bar, or click `Audit` in the Items tree. A list of Audit logs, if any exist, appears in the List pane.
2. To view general information about a specific Audit log, select a record in the list. The `Details` pane displays general information for the Audit record you selected, and extended attributes attached to the record that might give additional details about a particular audited action.

**End of procedure**

## How to View Alarm History

**Note:** Alarm History can be viewed only in Solution Control Interface—not in Genesys Adminsitrator.

SCI enables you to view the Alarm-History records. There is one record for each Alarm triggered by Solution Control Server.

When you select an Alarm-History record, you can view general information about the record and details about the related Alarm, such as:

- Whether the alarm has been cleared.
- The reason for alarm clearance.

The `Alarm History` view provides a summary of alarms. SCI produces this summary based on the Alarm-related log events that SCS generates and stores in the Centralized Log Database. You can also view Alarms-related log events directly through the `Log` View in SCI.

# How to Configure Alarm Conditions and Alarm Reactions

Although you can configure Alarm Condition objects and Alarm Reaction scripts manually, using Genesys Administrator or Configuration Manager, the Management Layer provides an automated procedure.

You can configure new Alarm Conditions based on either source for their detection:

- Log Events—These Alarm Conditions trigger an alarm when an application or applications generate a specified log event.
- Alarm Detection Scripts—These Alarm Conditions trigger an alarm when a certain system variable changes in a specified manner.

As mentioned in "Alarm-Signaling Functions" on page 35, for alarms based on log events, alarm detection takes place in Message Server. Thus, you must configure your applications to connect to Message Server if you configure log event–based Alarm Conditions.

## Using Log Events for Alarm Detection

**Genesys Administrator**

To configure alarm conditions or alarm reactions in Genesys Administrator, create the alarm condition or alarm reaction under `Provisioning` > `Environment` > `Alarm Conditions`, being sure to specify the appropriate log event on the Configuration tab. Refer to *Framework 8.1 Genesys Administrator Help* for detailed instructions about creating the necessary objects.

**Solution Control Interface**

To configure alarm conditions or alarm reactions in Solution Control Interface, launch, respectively, the Alarm Condition Wizard or the Alarm Reaction Wizard accessible via SCI. You can launch either wizard from the main toolbar using the `Run a Wizard` button. You can also launch the Alarm Condition Wizard from shortcut menus of the log records displayed in the `Log View`. When started from a shortcut menu, the Alarm Condition Wizard takes the ID of a currently selected log record as a value for the `Detection Log Event ID`.

You can use the Alarm Condition Wizard to launch the Alarm Reaction Wizard and to associate Alarm Reactions with the Alarm Condition you are configuring. When launched from the shortcut menu of either a previously configured Alarm Condition or a set of Alarm Conditions, the Alarm Condition Wizard associates the selected Alarm Condition(s) with configured Alarm Reaction scripts.

For complete specifications of log events reported at the Alarm, Interaction, Standard, and Trace levels, see *Framework 8.1 Combined Log Events Help.* To access this Help file, refer to Appendix C on page 185. *Framework 8.1 Solution Control Interface Help* contains detailed instructions on how to start wizards. Also, see Chapter 10 on page 149, for more information about `Alarm Condition` objects preconfigured in the Configuration Database.

## Using Detection Scripts for Alarm Detection

Starting with release 7.0, Management Layer provides an additional alarm-detection mechanism, called *Advanced Alarm Detection*. Through this mechanism, you can configure Alarm Conditions:

- Based on the threshold for a system performance variable (CPU or memory usage).
- Based on the threshold for a local or remote SNMP variable (available only when you have enabled SNMP functionality).

Set alarms based on the Advanced Alarm-Detection methods using Alarm-Detection scripts. These scripts are Script configuration objects of the `Alarm Detection` type.

**Genesys Administrator**

In Genesys Administrator, create an alarm-detection script object at `Provisioning` > `Environment` > `Scripts`. Then specify it on the Configuration tab of the appropriate `Alarm Condition` object. For more information, refer to *Framework 8.1 Genesys Administrator Help*.

**Solution Control Interface**

Use the Alarm Detection Wizard (introduced in release 7.0) to configure alarm detection scripts. You can access this Wizard via SCI: launch it either from the main toolbar using the `Run a Wizard` button or from the Alarm Condition Wizard, which supports configuring Alarm Conditions for both Log-Based and Advanced Alarm Detection. If the latter, the Alarm Condition Wizard calls the Alarm Detection Wizard to configure an `Alarm Detection Script` object and associates it with the `Alarm Condition` object that you are currently configuring.

You must associate Alarm-Detection Scripts with Alarm Conditions. When an Alarm Condition object refers to an Alarm-Detection Script, the alarm detection for this Alarm Condition is performed according to the specified Alarm-Detection Script, regardless of whether any log event is specified as a `Detection Event`.

**Note:** To add a large number of Alarm Condition, Alarm Reaction, and Alarm Detection Script objects, consider using the Genesys Configuration Import Wizard. The Wizard helps you import such objects into the Configuration Database, either from an XML file that you create or from an XML file you export from another Configuration Database. Refer to the *Framework 8.1 Imported Configuration Data Formats Reference Manual* for details.

# Notes on Alarm Reaction Configuration

You can configure alarm reactions of the following types:

- Start a specified application.
- Stop a specified application.
- Restart the application that generated the alarm.
- Start a specified solution.
- Send an e-mail.
- Send an SNMP trap.
- Switch over to the backup application.
- Execute OS command.
- Change application option.

The configuration procedure for most of the alarm reactions is self-explanatory in the Alarm Reaction Wizard. You'll have to supply information for these configuration parameters:

- A unique name for the Alarm Reaction configuration object (for all types of alarm reactions).
- A name of the application or solution the alarm reaction is configured for (for alarm reactions of such types as Start a specified application, Stop a specified application, Start a specified solution, and Restart the application that generated the alarm).

You'll have to provide additional information for alarm reactions of the following types:

- Switchover (see recommendations on page 72).
- Send an e-mail (see recommendations on page 73).
- Send an SNMP trap (see recommendations on page 73).
- Execute OS command (see recommendations on page 73).
- Change application option (see recommendations on page 76).

Starting with release 7.0, the Management Layer allows execution of Alarm Reactions not only at Alarm activation, but also at Alarm clearance. To achieve this, add the Alarm Reactions that are to be executed when the Alarm is cleared to the Clearance Scripts list of the corresponding Alarm Condition object. You can also use the Alarm Condition Wizard to accomplish this.

## Alarm Reactions of the Switchover Type

**Warning!** You must have a high-availability (HA) license to enable Solution Control Server to successfully process an alarm reaction of the Switchover type. The lack of the license prevents the switchover between primary and backup applications of any type.

When configuring an alarm reaction of the `Switchover` type, you can specify whether Solution Control Server should perform the switchover when an application, which generates an alarm, is running in a particular mode:

- Select `primary` if you want SCS to perform a switchover only if the application that has generated an alarm is currently operating in Primary mode.
- Select `backup` if you want SCS to perform a switchover only if the application that has generated an alarm is currently operating in Backup mode.
- Select `perform switchover always` if you want SCS to perform a switchover regardless of the operating mode of the application that generates the alarm.

You might use these options, for instance, when associating an alarm reaction of the `Switchover` type for T-Server with the `CTI Link Disconnected` log event. Selecting `primary` for the alarm reaction configuration may prevent an unwanted switchover if the T-Server that produced this log event currently operates in Backup mode.

## Alarm Reactions of the E-Mail Type

To configure an alarm reaction of type `Send an E-Mail`, specify the recipients of the e-mail in the Alarm Reaction Wizard. Then, compose the subject and text of the e-mail message, by using reserved variables.

See *Framework 8.1 Solution Control Interface Help* for detailed instructions for configuring the e-mail script, and an example. See Chapter 6 for more information about the e-mail interface itself.

## Alarm Reactions of the SNMP Trap Type

To configure an alarm reaction of the `Send an SNMP Trap` type, specify a Name for the Alarm Reaction configuration object. All necessary information is automatically provided by SCS, given that the SNMP Master Agent application is configured correctly.

See Chapter 7 on page 93, for more information about enabling the SNMP alarm signaling.

## Alarm Reactions of the OS Command Type

To configure an Alarm Reaction of the `Execute OS Command` type, specify the name of the operating system command that is to be executed when an alarm is detected. If necessary, include the full path to the executed command.

**Warning!** Although you can specify any valid command name, use alarm reactions of this type with caution. To avoid unauthorized actions, limit access to Solution Control Server and Solution Control Interface to the administrators' group.

SCS executes all alarm reactions. In the case of an alarm reaction of the `Execute OS Command` type, SCS executes the specified command on its own host computer. Therefore, a currently logged in user must have sufficient permissions to execute the specified operating system command.

SCS passes information about a detected alarm to the operating system command to be executed. For this purpose, SCS adds command-line arguments (listed in Table 4) to the command line you specify in the `Command` property when you configure the alarm reaction.

**Note:** Some applications started as a result of the `Execute OS Command` alarm reaction may not recognize the command-line arguments added by SCS. This means that these applications might not work properly in this circumstance; for example, they might exit. To make them work, you can call such applications indirectly; for instance, from within a script that passes correct command-line parameters to these applications. You then specify name of this script in the `Command` property of the alarm reaction.

### Table 4: Additional Command-Line Parameters

| Parameter | Description |
|---|---|
| -msgid | ID of the log event that resulted in the alarm |
| -msgtext | Text of the log event that resulted in the alarm |
| -condid | Alarm Condition `ID` |
| -condname | Alarm Condition `Name` |
| -conddesc | Alarm Condition `Description` |
| -appid | ID of the application that generated the log event that resulted in the alarm |
| -appname | Name of the application that generated the log event that resulted in the alarm |
| -hostname | Name of the application host that generated the log event that resulted in the alarm |

### Examples

The following are three examples using Alarm Reactions of `Execute OS Command` type to filter specific alarms:

**Filtering by type of log event**

To log occurrences of certain log events to a database other than the Genesys Log Database, create a `*.bat` file that provides logging in to your independent database. Name this file `process_alarm.bat`. Then using the Alarm Reaction Wizard, configure an Alarm Reaction of the `Execute OS Command` type and set the `Command` property to:

```
"/home/Genesys/SCServer/scripts/process_alarm.bat"
```

When a corresponding alarm is detected, SCS executes the following operating system command:

```
"/home/Genesys/SCServer/scripts/process_alarm.bat"
-msgid 20002 -msgtext "CTI Link disconnected" -condid 103 -condname
"CTI Link Failure" -conddesc "Failure of connection between any
T-Server and a switch." -appid 120 -appname "T-Server_Application"
-hostname "NameOfHost"
```

**Note:** If a specified operating system command normally would result in a screen display, the alarm reaction is performed, but the screen output cannot be enabled and, therefore, cannot be seen.

**Collecting information about alarms**

A script called `react_script.sh`, for bash UNIX shell, saves information about each alarm in the `reaction.log` text file located in the `/home/genesys/logs/` directory:

```
echo `date|cut -c4-16` : msgid=$2 : msgtext=$4 : condid=$6 :
condname=$8 :
conddesc=${10} : appid=${12} : appname=${14} >>
/home/genesys/logs/reactions.log
```

**Filtering by the host that generated the alarm:**

To save information about alarms generated by a specific host, you must create a script that writes only those logs generated by a specific host. The following sample script, `HostA_alarms.sh` for the bash UNIX shell, writes only those alarms generated by `HostA` to the `HostA_reactions.log` text file located in the `/home/genesys/logs/` directory.

```
while [ 0 -lt "$#" ]; do

case "$1" in
 "-msgid") shift; MSGID="$1" ;;
 "-msgtext") shift; MSGTEXT="$1" ;;
 "-condid") shift; CONDID="$1" ;;
 "-condname") shift; CONDNAME="$1" ;;
 "-conddesc") shift; CONDDESC="$1" ;;
 "-appid") shift; APPID="$1" ;;
 "-appname")shift; APPNAME="$1" ;;
 "-hostname")shift; HOSTNAME="$1" ;;
esac
shift
done
```

```
if [ $HOSTNAME = "HostA" ]; then
echo ·date|cut -c4-16· : msgid=$MSGID : msgtext=$MSGTEXT :
condid=$CONDID : condname=$CONDNAME : conddesc=$CONDDESC :
appid=$APPID : appname=$APPNAME >> HostA_reactions.log
fi
```

### Alarm Reactions of the Type Change Application Option

To configure an Alarm Reaction of the type `Change Application Option`, specify:

- The name of the application for which you want to change a value of a configuration option.

**Note:** If you don't specify the application name, the Management Layer updates the option configuration of the application that triggers the alarm.

- The name of the configuration section to which the option belongs.
- The name of the configuration option the value for which you want to change.
- The new value to which to set the configuration option.

**Warning!** The account under which you run SCS must have appropriate permissions for the application whose option configuration you want to change.

You might configure this alarm reaction, for example, so that an application changes its log level to a more detailed one once the application reports the first signs of a critical situation in a particular log event.

## How to Avoid an Unnecessary Switchover

Starting with release 7.0.1, you can minimize the chance that a network problem causes a switchover between a functioning primary server and its backup. When disconnected from an LCA running on any host, SCS initiates a switchover for all applications running on the same host with the LCA. However, the disconnect can result from either long-term issues (such as the host being down or LCA terminating) or temporary issues (such as slowness of the network or a temporary network problem).

You can configure SCS to verify the connection status in a few seconds to confirm whether the connection issue is resolved. To do so, create the `disconnect-switchover-timeout` option in the `general` section on the `Options` tab of the SCS `Properties` dialog box. Set the option value to any positive integer, which means the number of seconds and depends on your typical network conditions. When SCS initiates a switchover process, it waits for the interval you specified and checks the LCA connection:

- If the problem has been temporary, the connection is restored and the applications on the LCA host are in running status. In this case, SCS does not perform a switchover.
- If the problem is serious, the LCA remains disconnected and the status of the applications on the LCA host is unknown. In this case, SCS proceeds with the switchover.

**Note:** The `disconnect-switchover-timeout` option setting has no effect on a manual switchover, a switchover resulting from an alarm reaction, or a switchover resulting from service unavailability at the primary server.

Refer to the *Framework 8.1 Configuration Options Reference Manual* for detailed option descriptions.

# How to Check the Configuration

You can also use the Management Layer to perform an automatic check of the configuration and to generate a report that lists potential configuration problems and suggested solutions. The *Configuration Checker* (available only in SCI) verifies the configuration stored in the Configuration Database against various rules and conditions.

To check the existing configuration via the Management Layer, launch the Configuration Checker from the SCI's main toolbar using the `Run a Wizard` button. Then analyze the generated Check Report to fill configuration gaps and make all functions operable.

**Note:** The Configuration Checker projects how to maximize the Management Layer functionality for the existing configuration objects. If you don't plan to use a particular Management Layer function or if you don't plan to use a redundant component for this function, ignore the Configuration Checker warnings for it when analyzing the Check Report.

Refer to *Framework 8.1 Solution Control Interface Help* for detailed instructions for starting the Configuration Checker.

# How to Troubleshoot the Management Layer

Use the *Management Layer Troubleshooter* (available only in SCI) to troubleshoot Management Layer functions that do not work. The Troubleshooter presents the most commonly asked questions about enabling Management Layer functionality and suggestions for what actions to undertake in a particular situation.

The Troubleshooter presents information about five areas of Management Layer functionality:

- Alarming and alarm reactions
- Logging
- Status monitoring
- Solutions and applications management
- Distributed solution management

Launch the Management Layer Troubleshooter with the `Run a Wizard` button on the main toolbar of SCI. Select a problem summary that describes the issue you are experiencing and follow the on-screen instructions to find and correct the cause of the problem.

Refer to *Framework 8.1 Solution Control Interface Help* for detailed instructions for starting the Troubleshooter.

In addition, Chapter 11 on page 165, contains suggestions for identifying and handling the most common mistakes made when enabling the Management Layer functionality.

# How to Manage Environments that Use Two Configuration Servers

The Management Layer is designed to operate with one Configuration Server (one primary-backup pair of Configuration Servers). That is, you must connect the Management Layer components and all the applications that it controls to the same Configuration Server.

If you need to use the Management Layer capabilities in an environment with two Configuration Servers (two primary-backup pairs of Configuration Servers), you must use an independent Management Layer installation for each Configuration Server (each primary-backup pair). To make two Management Layer installations independent when a host computer serves two configurations, install two Local Control Agent applications on such a computer, one LCA communicating to one Management Layer installation and the other LCA communicating to the other, and make the LCA port numbers unique.

## How to Manage Environments That Use Configuration Server Proxy

The Management Layer fully supports the geographically distributed configuration environment available when using Configuration Server Proxy.

**Note:** Starting with release 7.0, the term *Configuration Server Proxy* is used to identify a Configuration Server instance running in so-called *Proxy* mode. Refer to the *Framework 8.1 Deployment Guide* for more information.

This support means that the Management Layer:

1. Displays the current real-time status of Configuration Server Proxy and performs its startup, shutdown, or automatic switchover to the backup application just as for any other Genesys application.

   **Note:** You cannot manually switch over Configuration Server Proxy applications.

2. Correctly starts applications that are clients of Configuration Server Proxy.

When receiving a request to start an application, the Management Layer determines whether the application is configured as a client of Configuration Server or of Configuration Server Proxy. For this purpose, the Management Layer checks the list of connections configured for an application.

The application is considered a client of Configuration Server Proxy when both of these conditions are met:

- The application's configuration contains a connection to an application of the Configuration Server type.
- In its turn, the application of Configuration Server type contains in its configuration a connection to another application of the Configuration Server type.

The application is considered a client of Configuration Server when either of these conditions is met:

- The application's configuration contains no connection to an application of the Configuration Server type.
- The application's configuration does contain a connection to an application of the Configuration Server type, but this latter application's configuration does not contain a connection to an application of the Configuration Server type.

**Note:** Genesys recommends that you configure connections to Configuration Server for applications that are clients of Configuration Server in an environment with Configuration Server Proxy.

If the Management Layer finds the application to be a client of Configuration Server, the Management Layer uses the Configuration Server parameters to start the application. For more information, see "Processing the Start Command for Applications" on page 50.

If the Management Layer finds the application to be a client of Configuration Server Proxy, the Management Layer also checks the configuration to determine whether a backup application is configured for this Configuration Server Proxy:

- If no backup application is configured, the stand-alone Configuration Server Proxy is considered to be running in Primary mode.
- If a backup application is configured, the Management Layer identifies which Configuration Server Proxy of the primary-backup redundancy pair is currently running in Primary mode.

Then, the Management Layer starts the application that is a client of Configuration Server Proxy. SCS generates a command line and passes it to LCA, which executes the command. The command line contains:

- The application's working directory as specified in the `Application` object's `Properties.`
- The host name of Configuration Server Proxy currently running in Primary mode.
- The port number of Configuration Server Proxy currently running in Primary mode.
- The application name as specified in the `Application` object's `Properties.`

**Note:** Make sure that Configuration Server Proxy is running during its client startup

Chapter

# 5 Stuck Calls Management

This chapter describes new functionality for handling stuck calls, including the various methods used to detect and handle them.

This chapter contains the following sections:

## Overview

A stuck call occurs when information about the release of a call in the communication system fails to reach one or more of the components of a CTI solution. One possible cause is the temporary loss of communication between CTI applications and devices, such as switching and interactive voice response systems, in the communication infrastructure.

Having missed the call release information, CTI applications continue to treat such calls as active, which results in less efficient operation and inaccurate reporting.

Because T-Servers are directly involved in communications with the switching systems, they play a critical role in detecting and handling stuck calls. This chapter describes the procedures related to detecting and ways of dealing with calls that appear to be stuck.

Stuck calls can be handled by any of three methods:

1. Using the T-Server switch-specific functionality (page 84)
2. Using the SNMP interface in the Management Layer (page 85)
3. Using the gstuckcalls utility in the Management Layer (page 86)

# Which Method To Use?

## T-Server Switch-Specific Functionality

This method offers stuck calls detection and cleanup built-in to T-Server.

This is the basic form of using the stuck call feature in T-Server that provides for minimum customization and management options from the user's perspective.

Characteristics:

- A simple, timeout-based detection mechanism is used internally in T-Server.
- This method does not utilize Management Layer capabilities—no automatic reactions to be executed upon detection of stuck calls.
- You must set up and manage each T-Server manually and individually, using Genesys Administrator or Configuration Manager.
- Unwieldy for managing multiple T-Servers.

## SNMP Interface in the Management Layer

This method provides an SNMP interface, good for SNMP-based installations and in an SNMP-oriented application.

Characteristics:

- Relies on external SNMP-aware applications (such as SNMP Perl scripts) to monitor and detect stuck calls in T-Server.
- Stuck call detection logic is highly customizable; information such as filters and timestamp properties lies in the new SNMP tables.
- The script provides for a central point of management, and can be tailored to manage a single or multiple T-Server applications.
- Does not utilize the Management Layer capability to monitor and react to alarm events; the script must do it.
- Requires SNMP licensing.

## gstuckcalls Utility

This method has the advantages of the second method but does not require SNMP.

Characteristics:

- The stuck call detection logic is highly customizable.
- This method is integrated with the Management Layer. A stuck calls event can be configured and reacted upon when corresponding log messages are received by SCS.

- This method does not require an SNMP license.
- The scripts in this utility generate an XML file with a summary of all calls retrieved from T-Server, which makes it useful as a quick-look diagnostic tool.

The gstuckcalls utility uses the logmsg utility, which allows sending specified log messages to trigger and clear Management Layer alarms based on conditions external to the Management Layer.

See "Using the gstuckcalls Utility" on page 86.

# Prerequisites

## Perl

The Stuck Calls functionality requires that Perl be installed on the SCS host computer. Table 5 lists the names and minimum versions of Perl extension modules required. Users may need to install some or all of them, depending on their current Perl installation.

**Table 5: Perl Extension Modules**

| Module | Recommended version |
|---|---|
| HTML::Parser | 3.25 and higher |
| SOAP::Lite | 0.60 and higher |
| XML::DOM | 1.43 and higher |
| XML::Parser | 2.34 and higher |
| XML::SAX | 0.12 and higher |
| XML::NamespaceSupport | 1.08 and higher |
| XML::RegExp | 0.03 and higher |
| HTTP::Cookies | 1.39 and higher |

These modules are available from the Comprehensive Perl Archive Network (CPAN) website:

`http://www.CPAN.org.`

The Framework 8.1 stuck calls functionality—including the Perl scripts `GStuckCallsDetect.pl` and `GStuckCallsClear.pl` and the above modules—were tested using Perl version 5.6.1.

### SOAP

The Perl scripts GStuckCallsDetect.pl and GStuckCallsClear.pl require Configuration Server to start with the SOAP (Simple Object Access Protocol) port enabled, a setting which is not the default in the Configuration Server configuration file. Refer to the *Framework 8.1 Configuration Options Reference Manual*.

## Using T-Server

To support the stuck calls handling in T-Server, a set of configuration options has been introduced. These options control stuck call detection, notification, and automatic cleanup. For more information, refer to the “T-Server Common Configuration Options” chapter in the latest version of the Deployment Guide for your T-Server.

To support the stuck calls handling in the Management Layer, a set of log messages have been added to the T-Server Common Part. See “T-Server Common Log Events” in *Framework 8.1 Combined Log Events Help* for more information. To access this Help file, refer to Appendix C on page 185.

To support the stuck calls handling in client applications of T-Server, a new property has been added to the T-Server events that define the end of the call (EventReleased and EventAbandoned). See the latest version of the *Genesys Events and Models Reference Manual* for more information.

Based on a specified timeout, T-Server waits for a call information being updated. After the timeout is expired, T-Server considers a call as a stuck call and reports a standard log message.

Processing of timeouts and notifications is implemented in the T-Server Common Part, but the actual call cleanup involves interaction with the switch-dependent part for each T-Server.

### Configuration Options Summary

Three new options must be configured in the section call-cleanup.

#### notify-idle-tout

This option specifies the time interval that T-Server waits for a call being updated from its last update. After this time elapses, if no new events about the call are received, T-Server reports this call as a stuck call.

#### cleanup-idle-tout

This option specifies the time interval that T-Server waits for a call being updated from its last update. After this time elapses, if no new events about the call are received, T-Server clears this call as a stuck call either by querying the switch (if a CTI link provides such capabilities), or by deleting call information from memory unconditionally. The option description in the latest version of the Deployment Guide for each T-Server reflects the actual implementation for that particular T-Server.

**periodic-check-tout**

This option specifies the time interval for periodic checks for stuck calls (affects both notification and cleanup functionality) by checking the T-Server's own call information with call information available in the switch. For performance reasons, T-Server does not verify whether the `notify-idle-tout` or `cleanup-idle-tout` option has expired before performing this checking.

### T-Server Common Log Events

Three T-Server common log events support stuck calls management: `01-20020`, `01-20021`, and `01-20022`. Refer to the *Framework 8.1 Combined Log Events Help* for detailed specifications of these log events. To access this Help file, refer to Appendix C on page 185.

### EventReleased on special DN

The value of the `TReliability` parameter indicates that the update was forced by external request:

```
TReliabilityExternal = 3
TReliabilityExternal - cleared by an external SNMP request
```

## Using the SNMP Interface

The following tables in the T-Server-Specific SNMP Objects support management of stuck calls using the SNMP Interface in the Management Layer: `tsCallFilterTable` and `tsCallInfoTable`.

These tables allow you to retrieve only those call instances that were defined by the filters in the `tsCallFilterTable` table thus reducing network traffic and increasing application performance.

- `tsCallFilterTable` provides the interface for setting call filter criteria for the `tsCallInfoTable` table. Also provides the interface for clearing calls by the call's Connection ID. See Table 14 on page 108.
- `tsCallInfoTable` stores the latest snapshot of active calls from a given T-Server, contains information about active calls filtered by conditions set in the `tsCallFilterTable`. See Table 15 on page 109.

# Using the gstuckcalls Utility

The gstuckcalls utility contains two stuck calls management scripts, `GStuckCallsDetect.pl` and `GStuckCallsClear.pl`, which support the detection and automatic clearing of stuck calls. Both scripts use the `gstuckcallsscript.cfg` configuration file, also part of the utility.

## How to Install the Utility

If you installed Solution Control Server, the gstuckcalls utility containing the stuck calls scripts and configuration file is already installed, and is located in the same folder in which Solution Control Server was installed.

Starting in 8.1.2, you can install the Solution Control Server utilities without installing Solution Control Server itself. If you have not installed the utilities, use the procedure "Installing the Solution Control Server Utilities" in the *Framework Deployment Guide*. After the utilities are installed, the gstuckcalls utility is stored in the location that you specified during the installation.

## How to Run the Utility

To start the stuck calls utility:

- In Windows, enter `gstuckcalls.exe` on the command line.
- On UNIX, enter `gstuckcalls` on the command line.

The utility will automatically run the scripts.

### GStuckCallsDetect.pl script

The `GStuckCallsDetect.pl` script performs these functions:

1. Retrieves all the information about all T-Servers from the configuration.
2. Queries each T-Server for stuck calls according to the specified filter using the `gstuckcalls` utility.
3. If stuck calls are found, sends log message 9500 on behalf of the T-Server.
4. If stuck calls are not found, sends log message 9501 on behalf of the T-Server.

If you require alarming for stuck calls, schedule this script for periodic execution, by using tools provided by your operating system, such as `Scheduled Tasks` for Windows and `Cron` for UNIX.

### GStuckCallsClear.pl script

The `GStuckCallsClear.pl` script clears stuck calls in the specified T-Server. If you need to clear stuck calls automatically, use this script as an alarm reaction for the active alarm `Stuck Calls Detected`. This script performs the following:

1. Connects to the specified T-Server.
2. Uses the `gstuckcalls` utility to clear all stuck calls according to the specified filter.

### gstuckcallsscript.cfg Configuration File

The `GStuckCallsDetect.pl` and `GStuckCallsClear.pl` scripts use this configuration file. It has the following format:

```
[cfgserver]
host=<host>
port=<port>
username = <username>
password = <password>

[msgserver]
host=<host>
port=<port>

[filter]
createdbefore=<seconds>
createdafter=<seconds>
updatedbefore=<seconds>
updatedafter=<seconds>
```

## Stuck Calls Alarm Log Messages

The following alarm log messages have been added to support detection and automatic clearance of stuck calls:

```
09500: Stuck calls detected
09501: Stuck calls not detected
```

## Configuring the Alarm Condition

To enable automatic stuck calls detection, configure the corresponding Alarm Condition with the following settings:

For `Detect Event`:

- `Log Event ID` set to `9500`
- `Selection Mode` set to `Select By Application Type`

- `Type` set to `T-Server`

For `Cancel Event`:

- `Log Event ID` set to `9501`

See “Using Log Events for Alarm Detection” on page 70 for more information.

The active alarm `Stuck Calls Detected` is communicated when log message `9500` is received. This happens when the `GStuckCallsDetect.pl` script detects stuck calls in a T-Server. Scheduling the `GStuckCallsDetect.pl` script for periodic execution (for instance, once per day) ensures automatic stuck calls detection and alarming.

To clear stuck calls automatically, follow these steps:

1. Configure the alarm reaction type `Execute OS Command` for the Alarm Condition `Stuck Calls Detected`.
2. Configure this alarm reaction to execute the `GStuckCallsClear.pl.` script.

The script clears stuck calls at the corresponding T-Server and sends log message `9501` to the Management Layer, which then clears the active alarm `Stuck Calls Detected`.

## Stuck Calls Scripts Flow Chart

Figure 2 provides a flow chart to help you better understand the scripts.

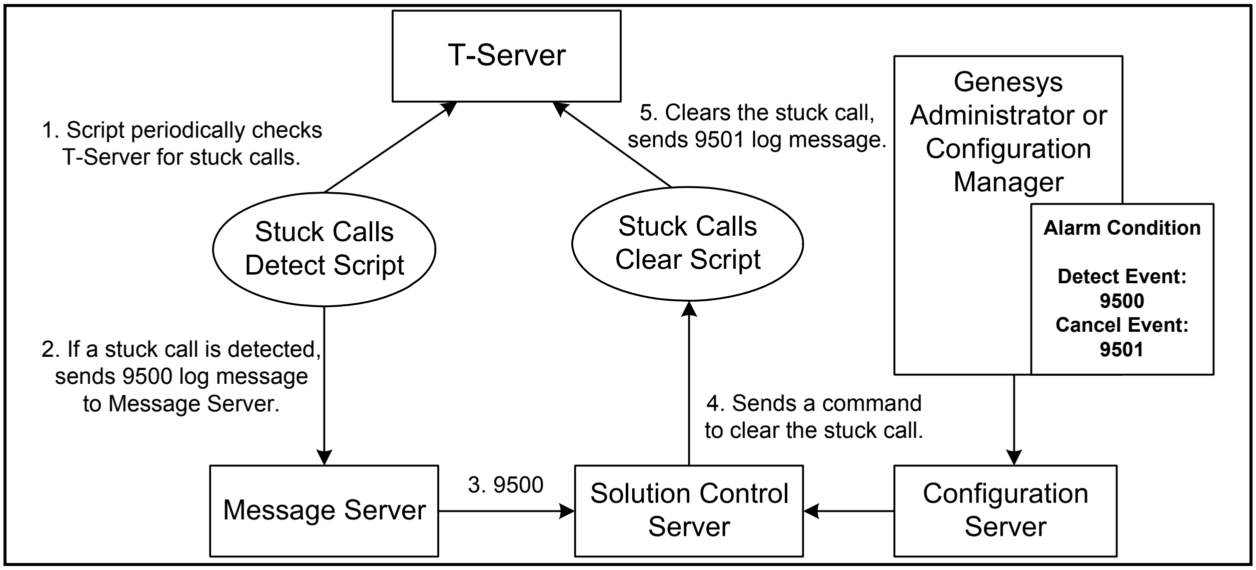

T-Server
1. Script periodically checks T-Server for stuck calls.
5. Clears the stuck call, sends 9501 log message.
Genesys Administrator or Configuration Manager
Stuck Calls Detect Script
Stuck Calls Clear Script
Alarm Condition
Detect Event: 9500
Cancel Event: 9501
2. If a stuck call is detected, sends 9500 log message to Message Server.
4. Sends a command to clear the stuck call.
Message Server
3. 9500
Solution Control Server
Configuration Server

**Figure 2: Stuck Calls Script Flow Chart**

Chapter

# 6 E-Mail Alarm-Signaling Interface

This chapter describes how the Management Layer processes alarm reactions of the *E-Mail* type and how to configure an e-mail system for this function.

This chapter contains the following sections:

## Alarm Reactions of the E-Mail Type

You can configure the Management Layer to send the content of an alarm as an e-mail message to one or more e-mail addresses. Simply create an alarm reaction of the `Send an e-mail` type for a corresponding alarm condition. See "How to Configure Alarm Conditions and Alarm Reactions" on page 70 for recommendations on configuring alarm reactions.

**Note:** An *alarm* is a message generated by a Genesys application when a certain alarm condition is met. For more information, refer to "Alarm-Signaling Functions" on page 35.

Figure 3 on page 90 illustrates the message flow through the Management Layer when such an alarm is triggered. That flow includes the e-mail system for your environment, which you must configure for the host running Solution Control Server.

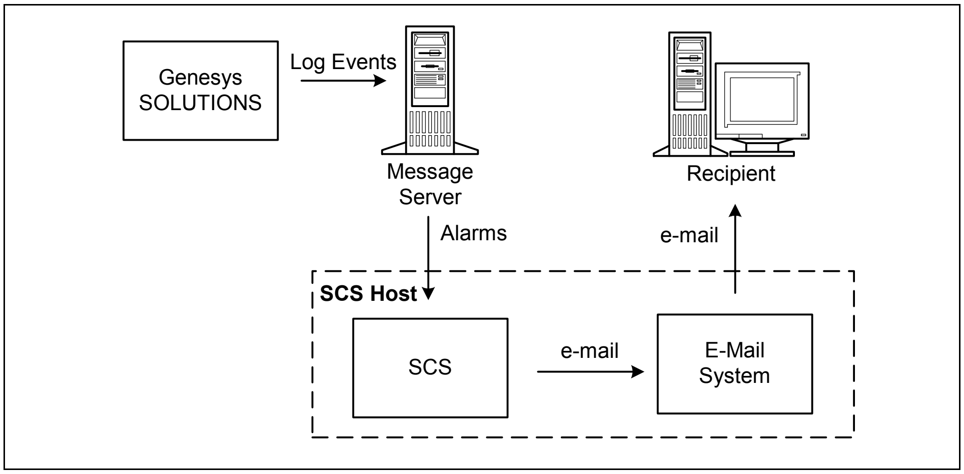


**Figure 3: E-Mail Alarm Reaction in Management Layer**

# Configuring E-Mail Systems

This section describes how to configure a UNIX- and a Windows-based e-mail systems to send the content of an alarm message in an e-mail.

## On UNIX

On UNIX operating systems, Solution Control Server can use either the `sendmail` command or Simple Mail Transfer Protocol (SMTP) to send e-mail messages. Depending on the protocol you prefer:

- You must correctly configure the `sendmail` command on the host computer running SCS.
- You must configure SMTP server host and port for SCS as values for the `smtp_host` and `smtp_port` configuration options.

**Note:** For more information about Solution Control Server configuration options, refer to the *Framework 8.1 Configuration Options Reference Manual.*

## On Windows

On Windows operating systems, Solution Control Server can use either the Messaging Application Programming Interface (MAPI) or Simple Mail Transfer Protocol (SMTP).

## MAPI

To enable the operation of e-mail alarm reactions via an MAPI e-mail system, you must install the system and configure it properly on the host computer running Solution Control Server. Also install Microsoft's CDO (Collaboration Data Objects).

The simplest way to set an MAPI e-mail system is to install Microsoft Outlook on the host computer for SCS. SCS then uses the default MAPI profile to connect to the system and send messages.

If you have installed SCS and are running it as a regular application (as opposed to a Windows Service), SCS uses the credentials of the user who is currently logged into Windows to open the default MAPI profile. So you must set permissions that allow users to use the default MAPI profile.

If you have installed SCS as a Windows Service, you must explicitly specify a user account to log on to Windows for this service. That account must have sufficient permissions to use the default MAPI profile.

## Procedure: Setting up a user account to access SCS as a Windows Service

**Purpose:** To create a user account to specify when installing SCS as a Window Service, and that has sufficient permissions to use the default MAPI profile.

### Prerequisites

- Solution Control Server has been installed and configured.

### Start of procedure

1. Open the `Control Panel` window.
2. Double-click the `Services` button to open the `Service` window.
3. Select the `Genesys Solution Control Server` service.
4. Click the `Startup` button.
5. Select `This Account` and specify user account information.
6. Click `OK` and close the `Service` window.

### End of procedure

## SMTP

To enable the operation of e-mail alarm reactions via an SMTP e-mail system, you must configure the `mailer` section. Specify the SMTP server host for SCS as the value for the configuration option `smtp_host`, and specify the SMTP server port for SCS as the value for the configuration option `smtp_port`.

**Note:** For more information about Solution Control Server configuration options, refer to the *Framework 8.1 Configuration Options Reference Manual.*

Genesys

Chapter

# 7 SNMP Interface

This chapter describes Management Layer built-in support for network management systems (NMS) that comply with the Simple Network Management Protocol (SNMP). It also describes how to activate this function. In particular, this chapter focuses on how the Management Layer distributes SNMP commands from an NMS and how it processes alarm reactions of the *SNMP Trap* type. It also describes the layout of the Genesys Management Information Base (MIB) file and the format of the SNMP traps, including the abbreviations for the Genesys application types.

This chapter contains the following sections:

## Built-in SNMP Support

The Management Layer provides a built-in support for SNMP-compliant third-party NMS. Solution Control Server (SCS) processes various NMS commands and generates SNMP traps based on changes in the current status of an individual application. With this built-in support for SNMPv1–v3, you can access Management Layer functions through your existing NMS interface.

**Note:** The Genesys built-in SNMP implementation for SNMPv1 passed all the tests developed and published by CERT/CC for this sort of application. For information about tests with which you can check your system against vulnerability to SNMPv1 malformed SNMP packets, go to `http://www.cert.org/advisories/CA-2002-03.html`.

The following subsections describe the architecture of the SNMP support. The Management Layer provides you, as a network administrator, with three ways to monitor and control Genesys products via an NMS user interface:

- You can start, stop, and monitor the status of any Genesys or third-party application that the Management Layer monitors and controls. In addition, you can modify log options for Genesys server applications.
- You can retrieve application-specific SNMP statistics and data as defined in the MIB file for those Genesys server applications that support application-specific SNMP requests.
- You can receive alarms from any Genesys server application in the form of SNMP traps.

With all three options, the communications between SCS and NMS require an SNMP master agent application that is compliant with the AgentX protocol. If your NMS does not contain such an application, you can use Genesys SNMP Master Agent to integrate the Management Layer into your NMS. The Genesys MIB file, which the NMS uses, defines the communication interface between the Management Layer and the NMS. Refer to Appendix D on page 189 for more information about the Genesys MIB file.

## SNMP Command Processing

Figure 4 on page 95 illustrates how the Management Layer processes the SNMP commands it receives from an NMS. The commands include:

- Start and stop commands for any Genesys or third-party application that the Management Layer monitors and controls.
- Change of log options settings for Genesys server applications.

With this architecture, you can also:

- View the configuration of any Genesys or third-party product that the Management Layer monitors and controls.
- Monitor the current status of an application (see if it is running or not) and, for redundant configurations, view the current redundancy mode (Primary or Backup) of a running application.
- View the configuration and status of any host registered as a `Host` object in the Configuration Database, including the LCA configuration of that host.
- View configured solutions and their statuses (see if solutions are running).

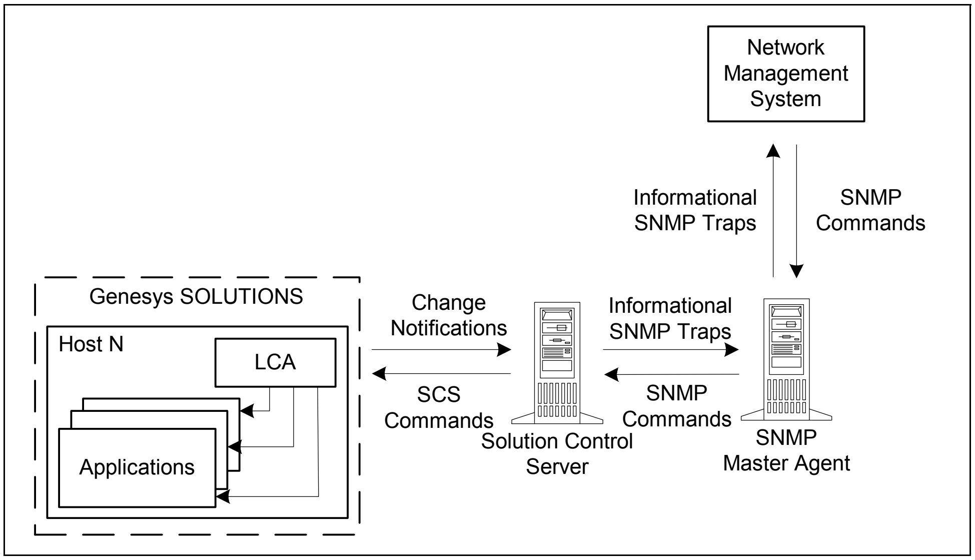


**Figure 4: Management Layer Processing of an SNMP Command from an NMS**

## Requesting SNMP Data

In addition to its application-monitoring functions, you can use the Management Layer to retrieve some SNMP data particular to applications of a given type. For example, you can request from T-Server the number of calls it is currently handling.

You can only retrieve SNMP data and prompt application-specific SNMP traps for applications built with the Genesys management library, such as:

- Call Concentrator
- Configuration Server
- DB Server
- Universal Routing Server
- T-Server

Figure 5 on page 96 illustrates how these applications interact with an NMS. As you can see, all requests from the NMS as well as data and traps from the applications come through Solution Control Server.

Consult product-specific documentation to see if your product supports SNMP.

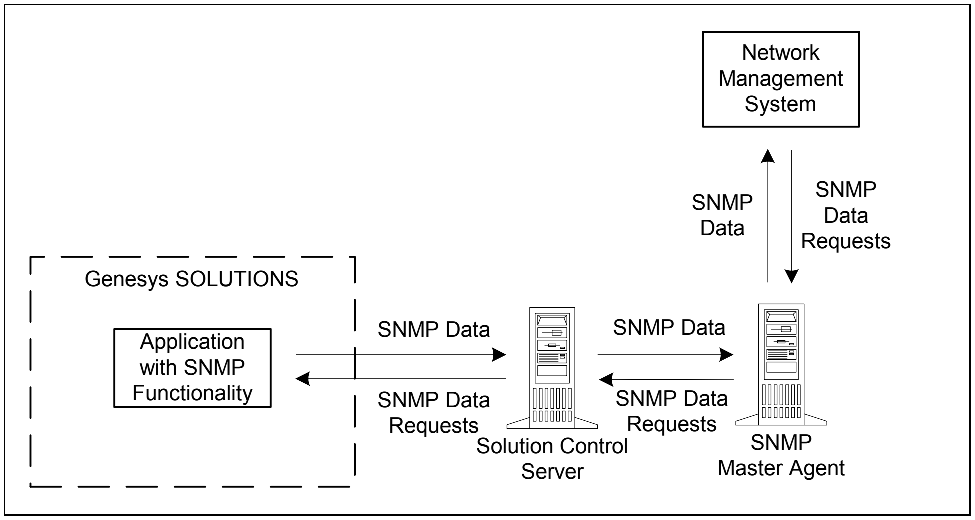


**Figure 5: SNMP Information Exchange Between Some Servers and NMS**

## Alarms and SNMP Trap Processing

To transmit the content of an alarm message to an SNMP-compliant third-party NMS, the Management Layer converts that information into an SNMP trap. An *alarm* is a message generated by a Genesys application when a certain Alarm Condition is met. For more information about alarm signaling, see Chapter 3 on page 25, and either *Framework 8.1 Genesys Administrator Help* or *Framework 8.1 Solution Control Interface Help*, as appropriate.

Figure 6 illustrates how the Management Layer reacts to an alarm of type `Send an SNMP trap`.

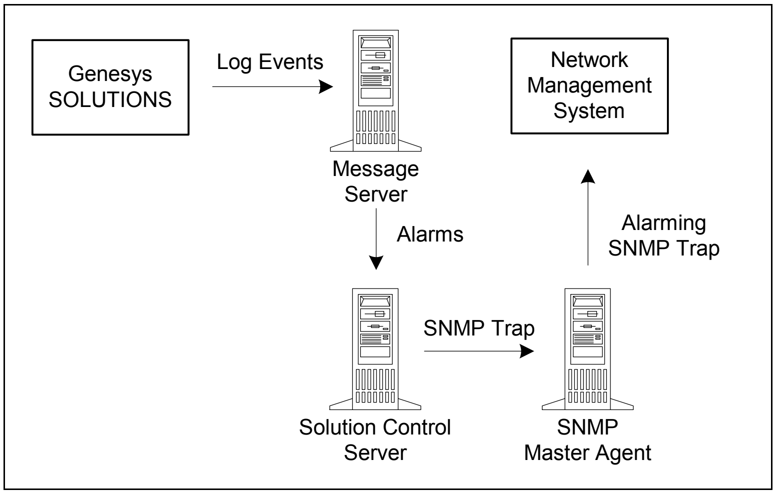


**Figure 6: Management Layer Processing of an SNMP Trap Alarm Reaction**

# SNMP-Managed Objects

The Genesys MIB file contains all SNMP objects available to the NMS.

Starting with release 7.0, Genesys MIB utilizes the SMI-v2 Row-Status mechanism and the control/data-tables concept to facilitate management of multiple Genesys servers simultaneously.

RowStatus—a TEXTUAL-CONVENTION defined in IETF’s SNMPv2-TC—is commonly used to control the dynamic creation and deletion of rows in SMIv2 defined tables.

A conceptual SMIv2 table using Row-Status also acts as a control table or configuration table. Using the control table, the NMS application configures the information to be monitored. A separated data table holds the information that is gathered.

One control entry (one row) is linked to the data that is gathered. Each control entry contains control parameters that specify which data or statistics you want to access and collect.

Genesys MIB 7.0 uses one control table and several data tables. Data tables are organized based on their functional areas and divided into two main groups: *server-generic* data tables and *server-specific* data tables.

Each data table, whether it is server-generic or server-specific, is assigned a unique identifier. (Refer to TableID Textual Convention in the Genesys MIB file, for the complete list of tables and their identifiers.)

This table identifier along with the Genesys server identifier (`server DBID` number) is used as an index in the control table. Thus, each row in the control table gathers particular data from a particular Genesys server.

Starting with release 7.0.1, you can enable an automatic refresh of MIB tables. When you set up a row to gather data from a particular table, you have a choice of automatic or manual refresh for the table. If you select `Automatic Refresh` mode, specify the time period, in seconds, after which Solution Control Server is to refresh the table.

In addition to control and data tables, you can access a group of server standard objects independently of the control/data-table mechanism.

The following sections present information about each object set:

- “Standard SNMP Objects” on page 98 describes the standard Genesys SNMP objects.
- “The gServerControlTable Table” on page 99 describes the Control Table objects.
- “T-Server-Specific SNMP Objects” on page 107 describes the SNMP objects specific to T-Server.

For information about supported traps, see “SNMP Traps” on page 114.

# Standard SNMP Objects

Tables in this section describe standard Genesys SNMP objects by object group.

## The gServersTable Table

Table 6 describes objects that belong to `gServersTable`, which contains information about server environments.

### Table 6: gServersTable Table of Standard SNMP Objects

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| gsCleanupTimeout | Unsigned 32 | read/write | The time, in minutes, the agent should keep rows in the gsControlTable and consequently in related data tables if there were no requests to objects of this row or corresponding rows from data table(s). After the timeout, the agent should automatically delete unattended rows. Value 0 set for this object specifies that MIB clean up should not be performed. |
| gServersTable | sequence | read | Specifies a sequence of the following objects. |
| gServerId | integer | read | Uniquely identifies a Genesys server. Corresponds to the number assigned to an object in the Configuration Database to identify the object among all objects of the same type. The gpServerCurrent object uses this value to switch from one Genesys server to another. |
| gServerName | string | read | Specifies the application name of a server application as configured in the Configuration Database. |
| gServerStatus | string | read | Specifies the current operational status of a server. The possible settings are UP or DOWN, which indicates if the server is running or not. |
| gServerType | string | read | Indicates the type of a server; that is, the application type specified for this application in the Configuration Database. |
| gServerVersion | string | read | Specifies the current version of the running server; that is, the application version specified for this application in the Configuration Database. |

**Table 6: gServersTable Table of Standard SNMP Objects (Continued)**

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| gServerWorkDir | string | read | Specifies the server's working directory; that is, the working directory specified for this application in the Configuration Database. |
| gServerCommandLine | string | read | Indicates the full command line used to start this server, as specified in the Configuration Database. For example:<br>scs -host host1 -port 4135 -app SCS_Primary |
| gServerPID | string | read | Specifies the process ID of the server that is currently running. |
| gServerCommand | integer | read/write | Specifies the command to start, shut down or gracefully shut down a server. Accepts the following values:<br>• 1 start<br>• 2 shutDown<br>• 3 shutDownGracefully |
| gServerDeleteClient | integer | read/write | Sends a delete-client command to the server. Enter the socket number of a client as the value |

## The gServerControlTable Table

Table 7 describes objects that belong to `gServerControlTable`, which configures the information to be monitored and controls the data-refresh process.

**Table 7: gServerControlTable Table of Standard SNMP Objects**

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| gServerControlTable | sequence | read | Specifies a sequence of the following objects. |
| gsCtrlServerID | integer | not-accessible | An index. Specifies the DBID of the server to be managed by this control row. The valid DBID number used to set rows in this control table is retrieved from the gServersTable. |

### Table 7: gServerControlTable Table of Standard SNMP Objects (Continued)

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| gsCtrlTableID | integer | not-accessible | An index. Specifies the data to be gathered for this server. Valid values and the tables they represent are as follows:<br>• gsLogTable(1)<br>• gsInfoTable(2)<br>• gsClientTable(3)<br>• gsPollingTable(4)<br>• tsInfoTable(5)<br>• tsCallTable(6)<br>• tsDtaTable(7)<br>• tsLinkTable(8)<br>• tsCallFilterTable(9)<br>• tsCallInfoTable(10)<br>• tsLinkStatsTable(11)<br>Detailed information about these tables is provided in the subsequent sections. |
| gsCtrlRefreshStatus | integer | read-only | Indicates refresh status of corresponding data table as specified by gsCtrlTableID. The following refresh statuses are reported:<br>• dataNotReady(1)<br>• dataRefreshInProgress(2)<br>• dataReady(3)<br>• mgmtIsNotAvailable(4)<br>• dataRefreshFailed(5)<br>Refer to the Genesys MIB file for detailed descriptions of these statuses. |

**Table 7: gServerControlTable Table of Standard SNMP Objects (Continued)**

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| gsCtrlLastRefreshed | timetick | read-only | Specifies the time in hundredths of seconds since the row was last successfully refreshed. |
| gsCtrlRowStatus | RowStatus | read/create | Controls and manages row creation and row deletion. Initiates data-refresh process for a data table managed by this control row, and reports the status of this row. Refer to the Genesys MIB file for a detailed description of the way this object is manipulated. |

## The gsInfoTable Table

Table 8 describes objects that belong to `gsInfoTable`, which contains miscellaneous statistics and data about a managed server.

**Table 8: gsInfoTable Table of Standard SNMP Objects**

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| gsInfoTable | sequence | read | Specifies a sequence of the following objects. Indexed by the gsCtrlServerID. |
| gsClientsExistNum | Unsigned32 | read | Indicates the number of clients currently connected to a server. |
| gsClientsTotalNum | Unsigned32 | read | Indicates the total number of clients connected so far to a server. |
| gsServerConfigFile | string | read | Indicates configuration file name, if any, used to start a server. |

## The gsPollingTable Table

Table 9 on page 102 describes objects that belong to `gsPollingTable`, which specifies a *heart-beat* feature; that is, a periodic signal sent over a network to the NMS.

### Table 9: gsPollingTable Table of Standard SNMP Objects

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| gsPollingTable | sequence | read | Specifies a sequence of the following objects. Indexed by the gsCtrlServerID. |
| gsPollingID | Unsigned32 | read/write | Specifies the amount by which each polling signal increases over the last one. The initial polling event equals the same integer. For more information about this variable, see Table 20 on page 114. |
| gsPollingInterval | Unsigned32 | read/write | Specifies the interval, in seconds, between two subsequent polling signals sent from a server. May be set to any integer. |
| gsPollingLastTrap | string | read | Specifies the last trap value of a polling signal sent to the NMS. For more information about this variable, see Table 20 on page 114. |
| gsPollingStatus | string | read/write | Activates or deactivates the server polling feature. Values are ON and OFF. Value ON causes a server to send periodic SNMP signals to SCS, which, in turn, converts these signals into SNMP traps. |

## The gsLogTable Table

Table 10 on page 105 describes objects that belong to gsLogTable, which contains information about log option settings. Values of the SNMP objects in this table correspond to values of configuration options specified in the log section in an Application object's Options. Check information in the Description column for the name of the particular option to which an object corresponds.

### Permission Requirements

To change log option settings for a particular application via SNMP, you must first use Configuration Manager to associate both the Application object and the Solution Control Server Application object with the account in the Configuration Database having permissions to modify configuration object properties. In other words, the account must have Change permissions for Application object(s).

**Note:** You can only set up this functionality by using Configuration Manager. It is not currently available via Genesys Administrator.

## Procedure: Configuration Manager—Associating Solution Control Server with a user account that has application change permissions

**Purpose:** To associate Solution Control Server with an account that has permissions to change Application objects. This association will enable you to change log option settings via SNMP.

### Prerequisites

- Management Layer components are installed and running.
- The Solution Control Server Application object has been installed and configured.
- Configuration Manager is started.

### Start of procedure

1. Log into Configuration Manager under a user account having Full Control permissions.
2. Go to the SCS Application Properties window in Configuration Manager.
3. On the Security tab, change the Log On As setting to This Account and select any user account (Person object). This can be one of the following:
   - An account that belongs to the Administrators default access group.
   - An account that belongs to the role-specific Access Group with the Change permissions you have created for this purpose.
   - An individual account to which you will grant Change permissions in any Access Group.
4. Click OK to save configuration changes.

### End of procedure

## Procedure: Configuration Manager—Associating other applications with the same account designated for Solution Control Server

**Purpose:** To associate applications with the same account you designated in the procedure "Configuration Manager—Associating Solution Control Server with a user account that has application change permissions" on page 103 for SCS.

### Prerequisites

- Management Layer components are installed and running.
- The Solution Control Server Application object has been associated with an account with application change permissions. Use the procedure "Configuration Manager—Associating Solution Control Server with a user account that has application change permissions" on page 103.
- You are logged in to Configuration Manager.

### Start of procedure

1. Select the Application object which you will be associating with the account the selected in the procedure on page 103. You can also select a folder or subfolder that contains Application objects, in which case permissions change for all Application objects in the selected folder.
2. Open the Application object's Properties window (or folder's Properties window) and click the Security tab.
3. Click the Permissions command button and add the user account you designated as This Account for SCS to this object's permissions. Be sure to set Type of Access to Change for this account.
4. Click OK to save changes in the Add dialog box, Permissions dialog box, and Application Properties window (or folder Properties window).

You could grant change permissions for Application objects to the SYSTEM account, but doing so (and making all servers connect to Configuration Server with change permissions) might impact data security.

### End of procedure

### Table 10: gsLogTable Table of Standard SNMP Objects

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| gsLogTable | sequence | read | Specifies a sequence of the following objects. Indexed by `gsCtrlServerID`. |
| logVerbose | string | read/write | Log level. Filters output of messages by their priorities. Corresponds to the `verbose` log option. |
| logTrace | string | read/write | Lists the set of log outputs for the log messages of the Trace level. Corresponds to the `trace` log option. |
| logStandard | string | read/write | Lists the set of log outputs for the log messages of the Standard level. Corresponds to the `standard` log option. |
| logDebug | string | read/write | Lists the set of log outputs for the log messages of the Debug level. Corresponds to the `debug` log option. |
| logAll | string | read/write | Lists the set of log outputs for the log messages of all levels. Corresponds to the `all` log option. |
| logBuffering | string | read/write | Turns on/off OS file buffering. Buffering increases performance of file output; however, log messages may appear in the log with a delay after they have been logged. Corresponds to the `buffering` log option. |
| logSegment | string | read/write | Sets the mode of log output segmentation. Currently implemented only for the file output. When a currently opened log segment exceeds the size set by this option, the current segment is closed and a new one is created (an empty new segment). Corresponds to the `segment` log option. |
| logExpire | string | read/write | Sets the expiration mode for old files (segments); that is, specifies whether to remove old files when new ones are created. Corresponds to the `expire` log option. |
| logMessageFile | string | read/write | Sets the name of the file that defines log messages specific to applications of this type. Corresponds to the `messagefile` log option. |

### Table 10: gsLogTable Table of Standard SNMP Objects (Continued)

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| logMessageFormat | string | read/write | Specifies the format of log record headers that an application uses when writing logs in the log file. Corresponds to the `message_format` log option. |
| logTimeFormat | string | read/write | Specifies how to represent in a log file the time when an application generates log records. Corresponds to the `time_format` log option. |
| logTimeConvert | string | read/write | Specifies in which system an application calculates the log record time when generating a log file. Corresponds to the `time_convert` log option. |

## The gsClientTable Table

Table 11 describes objects that belong to `gsClientTable`, which gathers statistics about server clients.

### Table 11: gsClientTable Table of Standard SNMP Objects

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| gsClientTable | sequence | read | Specifies a sequence of the following objects. Indexed by the gsCtrlServerID. |
| gsClientAppName | string | read | Specifies the client’s application name. |
| gsClientAuthorized | string | read | Specifies the client’s level of authorization. |
| gsClientGotEvents | Unsigned32 | read | Specifies the number of events the client has received. |
| gsClientSentReqs | Unsigned32 | read | Specifies the number of requests the client has sent. |
| gsClientSocket | Unsigned32 | read | Specifies the socket number through which the client is connected to the server. |
| gsClientType | integer | read | Specifies the client’s type. |

## The gsAlarmObjects Table

Table 12 on page 107 describes objects that belong to `gsAlarmObjects`, which specifies how the Management Layer converts the alarms it generates to SNMP traps and sends them to the NMS. Table 22 on page 118 lists Genesys application types as they appear in alarm-related traps.

**Table 12: gsAlarmObjects Table of Standard SNMP Objects**

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| gsServersLastAlarm | string | read | Specifies the last trap value sent to the NMS. For information about traps that use this variable, see Table 20 on page 114. |
| gsServersLastTrap | string | read | Specifies the last server-status (server up or down) trap sent to NMS. |
| gsAlarmID | Unsigned32 | read | The unique identifier of the Alarm Condition name as configured in the Configuration Database. |
| gsAlarmLogText | string | read | The text of the log event that triggered this alarm. |
| gsAlarmMessagesIds | string | read | The unique identifier of the log event that triggered or removed this alarm. |
| gsAlarmApplicationName | string | read | The name of the application that reported this alarm as specified in the Configuration Database. |
| gsAlarmApplicationType | string | read | The type of application that reported this alarm as specified in the Configuration Database. |
| gsAlarmCategory | string | read | The alarm category as specified in the Configuration Database: Critical, Major, or Minor. |

# T-Server-Specific SNMP Objects

The following tables summarize the SNMP objects specific to T-Server. These objects give you access to internal T-Server tables that contain information about call states, addresses, and CTI links.

## The tsInfoTable Table

Table 13 on page 108 describes objects that belong to tsInfoTable, which collects miscellaneous data and statistics specific to T-Server.

### Table 13: tsInfoTable Table of T-Server-Specific SNMP Objects

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| tsInfoTable | sequence | read | Specifies a sequence of the following objects. Indexed by the gsCtrlServerID. |
| tsCallsExistNum | Unsigned32 | read | Specifies the current number of calls being handled by T-Server. |
| tsCallsTotalNum | Unsigned32 | read | Specifies the number of calls T-Server has handled since it started. |
| tsLinksCommand | string | read/write | Specifies the command to be sent to T-Server.<br>**Note:** Reserved for future use. |
| tsLastChangedLink-Status[a] | string | read | Specifies the server name, link name, and link's new status.<br>**Note:** Reserved for future use. |

a. This object is specific to environments with X.25 links.

## The tsCallFilterTable

Table 14 supports stuck calls functionality by describing objects that belong to tsCallFilterTable, which provides the interface for setting call filter criteria for tsCallInfoTable (see Table 15 on page 109), and for cleaning calls by their ConnectionID.

### Table 14: tsCallFilterTable Table of T-Server-Specific SNMP Objects

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| gsCtrlServerId | ServerDBID | not-accessible | Uniquely identifies a T-Server application. |
| fltCallCreatedBefore | Unsigned32 | read-write | Reports the calls that were created earlier than a specified number of seconds counting from the time of the request. The 0 (zero) value means the filter is not used. |

**Table 14: tsCallFilterTable Table of T-Server-Specific SNMP Objects (Continued)**

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| fltCallCreatedAfter | Unsigned32 | read-write | Reports the calls that were created later than a specified number of seconds counting from the time of the request. |
| fltCallUpdatedBefore | Unsigned32 | read-write | Reports the calls that were last time updated earlier than a specified number of seconds counting from the time of the request. |
| fltCallUpdatedAfter | Unsigned32 | read-write | Reports the calls that were last time updated later than a specified number of seconds counting from the time of the request. |
| clearCallByConnId | DisplayString | read-write | Connection ID (converted to string by connid_to_str function) of the call to be cleared. |

## The tsCallInfoTable Table

Table 15 supports stuck calls functionality by describing objects that belong to `tsCallInfoTable`, which stores the latest snapshot of active calls from a given T-Server, and contains a set of attributes that facilitates the discovery of stuck calls.

**Table 15: tsCallInfoTable Table of T-Server-Specific SNMP Objects**

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| gsCtrlServerId | ServerDBID | not-accessible | Uniquely identifies a T-Server application. |
| callInfoInstanceID | Unsigned32 | not-accessible | Reports the call instance ID. |
| callInfoType | Unsigned32 | read-only | Reports a call type. |
| callInfoCreationTimestamp | Unsigned32 | read-only | Reports a call creation timestamp. |
| callInfoLastUpdatedTimestamp | Unsigned32 | read-only | Reports a timestamp of the last update on this call. |
| callInfoInternalParties | DisplayString | read-only | Reports an Internal DN. |

## The tsLinkStatsTable Table

Table 16 on page 110 describes information about the current statistics for messages sent and received over a T-Server link.

### Table 16: tsLinkStatsTable Table of T-Server-Specific SNMP Objects

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| gsCtrlServerId | ServerDBID | not-accessible | Uniquely identifies a T-Server application. |
| LinkID | Unsigned32 | read-only | Uniquely identifies a T-server link to which these statistics apply. |
| timeElapsedSec | Unsigned32 | read-only | The time (in seconds) that have elapsed since the last statistics were measured. |
| numberMessagesTx | Unsigned32 | read-only | The number of CTI messages that have been sent by T-Server over this link since the last statistics were measured. |
| rateMessagesTx | Unsigned32 | read-only | The rate at which CTI messages are sent by T-Server over this link.<br>Literally, the ratio of numberMessagesTx to timeElapsedSec. |
| numberMessagesRx | Unsigned32 | read-only | The number of CTI messages that have been received by T-Server over this link since the last statistics were measured. |
| rateMessagesRx | Unsigned32 | read-only | The rate at which CTI messages are received by T-Server over this link.<br>Literally, the ratio of numberMessagesRx to timeElapsedSec. |

## The tsCallTable Table

Table 17 describes objects that belong to `tsCallTable`, which contains data about telephony calls being processed by T-Server.

### Table 17: tsCallTable Table of T-Server-Specific SNMP Objects

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| tsCallTable | sequence | read | Specifies a sequence of the following objects. Indexed by gsCtrlServerID and callInstanceID. |

### Table 17: tsCallTable Table of T-Server-Specific SNMP Objects (Continued)

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| callANI | string | read | Automatic Number Identification. Provides calling party information (typically, the telephone number or billing account number) to the called party. |
| callCallID | string | read | Specifies the current call identifier that the switch has assigned to a call. |
| callConnID | string | read | Specifies the identifier that T-Server has assigned to a call. |
| callCustomerID | string | read | Specifies the Customer (Tenant) identifier used when a call was initiated. |
| callDNIS | string | read | Directory Number Identification Service. Identifies to the called system the last three or four digits of the number actually dialed by the caller. |
| callFirstTransferDN | string | read | Specifies the DN on a remote T-Server from which a call was first made. |
| callFirstTransfer-Location | string | read | Specifies the location of the remote T-Server from which a call was first transferred. |
| callInstanceID | counter | read | Specifies the instance number for each call. |
| callLastTransferDN | string | read | Specifies the DN on a remote T-Server from which a call was last transferred. |
| callLastTransfer-Location | string | read | Specifies the location of the remote T-Server from which a call was last transferred. |
| callNumParties | string | read | Specifies the number of parties currently involved in a call. |
| callPartiesList | string | read | Specifies a list of parties involved in a call. |
| callReferenceID | string | read | Specifies the reference ID of a call. |
| callTimeStamp | string | read | Specifies the timestamp of when the call was created, in seconds starting from January 1, 1970. |
| callType | string | read | Specifies the type of a call. |

Table 17: tsCallTable Table of T-Server-Specific SNMP Objects (Continued)

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| callState | string | read | Specifies the current state of the call in question. |

## The tsDtaTable Table

Table 18 describes objects that belong to tsDtaTable, which holds information about all registered DNs.

Table 18: tsDtaTable Table of T-Server-Specific SNMP Objects

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| tsDtaTable | sequence | read | Specifies a sequence of the following objects. Indexed by the gsCtrlServerID and tsDtaInstanceID. |
| tsDtaDigits | string | read | Specifies the digits field of the DTA structure. |
| tsDtaInstanceID | counter | read | Specifies the instance field of the DTA structure. |
| tsDtaMode | string | read | Specifies the mode field of the DTA structure. |
| tsDtaState | string | read | Specifies the state field of the DTA structure. |
| tsDtaType | string | read | Specifies the type field of the DTA structure. |

## The tsLinkTable Table

Table 19 on page 112 describes objects that belong to tsLinkTable, which contains information about the CTI links that exist between T-Server and switch(es) and the links' attributes.

Table 19: tsLinkTable Table of T-Server-Specific SNMP Objects

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| tsLinkTable | sequence | read | Specifies a sequence of the following objects. Indexed by the gsCtrlServerID and tsLinkID. |
| tsLinkAddress | string | read | Specifies the address of the link. |

**Table 19: tsLinkTable Table of T-Server-Specific SNMP Objects (Continued)**

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| tsLinkDelay[a] | string | read | Specifies a link reconnect delay in the case of an unsuccessful attempt to reconnect to the line. |
| tsLinkDTEClass[a] | string | read | Specifies the DTE class for the X.25 connection. |
| tsLinkName | string | read | Specifies the name of the link. |
| tsLinkID[a] | integer | read | Specifies the link identifier.<br>**Note:** Reserved for future use. |
| tsLinkMode[a] | string | read | Specifies the mode of the link. |
| tsLinkPID[a] | string | read | Specifies the link's process ID. |
| tsLinkPort | string | read | Specifies the physical port number of the link. |
| tsLinkProtocol | string | read | Specifies a protocol type. |
| tsLinkSocket[a] | string | read | Specifies the socket number through which the server is connected to the link.<br>**Note:** Reserved for future use. |
| tsLinkStatus | string | read | Specifies the current status of the link. The *status* property is used in fault monitoring, providing the NMS with information about which links are currently established and which links have lost physical or logical connection.<br>Here are the valid values:<br>• `0` Link is not configured properly.<br>• `1` No physical or TCP/IP connection between T-Server and the switch.<br>• `2` Both physical and logical connections are OK.<br>• `3` Physical connection exists between T-Server and switch, but the logical connection is missing. |
| tsLinkTemplate | string | read | Specifies a template for the connection. |

**Table 19: tsLinkTable Table of T-Server-Specific SNMP Objects (Continued)**

| Object Name | Value Type | Access Level | Description |
|---|---|---|---|
| tsLinkX25Device[a] | string | read | Specifies the X.25 device connected to the link. |
| tsLinkX25LocalAddress[a] | string | read | Specifies the local address for an X.25 connection. |

[a] This object is specific to environments with X.25 links.

## SNMP Traps

This section discusses the SNMP trap messages you can receive from Genesys applications:

- Table 20 on page 114 lists SNMP traps that only server applications built with the management library can generate. (See the list on page 95).
- Table 21 on page 117 lists alarm-related SNMP traps that SCS generates on behalf of any Genesys server it monitors.

**Table 20: Application-Generated SNMP Traps**

| Trap | Variable Name | String Format and Valid Values | Description |
|---|---|---|---|
| gsServerUpTrap | gsServersLastTrap | String format:<br>`<server name>:"server is UP"` | Reports that a server was down but is running again.<br>The `<server name>` in the message is the server's application name in the Configuration Database.<br>For example:<br>tserver_1:server is UP |
| gsServerDownTrap | gsServersLastTrap | String format:<br>`<server name>:"server is DOWN"` | Reports that a server has gone down.<br>The `<server name>` in the message is the server's application name in the Configuration Database.<br>For example:<br>tserver_1:server is DOWN |

### Table 20: Application-Generated SNMP Traps (Continued)

| Trap | Variable Name | String Format and Valid Values | Description |
|---|---|---|---|
| gsAlarm | gsServersLastAlarm | String format:<br>`<server name>:`<br>`0:<alarm code>:`<br>`<alarm message>` | Reports that a server has encountered an alarm situation. A server itself generates this trap. Do not confuse with gSmlSCserverAlarm which is generated by Solution Control Server based on log events it receives from other servers. |
| tsLinkStatusTrap | tsLastChangedLinkStatus | String format:<br>`server <server name>`<br>`(<server number>) -`<br>`link <link name>`<br>`status changed -`<br>`<0, 1>` (<DOWN,UP>) | Reports that T-Server link status has been changed. |

### Table 20: Application-Generated SNMP Traps (Continued)

| Trap | Variable Name | String Format and Valid Values | Description |
|---|---|---|---|
| gsPollingSignal | gsPollingLastTrap | String format:<br>`<server name>:<value>`<br>Valid Value:<br>Any positive integer | Sent in response to a polling signal from the server. If the polling ID status is on (see Table 9 on page 102), the server's polling ID increases by a set amount every time the server sends the polling signal. The server then sends the new value to SCS. The new value becomes the value inside of the variable gpPollingLastTrap, which SCS sends to NMS as the trap gpPollingSignal.<br>The `server name` in the message is the server's application name in the Configuration Database.<br>For example:<br>`tserver_1:122`<br>This message means that the last SNMP polling signal, which the T-Server application named tserver_1 sent to the SCS, was number 122. |

**Table 21: Alarm-Related SNMP Trap**

| Trap | Variable Name | String Format and Valid Values | Description |
|---|---|---|---|
| gsMLAlarm | gsAlarmId | integer | The unique identifier of the Alarm Condition name as configured in the Configuration Database. |
| | gsAlarmLogText | string | The text of the log event that triggered this alarm. |
| | gsAlarmMessagesIds | string | The unique identifier of the log event that triggered or removed this alarm. |
| | gsAlarmApplicationName | string | The name of the application that reported this alarm as specified in the Configuration Database. |
| | gsAlarmApplicationType | string | The type of application that reported this alarm as specified in the Configuration Database. |
| | gsAlarmAppHostName | string | The host name of the application that reported this alarm. |
| | gsAlarmCategory | string | The alarm category as specified in the Configuration Database: Critical, Major, or Minor. |
| | gsAlarmGUID | string | The unique identifier of the Alarm Clearance Trap with Alarm Creation Trap. |

## Application Types in SNMP Traps

Table 22 on page 118 lists application types for server applications—their database IDs and abbreviations—as these types are displayed in alarm-related SNMP traps. The table also presents the application type names as displayed in the Configuration Layer.

### Table 22: Application Type Representations

| Type ID | Type Abbreviation | Type Name |
|---|---|---|
| 01 | TServer | T-Server |
| 01 | N/A | Programmable Gateway Framework |
| 02 | StatServer | Stat Server |
| 03 | BillingServer | Billing Server |
| 06 | VoiceTreatmentServer | Voice Treatment Server |
| 08 | DBServer | Database Access Point |
| 09 | CallConcentrator | Call Concentrator |
| 10 | SDialer | CPD Server |
| 11 | ListManager | List Manager |
| 12 | OSServer | Outbound Contact Server |
| 15 | RouterServer | Universal Routing Server |
| 21 | ConfigurationServer | Configuration Server |
| 23 | ThirdPartyServer | Third Party Server |
| 26 | DARTServer | Obsolete |
| 28 | CustomServer | Custom Server |
| 29 | ExternalRouter | External Router |
| 31 | VirtualRP | Virtual Routing Point |
| 32 | Database | Obsolete |
| 33 | NetVector | Web Option |
| 34 | DetailBiller | Detail Biller |
| 35 | SummaryBiller | Summary Biller |
| 36 | NACD | Network Overflow Manager |
| 37 | BackUpControlClient | Backup Control Client |
| 38 | InfomartStatCollector | CC Analyzer Data Sourcer |
| 40 | IVRInterfaceServer | IVR Interface Server |

### Table 22: Application Type Representations (Continued)

| Type ID | Type Abbreviation | Type Name |
|---|---|---|
| 41 | IServer | I-Server |
| 42 | MessageServer | Message Server |
| 43 | SCS | Solution Control Server |
| 45 | SNMPAgent | SNMP Agent |
| 46 | RealDBServer | DB Server |
| 48 | WFMDataAggregator | WFM Data Aggregator |
| 50 | WFMScheduleServer | WFM Schedule Server |
| 52 | ETLProxy | ETL Proxy |
| 54 | GVCServer | GVP-Voice Communication Server |
| 55 | VSSSystem | VSS System |
| 58 | CCAnalyzerDataMart | CC Analyzer Data Mart |
| 59 | ChatServer | Chat Server |
| 60 | CallbackServer | Callback Server |
| 61 | CoBrowsingServer | Co-Browsing Server |
| 62[a] | SMSServer | SMS Server |
| 63 | ContactServer | Contact Server |
| 64 | EmailServer | E-Mail Server |
| 65 | MediaLink | MediaLink |
| 66 | WebInteractionRequestsServer | Web Interaction Requests Server |
| 67 | WebStatServer | Web Stat Server |
| 68 | WebInteractionServer | Web Interaction Server |
| 69 | WebOptionRoutePoint | Web Option Route Point |
| 74 | VoIPController | Voice over IP Controller |
| 77 | HAProxy | High Availability Proxy |
| 78 | VoIPStreamManager | Voice over IP Stream Manager |

**Table 22: Application Type Representations (Continued)**

| Type ID | Type Abbreviation | Type Name |
|---|---|---|
| 79 | VoIPDMXServer | Voice over IP DMX Server |
| 80 | WebAPIServer | Web API Server |
| 81 | LoadBalancer | Load Balancer |
| 82 | ApplicationCluster | Application Cluster |
| 83 | LoadDistributionServer | Load Distribution Server |
| 84 | GProxy | G-Proxy |
| 85 | GIS | Genesys Interface Server |
| 86 | AgentDesktopDeliveryServer | GCN Delivery Server |
| 88 | IVRDT | IVR DirectTalk Server |
| 89 | GCNThinServer | GCN Thin Server |
| 90 | ClassificationServer | Classification Server |
| 91 | TrainingServer | Training Server |
| 92 | UniversalCallbackServer | Universal Callback Server |
| 93 | CPDServerProxy | CPD Server Proxy |
| 94 | XLinkController | XLink Controller |
| 95 | KWorkerPortal | K-Worker Portal |
| 96 | WFMServer | WFM Server |
| 97 | WFMBuilder | WFM Builder |
| 98 | WFMReports | WFM Reports |
| 99 | WFMWeb | WFM Web |
| 100 | KnowledgeManager | Knowledge Manager |
| 101 | IVRDriver | IVR Driver |
| 102 | IVRLibrary | IVR Library |
| 103 | LCSAdapter | LCS Adapter |
| 104 | DesktopNETServer | Desktop NET Server |

### Table 22: Application Type Representations (Continued)

| Type ID | Type Abbreviation | Type Name |
|---|---|---|
| 105 | Siebel7ConfSynchComponent | Siebel7 ConfSynchComponent |
| 106 | Siebel7CampSynchComponent | Siebel7 CampSynchComponent |
| 107 | GenericServer | Generic Server |
| 108 | GenericClient | Generic Client |
| 109 | CallDirector | Call Director |
| 110 | SIPCommunicationServer | SIP Communication Server |
| 111 | InteractionServer | Interaction Server |
| 112 | IntegrationServer | Integration Server |
| 113 | WFMDaemon | WFM Daemon |
| 114 | GVPPolicyManager | GVP Policy Manager |
| 115 | GVPCiscoQueueAdapter | GVP Cisco Queue Adapter |
| 116 | GVPTextToSpeechServer | GVP Text To Speech Server |
| 117 | GVPASRLogManager | GVP ASR Log Manager |
| 118 | GVPBandwidthManager | GVP Bandwidth Manager |
| 119 | GVPEventsCollector | GVP Events Collector |
| 120 | GVPCacheServer | GVP Cache Server |
| 121 | GVPASRLogServer | GVP ASR Log Server |
| 122 | GVPASRPackageLoader | GVP ASR Package Loader |
| 123 | GVPIPCommunicationServer | GVP IP Communication Server |
| 124 | GVPResourceManager | GVP Resource Manager |
| 125 | GVPSIPSessionManager | GVP SIP Session Manager |
| 126 | GVPMediaGateway | GVP Media Gateway |
| 127 | GVPSoftSwitch | GVP Soft Switch |
| 128 | GVPCoreService | GVP Core Service |
| 129 | GVPVoiceCommunicationServer | GVP Voice Communication Server |

### Table 22: Application Type Representations (Continued)

| Type ID | Type Abbreviation | Type Name |
|---|---|---|
| 130 | GVPUnifiedLoginServer | GVP Unified Login Server |
| 131 | GVPCallStatusMonitor | GVP Call Status Monitor |
| 132 | GVPReporter | GVP Reporter |
| 133 | GVPH323SessionManager | GVP H323 Session Manager |
| 134 | GVPASRLogManagerAgent | GVP ASR Log Manager Agent |
| 135 | GVPGenesysQueueAdapter | GVP Genesys Queue Adapter |
| 136 | GVPIServer | GVP IServer |
| 137 | GVPSCPGateway | GVP SCP Gateway |
| 138 | GVPSRPServer | GVP SRP Server |
| 139 | GVPMRCPTTSServer | GVP MRCP TTS Server |
| 140 | GVPCCSServer | GVP CCS Server |
| 141 | GVPMRCPASRServer | GVP MRCP ASR Server |
| 142 | GVPNetworkMonitor | GVP Network Monitor |
| 143 | GVPOBNManager | GVP OBN Manager |
| 144 | GVPSelfServiceProvisioningServer | GVP Self Service Provisioning Server |
| 145 | GVPMediaControlPlatform | GVP Media Control Platform |
| 146 | GVPFetchingModule | GVP Fetching Module |
| 147 | GVPMediaControlPlatformLegacyInterpreter | GVP Media Control Platform Legacy Interpreter |
| 148 | GVPCallControlPlatform | GVP Call Control Platform |
| 149 | GVPResourceManager | GVP Resource Manager |
| 150 | GVPRedundancyManager | GVP Redundancy Manager |
| 151 | GVPMediaServer | GVP Media Server |
| 152 | GVPPSTNConnector | GVP PSTN Connector |
| 153 | GVPReportingServer | GVP Reporting Server |
| 154 | GVPSSG | GVP SSG |

**Table 22: Application Type Representations (Continued)**

| Type ID | Type Abbreviation | Type Name |
|---|---|---|
| 157 | InteractionWorkspace | Interaction Workspace |
| 158 | Advisors | Advisors |
| 159 | ESSExtensibleServices | ESS Extensible Services |
| 160 | CustomerView | Customer View |
| 161 | OrchestrationServer | Orchestration Server |
| 162 | Reserved | |
| 163 | CapturePoint | Capture Point |
| 164 | RulesESPServer | Rules ESP Server |
| 165 | GenesysAdministrator | Genesys Administrator |
| 166 | iWDManager | iWD Manager |
| 167 | iWDRuntimeNode | iWD Runtime Node |
| 168 | BusinessRulesExecutionServer | Business Rules Execution Server |
| 169 | BusinessRulesApplicationServer | Business Rules Application Server |
| 170 | VPPolicyServer | VP Policy Server |
| 171 | SocialMS | Social Messaging Server |
| 172 | CSTAConnector | CSTA Connector |
| 173 | VPMRCPProxy | VP MRCP Proxy |
| 174 | UCMConnector | UCM Connector |
| 175 | OTICSServer | OT ICS Server |
| 176 | OTICSOMPInfra | OT ICS OMP Infrastructure |
| 177 | Advisors-CCAdviser | Advisors–Contact Center Advisor |
| 178 | Advisors-FAAdviser | Advisors–Frontline Advisor |
| 179 | Advisors-AdvisorsPlatform | Advisors–Advisors Platform |
| 180 | Advisors-AdvisorsGenesysAdapter | Advisors–Advisors Genesys Adapter |
| 181 | Advisors-AdvisorsCiscoAdapter | Advisors–Advisors Cisco Adapter |

**Table 22: Application Type Representations (Continued)**

| Type ID | Type Abbreviation | Type Name |
|---|---|---|
| 182 | FederationServer | Federation Server |
| 183 | FederationStatProvider | Federation Stat Provider |
| 184 | GenesysAdministratorServer | Genesys Administrator Server |
| 185 | WebEngagementBackendServer | Web Engagement Backend Server |
| 186 | WebEngagementFrontendServer | Web Engagement Frontend Server |
| 187 | WebRTCGateway | WebRTC Gateway |

a. Prior to release 8.0, this value was used for the Application Type IS Transport Server. In release 8.0 and later, this value is used for the new application type SMS Server.

# How to Activate SNMP Support

This section describes what changes you must make in your Genesys installation to enable SNMP communications between the Management Layer and your Network Management System.

As already mentioned, the communications between SCS and NMS require an AgentX-compliant SNMP master agent application:

- If your NMS already contains such an application, configure an `Application` object for it in the Genesys Configuration Database. See the section for instructions.
- If you would like to use Genesys SNMP Master Agent, deploy it as described in the *Framework 8.1 Deployment Guide.*

For either configuration, order licenses that enable the SNMP functionality of the Management Layer and modify the licensing system as needed. Refer to the *Genesys Licensing Guide* for more information.

## Stand-alone or Redundant SNMP Master Agents

The Management Layer supports two types of configuration—stand-alone and redundant. Stand-alone configuration consists of a single SNMP Master Agent. Redundant configuration consists of two SNMP master agent applications, one primary and one backup. When Solution Control Server loses a connection with the primary SNMP master agent, SCS switches all NMS communications to the backup SNMP master agent.

Refer to the *Framework 8.1 Deployment Guide* for detailed instructions about deploying both stand-alone and redundant SNMP Master Agents.

## Setting Configuration Options

The *Framework 8.1 Configuration Options Reference Manual* describes configuration options and their values for Genesys SNMP Master Agent. You can alter trap configuration (target host, port, community, and multiple trap destinations) by modifying `SNMP_TARGET_MIB`, which you maintain externally with SNMP commands through Genesys Master Agent. Refer to RFC 2273 (SNMPv3 Applications) for further information about how to configure traps via standard management MIBs.

Solution Control Server reads the configuration settings of the SNMP Master Agent `Application` object and uses option values from the `agentx` configuration section to connect to SNMP Master Agent. This is true for both Genesys SNMP Master Agent and a third-party SNMP master agent. Therefore, if you are using a third-party SNMP master agent, make sure that the option values configured for the SNMP Master Agent `Application` object in the Configuration Database match the actual configuration settings in your third-party SNMP master agent.

# How to Use Graceful Contact-Center Shutdown Script

Starting with 7.0 release, the Management Layer provides authorized users with capability to gracefully shut down contact-center software. This functionality, implemented in the form of a PERL script, operates through the Management Layer SNMP Interface.

The Contact-Center Graceful Shutdown script (called `ccgs.pl`) does the following:

1. Enumerates all currently running T-Servers.
2. Determines if there are ongoing interactions in the contact center by querying the number of active calls from each T-Server.
3. If there are active calls, waits one minute and then checks again.
4. When there are no more active calls, shuts down T-Servers.

**Installing the Script**

If you installed Solution Control Server, `ccgs.pl` is already installed, and is located in the same folder in which Solution Control Server was installed.

Starting in 8.1.2, you can install the Solution Control Server utilities without installing Solution Control Server itself. If you have not installed the utilities, use the procedure “Installing Solution Control Server Utilities” in the *Framework 8.1 Deployment Guide*. After the utilities are installed, `ccgs.pl` is stored in the location you specified during the installation.

The SCS installation package also includes all additional PERL modules necessary to utilize the Management Layer SNMP Interface with PERL scripts.

### Using the Script

Start the script file with the command line in this format:

```
ccgs.pl [<flag> <value>] [<flag> <value>] ...
```

The script accepts the following command-line parameters:

`-h` Master Agent host name or IP address

Default: localhost

`-p` Master Agent SNMP port

Default: 161

`-c` Community string

Default: public

`-v` SNMP version (v1 or v2c)

Default: v2c

`-pt` Polling timeout—the time, in seconds, the script waits for the end of data tables refresh. Should be equal to or greater than 5

Default: 60

`-mic` Maximum number of idle polling cycles the script waits before exit. An *idle polling cycle* is one when no shutdown requests are sent. Should be equal to or greater than 1

Default: 100

`-st` SNMP timeout—the timeout, in seconds, during which the script waits for a response after a request is sent. Note that when a request is retried, the timeout is increased by the SNMP backoff factor (see below). Should be equal to or greater than 1.

Default: 2

`-sr` SNMP retries—number of attempts that the script prompts for a reply to the SNMP request. If no response is received within the timeout specified in value `-st`, the request is resent and a new response awaited with a longer timeout. Should be equal to or greater than 1.

Default: 5

`-sb` SNMP backoff—factor used to increase the timeout every time an SNMP request is retried. Should be equal to or greater than 1.

Default: 1

Separate flags from their values with a word space.

Chapter

# 8 Managing Third-Party Applications

This chapter describes which Management Layer functions you can use with third-party applications and how the Management Layer processes the related commands. It also lists the software prerequisites for and describes how to configure these applications.

This chapter contains the following sections:

## Prerequisites

In Genesys terms, a *third-party application* is an application not instrumented with Genesys libraries. The Management Layer can monitor, start, and stop a third-party application as long as that application:

- Supports a startup from a command line.
- Starts if the computer it runs on is unattended (for instance, on a Windows computer with no user logged in); however, this is not mandatory.
- Works without a console window on Windows; however, this is not mandatory.
- Is registered in the Configuration Database as an Application of the Third Party Server type.
- Runs on an operating system that Genesys supports.

**Note:** You cannot perform the centralized logging and alarm-signaling functions (including switchover) over a third-party application because they require built-in support on the application side.

# Required Components and Configuration

If you have configured third-party applications in the Genesys Configuration Database, Management Layer can control, monitor, start, and stop them. Even if you do not use the Management Layer to start a particular application, the application's runtime status is displayed. This functionality is also supported for:

- Third-party applications installed as Windows Services.
- Third-party applications started with a script.

Managing third-party applications requires the installation of:

- Solution Control Server.
- An instance of Local Control Agent (LCA) for each host computer running third-party applications.
- Any combination of the following:
  - Genesys Administrator.
  - An instance of Solution Control Interface (SCI) for each user who performs the monitoring and control functions.

The monitoring views and control commands are available through Genesys Administrator and SCI, just as they are for managing Genesys applications.

*Framework 8.1 Genesys Administrator Help* provides detailed instructions for viewing the applications, and starting and stopping them. *Framework 8.1 Solution Control Interface Help* provides detailed instructions for displaying each view for an application, and for starting and stopping applications.

## Configuring Third-Party Applications

Use one of the following procedures to create an `Application` object for the third-party application and configure it appropriately.

### Procedure: Genesys Administrator—Configuring an Application object for a third-party application

**Purpose:** To create and configure an Application object for a third-party application using Genesys Administrator.

### Prerequisites

- Management Layer components are installed and running.
- The third-party application is installed.
- You are logged in to Genesys Administrator.

### Start of procedure

1. Register the third-party application in the Configuration Database as an `Application` object of the `Third Party Server` type.
2. In the `Server Info` section of the `Application` object's `Configuration` tab, specify the following:
   - `Working Directory`— The full path to the directory from which the application starts.
   - `Command Line`— The command line used for starting the application; usually, it is the name of the executable file.
3. If you also want to start and stop the third-party applications, do the following:
   a. In the `Server Info` section of the `Application` object's `Configuration` tab, in the `Command Line Arguments` field, specify any additional parameters used to start the application.
   b. If the application is started with a batch file or script, specify the name of the command used for launching that file or script. In the `Annex` list of the `Application` object's `Options` tab, create a section named `start_stop` and an option named `start_command`.
   c. As a value for this option, specify the command that launches the batch file or script, including the full path to the executed file or script. (The `start_command` option may also contain the command to start the executable file.)
   d. If the application is stopped with a batch file or script that performs the correct shutdown of the application, specify the name of the command used for launching that file or script. In the `Annex` list of the `Application` object's `Options` tab, create (or open) a section named `start_stop` and create an option named `stop_command`. For its value, specify the command that launches the batch file or script, including the full path to the executed file or script.

### End of procedure

## Procedure: Configuration Manager—Configuring an Application object for a third-party application

**Purpose:** To create and configure an Application object for a third-party application using Configuration Manager.

### Prerequisites

- Management Layer components are installed and running.
- The third-party application installed.
- You are logged in to Configuration Manager.

### Start of procedure

1. Register the third-party application in the Configuration Database as an `Application` object of the `Third Party Server` type.
2. On the `Start Info` tab of the `Application` object's `Properties` window, specify the following:
    - `Working Directory`— The full path to the directory from which the application starts.
    - `Command Line`—The command line used for starting the application; usually, it is the name of the executable file.
3. If you also want to start and stop the third-party applications, do the following:
    a. On the `Start Info` tab of the `Application` object's `Properties` window, in the `Command Line Arguments` field, specify any additional parameters used to start the application.
    b. If the application is started with a batch file or script, specify the name of the command used for launching that file or script. On the `Annex` tab of the `Application Properties` window, create a section named `start_stop` and an option named `start_command`.
    c. As a value for this option, specify the command that launches the batch file or script, including the full path to the executed file or script. (The `start_command` option may also contain the command to start the executable file.)
    d. If the application is stopped with a batch file or script that performs the correct shutdown of the application, specify the name of the command used for launching that file or script. On the `Annex` tab of the `Application Properties` window, create (or open) a section named `start_stop` and create an option named `stop_command`. For its value,

specify the command that launches the batch file or script, including the full path to the executed file or script.

**End of procedure**

# Monitoring Third-Party Applications

Monitoring functionality is provisioned by the ability of Local Control Agent to determine whether:

- A third-party application is started.
- A third-party application is stopped.

## Determining Whether Applications Are Started

LCA uses the so-called command–line matching mechanism to determine if a third-party application is started. This means that LCA periodically retrieves a list of all currently running processes, and then compares command lines of all processes from that list with possible command lines of the third-party applications being checked.

LCA uses the following command–line matching rules for this comparison:

- The command line of a process is equal to the set of these elements:

  ```
  Working Directory + Command Line [+ Command Line Arguments]
  ```

  where `Working Directory`, `Command Line`, and `Command Line Arguments` are the properties of the `Application` object in the Configuration Database. If the `Command Line Arguments` property is empty, it is not used.

  The processes started with the full path specification are evaluated based on this rule.

- The command line of a process is equal to the set of these elements:

  ```
  Command Line [+ Command Line Arguments]
  ```

  where `Command Line` and `Command Line Arguments` are the properties of the `Application` object in the Configuration Database. If the `Command Line Arguments` property is empty, it is not used.

  The processes started without the full path specification are evaluated based on this rule.

- For Windows operating systems only, the command line of a process is equal to the set of these elements:

  ```
  " + Working Directory + Command Line + " [+ Command Line Arguments]
  ```

  where `Working Directory`, `Command Line`, and `Command Line Arguments` are the properties of the `Application` object in the Configuration Database. If the property `Command Line Arguments` is empty, it is not used.

The processes started with the full path specification are evaluated based on this rule when the path contains spaces.

If LCA finds a process whose command line matches that of a third-party application, LCA assumes that the application has started and then:

1. Stores the PID (process identifier) for that application.
2. Sets the application status to `Started`.
3. Sends a notification to SCS.

## Determining Whether Applications Are Stopped

LCA uses the so-called PID-check mechanism to determine if a third-party application is stopped. This means that LCA tracks the PIDs (process identifiers) for all currently running processes. Using relevant operating system commands, LCA determines if a process with a particular PID is running. If not, LCA considers the corresponding third-party application stopped and:

1. Sets the application status to `Stopped`.
2. Sends a notification to SCS.

# Starting Third-Party Applications

You can start a third-party application in the following ways:

- User command from Genesys Administrator or SCI
- Alarm Reaction
- Automatically at host start-up
- Autorestart

When the Management Layer receives a request to start a particular third-party application, SCS generates a command line and passes it to LCA, which executes the required operating system function.

SCS generates the command line based on the parameters you configured for a specific third-party `Application` object in the Configuration Database, which may include:

- Working Directory—as specified in the `Application` object's properties.
- Command Line—as specified in the `Application` object's properties.
- Command Line Arguments—as specified in the `Application` object's properties.
- Start Command—as specified by the `start_command` option of the `Application` object's `Annex`.

If you have not specified values for the first three listed parameters or have not created a `start_command` option and provided a value for it, the Management

Layer cannot start the application. For more information about these parameters, refer to “Configuring Third-Party Applications” on page 128.

Solution Control Server forms the command line as follows:

- If you have specified the `start_command` option, SCS uses its value to form the command line and ignores the other parameters.
- If you have not specified the `start_command` option, SCS uses the values of the `Command Line` and `Command Line Arguments` to form the command line, while LCA executes an appropriate operating system function in the directory specified as the `Working Directory` for this application.

LCA passes all required parameters to the operating system function (`CreateProcess` on Windows or `execvp` on UNIX) and calls the function, after which two scenarios can occur:

- The operating system function returns an error. In this case, LCA passes the error to SCS, which retains the Stopped status for the third-party application.
- The operating system function does not return an error. In this case, LCA determines the status of the third-party application and passes the status to SCS. (See “Determining Application Status” on page 134 for a description of the methods LCA uses to determine the application status.)

Whichever scenario occurs, the startup process is then considered finished.

## Starting Third-Party Applications Automatically

Management Layer supports the automatic start-up of third-party applications. You must be aware of, and correct if necessary, the configuration of the startup timeout for automatically started third-party applications.

The Management Layer must know the correct status of the third-party application at the exact moment when determining if the application is stopped and is to be started, that is, when the startup timeout for the third-party application expires.

Incorrect configuration of the automatic third-party application start-up configuration can cause multiple instances of the same application to be started.

## Restarting Third-Party Applications Automatically

The Management Layer also supports the automatic restart of third-party applications following an unexpected termination. You must select `Autorestart` when configuring the application. Note that the startup timeout is not used when restarting third-party applications.

## Determining Application Status

The method LCA uses to determine the current status of a third-party application depends on the method SCS uses for forming the startup command line:

- If you have not configured the `start_command` option, SCS uses the values of the `Command Line` and `Command Line Arguments` to form the command line. In this case, LCA stores the PID returned by the operating system function and immediately passes the Started status to SCS.
- If you have configured the `start_command` option, SCS uses the value of this option to form the command line. In this case, LCA passes the Pending status to SCS and determines if the application has started successfully, as described in “Determining Whether Applications Are Started” on page 131.

### Ensuring Command Line Correctness

If you want to monitor a third-party application in SCI, use the running process and its arguments as a model for the command line and command line arguments in Genesys Administrator or Configuration Manager. Follow these steps:

1. Use a system tool (for example, the UNIX tool `ps`) to display the running process and its arguments.
2. In the third-party `Application` object’s properties, do the following:
   - In the `Command Line` field, enter the exact value of the running process.
   - In the `Command Line Arguments` field, enter the exact value of the running process arguments.

# Stopping Third-Party Applications

You can stop a third-party application in the following ways:

- User command from Genesys Administrator or SCI
- Alarm Reaction

**Note:** The Shutdown Timeout does not apply to third-party applications.

The Management Layer can only stop a third-party application that has a status of `Started`; that is, LCA knows the PID for this application.

When the Management Layer receives a request to stop a particular third-party application, SCS passes it to LCA, which executes the required operating system function.

LCA processes the request as follows:

- If you have not configured the `stop_command` option and the application runs on UNIX, LCA sends the `SIGINT` signal to the process with the PID corresponding to the third-party application. Then, LCA sets the application status to `Stopped` and notifies SCS.

  **Note:** For more information about the `stop_command` option, refer to "Configuring Third-Party Applications" on page 128.

- If you have not configured the `stop_command` option and the application runs on Windows, LCA calls the `TerminateProcess` function for the process with the PID corresponding to the third-party application. Then, LCA sets the application status to `Stopped` and notifies SCS.
- If you have configured the `stop_command` option, LCA either executes the specified operating system command or launches the specified script or batch file. LCA sets the status of the third-party application to `Pending` and then determines the actual status of the application, which, when determined, it passes to SCS. (See "Determining Whether Applications Are Stopped" on page 132 for a description of the methods LCA uses to determine the application status.)

At this point, the process of stopping a third-party application is considered finished.

# Example

To demonstrate how you can use the Management Layer to control a third-party application, such as License Manager, running on a Windows-based computer, do the following:

1. Install License Manager to directory `d:\flexlm`. Use the License Manager installation procedure for Windows described in the *Genesys Licensing Guide* document.

2. Create two *.bat files, one (named `lmgrd_run.bat`) for starting License Manager and the other (named `lmgrd_stop.bat`) for shutting down the application. The file content should be as described in "Start Script and Stop Script Content" on page 136.

3. Save both files to the `d:\flexlm` directory.

4. Create an `Application` object of the `Third Party Server` type and name it `FLEXlm`. Refer to the configuration procedures in "Configuring Third-Party Applications" on page 128.

   In the third-party `Application` object's `Startup Info,` set the parameters as follows:

   - Specify `d:\flexlm` as the value for `Working Directory.`

- Specify `lmgrd` as the value for `Command Line`.
- Specify `-c d:\flexlm\license.dat` as the value for `Command Line Arguments`.

**Note:** Make sure that the combined string `<Working directory> + <Command Line> + <space> + <Command Line Arguments>` matches the command line in the `lmgrd_run.bat` file, which is `d:\flexlm\lmgrd -c d:\flexlm\license.dat`.

5. Start `lmgrd_run.bat` manually. After 20 or so seconds, check the `Application` object's status. Its status should be `Started`.
6. In the FLEXlm `Application` object's `Annex`:

   **a)** Create a section called `start_stop`.

   **b)** Create two options, `start_command` and `stop_command`, in this new section. Specify full paths to the appropriate *.bat files as the option values:

   ```
   start_command = d:\flexlm\lmgrd_run.bat
   stop_command = d:\flexlm\lmgrd_stop.bat
   ```

   **c)** Save configuration changes.
7. Try to stop and start the FLEXlm application using appropriate commands in whichever interface you are using—Genesys Administrator or SCI.

## Start Script and Stop Script Content

The content of the `lmgrd_run.bat` and `lmgrd_stop.bat` files depends on whether you run License Manager as a regular application or as a Windows Service.

For a regular application, the file content should be as follows:

**lmgrd_run.bat**

```
@echo "Starting FLEXlm License Manager"
d:\flexlm\lmgrd -c d:\flexlm\license.dat
```

**lmgrd_stop.bat**

```
@echo "Stopping FLEXlm License Manager"
d:\flexlm\lmutil lmdown -q -c d:\flexlm\license.dat
```

For a Windows Service, the file content should be as follows:

**lmgrd_run.bat**

```
net start <FLEXlm Service Name>
d:\flexlm\lmgrd -c d:\flexlm\license.dat
```

**lmgrd_stop.bat**

```
net stop <FLEXlm Service Name>
```

Genesys

Chapter

# 9 Log Format

A *log record* is a data record that stores information communicated in a single log event. In SCI, you can store log records in plain text format or in XML (Extensible Markup Language) format. This chapter describes these formats, and the format of the Centralized Log Database.

This chapter contains the following sections:

## Plain Text Format

Log records stored in the plain text format contain the following fields separated from each other by three spaces:

- `Timestamp`: the date and time when the reported event occurred, in the time format specified for the operating system on which the application runs.
- `Level`: the log level of the reported event. The Standard level of logging contains high-level events that report both major problems and normal operations of in-service solutions. The Interaction level of logging reports the details of an interaction process by solution components that handle interactions and contains information about the processing steps for each interaction by each solution component. The Trace level of logging reports the details of communication between the various solution components and contains information about the processing steps for each interaction by each solution component. The Alarm level of logging reports the events related to alarm detection, processing, and removal.
- `Host Name`: the name of the host, as specified in the Configuration Database, on which an application that reported the event runs.
- `Application Name`: the name of the application, as specified in the Configuration Database, that reported the event.

- `Log Event ID:` the unique identifier of the event in the form of *YY-XXXXX,* where *YY* is the origin (identifier of the `*.lms` file) and *XXXXX* is the event ID uniquely identifying the event within the origin.
- `Text:` the text describing and specifying the event.

### Sample

```
01/30/02 18:40:33   Standard   localhost   StatServer2 00-04502   Cannot
connect to T-Server M1 at host alpha, port 3000
```

## XML Format

A log record in an XML file has the following format:

```
<logmessage>
   <id>Log Event ID</id>
   <timegenerated>Time</timegenerated>
   <priority>Level</priority>
   <hostname>Host Name</hostname>
   <appname>Application Name</appname>
   <text>Text</text>
</logmessage>
```

Log records stored in the XML format contain these fields:

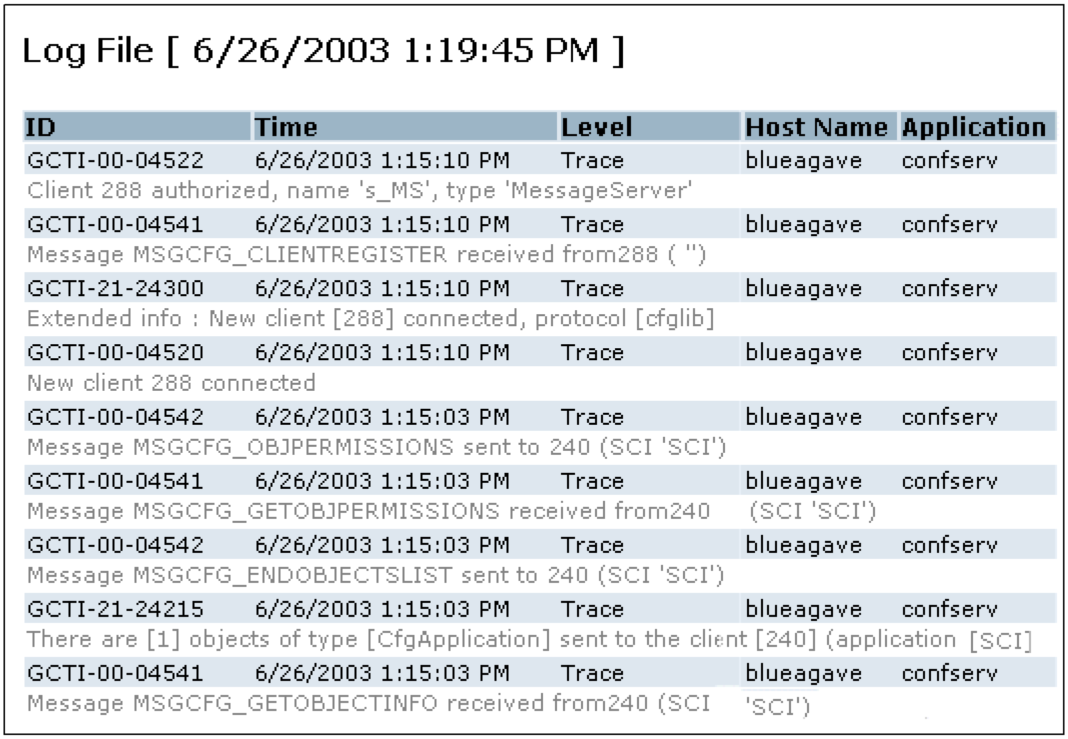
- `Log Event ID:` the unique identifier of the event in the form of *YY-XXXXX,* where *YY* is the origin (identifier of the `*.lms` file) and *XXXXX* is the event ID uniquely identifying the event within the origin.
- `Time:` the date and time when the reported event occurred, in the time format specified for the operating system on which the application runs.
- `Level:` the log level of the reported event. The Standard level of logging contains high-level events that report both major problems and normal operations of in-service solutions. The Interaction level of logging reports the details of an interaction process by solution components that handle interactions and contains information about the processing steps for each interaction by each solution component. The Trace level of logging reports the details of communication between the various solution components and contains information about the processing steps for each interaction by each solution component. The Alarm level of logging reports the events related to alarm detection, processing, and removal.
- `Host Name:` the name of the host, as specified in the Configuration Database, on which an application that reported the event runs.
- `Application Name:` the name of the application, as specified in the Configuration Database, that reported the event.
- `Text:` the text defining and describing the event.

### Sample

```
<logmessage>
   <id>42-23000</id>
   <timegenerated>1/14/02 11:17:41</timegenerated>
   <priority>Trace</priority>
   <hostname>localhost</hostname>
   <appname>Message_Server_1</appname>
   <text>Message ID=252 of type 1101 received from client 276</text>
</logmessage>
```

### Sample Screenshot

Figure 7 shows how log records look when exported in an XML file.



Log File [ 6/26/2003 1:19:45 PM ]

| ID | Time | Level | Host Name | Application |
|---|---|---|---|---|
| GCTI-00-04522 | 6/26/2003 1:15:10 PM | Trace | blueagave | confserv |
| Client 288 authorized, name 's_MS', type 'MessageServer' | | | | |
| GCTI-00-04541 | 6/26/2003 1:15:10 PM | Trace | blueagave | confserv |
| Message MSGCFG_CLIENTREGISTER received from288 ( '') | | | | |
| GCTI-21-24300 | 6/26/2003 1:15:10 PM | Trace | blueagave | confserv |
| Extended info : New client [288] connected, protocol [cfglib] | | | | |
| GCTI-00-04520 | 6/26/2003 1:15:10 PM | Trace | blueagave | confserv |
| New client 288 connected | | | | |
| GCTI-00-04542 | 6/26/2003 1:15:03 PM | Trace | blueagave | confserv |
| Message MSGCFG_OBJPERMISSIONS sent to 240 (SCI 'SCI') | | | | |
| GCTI-00-04541 | 6/26/2003 1:15:03 PM | Trace | blueagave | confserv |
| Message MSGCFG_GETOBJPERMISSIONS received from240 | | | (SCI 'SCI') | |
| GCTI-00-04542 | 6/26/2003 1:15:03 PM | Trace | blueagave | confserv |
| Message MSGCFG_ENDOBJECTSLIST sent to 240 (SCI 'SCI') | | | | |
| GCTI-21-24215 | 6/26/2003 1:15:03 PM | Trace | blueagave | confserv |
| There are [1] objects of type [CfgApplication] sent to the client [240] (application [SCI] | | | | |
| GCTI-00-04541 | 6/26/2003 1:15:03 PM | Trace | blueagave | confserv |
| Message MSGCFG_GETOBJECTINFO received from240 (SCI | | | 'SCI') | |

**Figure 7: Sample Log File in the XML Format**

# Database Format

In release 7.0, the database format was extended. In particular, a table was added to the Centralized Log Database and some data types are changed. The table, `G_LOG_ATTRS`, stores extended log attributes, which applications sometimes attach to log events they generate.

## G_LOG_MESSAGES Table Structure

The structure of the G_LOG_MESSAGES table is described in Table 23:

### Table 23: Structure of G_LOG_MESSAGES Table

| Field Name | Type | Description |
|---|---|---|
| ID | numeric | The unique identifier of the record stored in this table. |
| MESSAGE_ID | integer | The unique identifier of the event. |
| TIMEGENERATED | datetime | The time when the record was written to the database, in GMT format. |
| TIMEWRITTEN | datetime | The time when the record was written to the database, in GMT format. |
| PRIORITY | integer | The log level of the reported event. The Standard level of logging contains high-level events that report both major problems and normal operations of in-service solutions. The Interaction level of logging reports the details of an interaction process by solution components that handle interactions and contains information about the processing steps for each interaction by each solution component. The Trace level of logging reports the details of communication between the various solution components and contains information about the processing steps for each interaction by each solution component. The Alarm level of logging reports the events related to alarm detection, processing, and removal. The PRIORITY field can have one of the following values:<br>2—for Trace-level events<br>3—for Interaction-level events<br>4—for Standard-level events<br>5—for Alarm-level events |
| ORIGIN | integer | Reserved for future use. |
| CATEGORY | integer | Identifies the type of the log record and can have one of the following values:<br>0—for application-related log events<br>2—for audit-related log events |
| DATALEN | integer | Reserved for future use. |
| APPDBID | integer | Reserved for future use. |
| APPTYPE: | integer | The type of application that reported the event. Refer to Table 24 on page 141 for a list of the valid values. |

### Table 23: Structure of G_LOG_MESSAGES Table (Continued)

| Field Name | Type | Description |
|---|---|---|
| APPNAME | string | The name of the application, as specified in the Configuration Database, that reported the event. |
| HOSTNAME | string | The name of the host, as specified in the Configuration Database, on which an application that reported the event runs. |
| MESSAGETEXT: | string | The text defining and describing the event. |

### Table 24: Application Types for APPTYPE Field in the G_LOG_MESSAGES Table

| Value | Application Type |
|---|---|
| 00 | Applications of all types when reporting log events common to all Genesys applications |
| 01 | T-Server |
| 01 | Programmable Gateway Framework |
| 02 | Stat Server |
| 03 | Billing Server |
| 06 | Voice Treatment Server |
| 08 | DB Server |
| 09 | Call Concentrator |
| 10 | CPD (Call Progress Detection) Server |
| 11 | List Manager |
| 12 | Outbound Contact Server |
| 15 | Universal Routing Server |
| 21 | Configuration Server |
| 23 | Third Party Server |
| 26 | DART Server (Obsolete) |
| 28 | Custom Server |
| 29 | External Router |

### Table 24: Application Types for APPTYPE Field in the G_LOG_MESSAGES Table (Continued)

| Value | Application Type |
|---|---|
| 31 | Virtual Routing Point |
| 32 | Database (Obsolete) |
| 33 | Web Option |
| 34 | Detail Biller |
| 35 | Summary Biller |
| 36 | Network Overflow Manager |
| 37 | Backup Control Client |
| 38 | CC Analyzer Data Sourcer |
| 40 | IVR Interface Server |
| 41 | I-Server |
| 42 | Message Server |
| 43 | Solution Control Server |
| 46 | DB Server |
| 48 | WFM Data Aggregator |
| 50 | WFM Schedule Server |
| 52 | ETL Proxy |
| 54 | GVP-Voice Communication Server |
| 55 | VSS System |
| 58 | CC Analyzer Data Mart |
| 59 | Chat Server |
| 60 | Callback Server |
| 61 | Co-Browsing Server |
| 62[a] | SMS Server |
| 63 | Contact Server |
| 64 | E-Mail Server |

**Table 24: Application Types for APPTYPE Field in the G_LOG_MESSAGES Table (Continued)**

| Value | Application Type |
|---|---|
| 65 | Media Link |
| 66 | Web Interaction Requests Server |
| 67 | Web Stat Server |
| 68 | Web Interaction Server |
| 69 | Web Option Route Point |
| 74 | Voice over IP Controller |
| 77 | HA Proxy |
| 78 | Voice over IP Stream Manager |
| 79 | Voice over IP DMX Server |
| 80 | Web API Server |
| 81 | Load Balancer |
| 82 | Application Cluster |
| 83 | Load Distribution Server |
| 84 | G-Proxy |
| 85 | Genesys Interface Server |
| 86 | GCN Delivery Server |
| 88 | IVR DirectTalk Server |
| 89 | GCN Thin Server |
| 90 | Classification Server |
| 91 | Training Server |
| 92 | Universal Callback Server |
| 93 | CPD Server Proxy |
| 94 | XLink Controller |
| 95 | K-Worker Portal |
| 96 | WFM Server |

### Table 24: Application Types for APPTYPE Field in the G_LOG_MESSAGES Table (Continued)

| Value | Application Type |
|---|---|
| 97 | WFM Builder |
| 98 | WFM Reports |
| 99 | WFM Web |
| 100 | Knowledge Manager |
| 101 | IVR Driver |
| 102 | IVR Library |
| 103 | LCS Adapter |
| 104 | Desktop NET Server |
| 105 | Siebel7 ConfSynchComponent |
| 106 | Siebel7 CampSynchComponent |
| 107 | Generic Server |
| 108 | Generic Client |
| 109 | Call Director |
| 110 | SIP Communication Server |
| 111 | Interaction Server |
| 112 | Integration Server |
| 113 | WFM Daemon |
| 114 | GVP Policy Manager |
| 115 | GVP Cisco Queue Adapter |
| 116 | GVP Text To Speech Server |
| 117 | GVP ASR Log Manager |
| 118 | GVP Bandwidth Manager |
| 119 | GVP Events Collector |
| 120 | GVP Cache Server |
| 121 | GVP ASR Log Server |

### Table 24: Application Types for APPTYPE Field in the G_LOG_MESSAGES Table (Continued)

| Value | Application Type |
|---|---|
| 122 | GVP ASR Package Loader |
| 123 | GVP IP Communication Server |
| 124 | GVP Resource Manager |
| 125 | GVP SIP Session Manager |
| 126 | GVP Media Gateway |
| 127 | GVP Soft Switch |
| 128 | GVP Core Service |
| 129 | GVP Voice Communication Server |
| 130 | GVP Unified Login Server |
| 131 | GVP Call Status Monitor |
| 132 | GVP Reporter |
| 133 | GVP H323 Session Manager |
| 134 | GVP ASR Log Manager Agent |
| 135 | GVP Genesys Queue Adapter |
| 136 | GVP IServer |
| 137 | GVP SCP Gateway |
| 138 | GVP SRP Server |
| 139 | GVP MRCP TTS Server |
| 140 | GVP CCS Server |
| 141 | GVP MRCP ASR Server |
| 142 | GVP Network Monitor |
| 143 | GVP OBN Manager |
| 144 | GVP Self Service Provisioning Server |
| 145 | GVP Media Control Platform |
| 146 | GVP Fetching Module |

**Table 24: Application Types for APPTYPE Field in the G_LOG_MESSAGES Table (Continued)**

| Value | Application Type |
|---|---|
| 147 | GVP Media Control Platform Legacy Interpreter |
| 148 | GVP Call Control Platform |
| 149 | GVP Resource Manager |
| 150 | GVP Redundancy Manager |
| 151 | GVP Media Server |
| 152 | GVP PSTN Connector |
| 153 | GVP Reporting Server |
| 154 | GVP SSG (formerly GVP ASG) |
| 155 | GVP CTI Connector |
| 156 | Resource Access Point |
| 157 | Interaction Workspace |
| 158 | Advisors |
| 159 | ESS Extensible Services |
| 160 | Customer View |
| 161 | Orchestration Server |
| 162 | Reserved |
| 163 | Capture Point |
| 164 | Rules ESP Server |
| 165 | Genesys Administrator |
| 166 | iWD Manager |
| 167 | iWD Runtime Node |
| 168 | Business Rules Execution Server |
| 169 | Business Rules Application Server |
| 170 | VP Policy Server |
| 171 | Social Messaging Server |

**Table 24: Application Types for APPTYPE Field in the G_LOG_MESSAGES Table (Continued)**

| Value | Application Type |
|---|---|
| 172 | CSTA Connector |
| 173 | VP MRCP Proxy |
| 174 | UCM Connector |
| 175 | OT ICS Server |
| 176 | OT ICS OMP Infrastructure |
| 177 | Advisors–Contact Center Advisor |
| 178 | Advisors–Frontline Advisor |
| 179 | Advisors–Advisors Platform |
| 180 | Advisors–Advisors Genesys Adapter |
| 181 | Advisors–Advisors Cisco Adapter |
| 182 | Federation Server |
| 183 | Federation Stat Provider |
| 184 | Genesys Administrator Server |
| 185 | Web Engagement Backend Server |
| 186 | Web Engagement Frontend Server |
| 187 | WebRTC Gateway |

[a] Prior to release 8.0, this value was used for the Application Type IS Transport Server. In release 8.0 and later, this value is used for the new application type SMS Server.

## G_LOG_ATTRS Table Structure

The structure of the `G_LOG_ATTRS` table is described in Table 25:

**Table 25: Structure of G_LOG_ATTRS Table**

| Field Name | Type | Description |
|---|---|---|
| ID | numeric | The unique identifier of the record stored in this table. |

### Table 25: Structure of G_LOG_ATTRS Table (Continued)

| Field Name | Type | Description |
|---|---|---|
| LRID | numeric | The unique identifier of the log record stored in the `G_LOG_MESSAGES` table to which this extended attribute belongs |
| MESSAGE_ID | integer | The unique identifier of the event |
| ATTR_NAME | string | The name of the extended attribute. |
| ATTR_VALUE | string | The value of the extended attribute in string format. |

Chapter

# 10 Predefined Alarm Conditions

This chapter describes alarm conditions that are preconfigured by Genesys and become available immediately after you set up Framework 8.1. The conditions under which alarms are generated, the actions automatically taken by the system to cope with or recover from the failure, and the maintenance actions appropriate in each situation are discussed for each alarm condition:

## Connection Failure

This section describes the `Connection Failure` predefined alarm condition.

### Configuration

Table 26 on page 150 describes the default configuration of the `Connection Failure` predefined alarm condition.

**Table 26: Connection Failure Predefined Alarm Condition**

| Field Name | Value and Description |
|---|---|
| Name | Connection Failure |
| Description | The connection between any two Genesys components has been lost. |
| Category | Major |
| Detect Event | 00-04504: Connection to [server type] [server name] at host [host name] port [port number] lost |
| Selection Mode | Select By Any |
| Cancel Event | 00-04503: Connected to [server type] [server name] at host [host name] port [port number] |
| Cancel Timeout | 48 hours |
| Reaction Scripts | None |
| Clearance Scripts | None |
| State Enabled | True |

## Detailed Description

Reports that the specified connection between any two applications has been lost. Always reported by the client application and might indicate one of the following:

- The connection was intentionally closed by the server (for example, in response to an overload situation).
- The connection was closed by a networking software (for example, in response to a long interval without any data exchange through the given connection).
- The server terminated.
- The server stopped responding.
- The server host failed.
- A network connectivity problem occurred between the computers that run the given client application and the server.

### Automatic Recovery Actions

- If a backup server for the specified server is not configured, the client application that reported the connection failure periodically attempts to reconnect to the specified server.
- If a backup server for the specified server is configured, the client application that reported the connection failure attempts to connect interchangeably to the specified server and the backup server.

**Note:** The number of reconnect attempts is unlimited.

- After a successful reconnect attempt, the alarm condition is automatically cleared.

### Suggested Maintenance Actions

1. Check the condition of the server host computer.
2. Check the condition of the server.
3. Check the server log to see if the given application has disconnected intentionally. Look for log events with ID 4523.
4. Check the network connectivity between the computers that run the given application and the server.

## Application Failure

This section describes the `Application Failure` predefined alarm condition.

### Configuration

Table 27 describes the default configuration of the `Application Failure` predefined alarm condition

**Table 27: Application Failure Predefined Alarm Condition**

| Field Name | Value and Description |
|---|---|
| Name | Application Failure |
| Description | Failure of any daemon Genesys component monitored by the Management Layer |
| Category | Major |
| Detect Event | 00-05064: Application terminated due to internal condition |

**Table 27: Application Failure Predefined Alarm Condition (Continued)**

| Field Name | Value and Description |
|---|---|
| Selection Mode | Select by any |
| Cancel Event | 00-05090: Application start detected by Management Layer |
| Cancel Timeout | 48 hours |
| Reaction Scripts | None |
| Clearance Scripts | None |
| State Enabled | True |

## Detailed Description

Reports that the specified application has either terminated or stopped responding. It might indicate one of the following:

- The application terminated because of an internal condition.
- The application was closed by means other than the Management Layer (for example, with an operating system command).
- The application entered a no-response condition.

## Automatic Recovery Actions

- If a backup application for the specified application is not configured and the autorestart function is enabled, the Management Layer attempts to restart the specified application.
- If a backup application for the specified application is not configured and the autorestart function is disabled, no automatic recovery action takes place.
- If a backup application for the specified application is configured and the autorestart function is enabled, the Management Layer switches operations over to the backup application and attempts to restart the specified application in the Standby mode.
- Upon a successful attempt to restart the specified application, the alarm is automatically cleared.

## Suggested Maintenance Actions

1. Using Solution Control Interface (SCI), locate the exact source of the alarm and check the current status of the application. It is likely that the fault has been eliminated through an automatic recovery action.

2. If the alarm is still active, check the status of the application through the operating system tools.
3. If the application is running but not responding, restart the application with an operating system command.
4. If the application is not running, start the application with an operating system command.
5. Ensure that SCI shows the status of the application correctly.
6. Verify that the `Auto-Restart` check box for this application is selected.

# Licensing Error

This section describes the `Licensing Error` predefined alarm condition.

## Configuration

Table 28 describes the default configuration of the `Licensing Failure` predefined alarm condition.

**Table 28: Licensing Failure Predefined Alarm Condition**

| Field Name | Value and Description |
|---|---|
| Name | Licensing Error |
| Description | Any licensing error identified by any Genesys component |
| Category | Critical |
| Detect Event | 00-07100: Licensing violation is identified, the violation type [type] |
| Selection Mode | Select by any |
| Cancel Event | None |
| Cancel Timeout | 48 hours |
| Reaction Scripts | None |
| Clearance Scripts | None |
| State Enabled | True |

## Detailed Description

Reports that a licensing error occurred. Possible violation types are as follows:

1. License information is invalid.

2. Licensable feature has expired.
3. Feature usage level has been exceeded.
4. Licensing system has experienced a general failure.

### Automatic Recovery Actions

None

### Suggested Maintenance Actions

Depending on the value of the error code:

- Check the condition of License Manager. If the type of license you have requires License Manager, it should be running and accessible by the Genesys applications. Check that the host and port of License Manager are specified correctly.
- Make sure that the actual location of the license file or license server corresponds to the location specified in the command-line parameter used for application startup.
- Make sure that the specified license file is the exact copy of the license file received from Genesys.
- Locate the exact source of the alarm and apply to Genesys for an extension of the license.
- Locate the exact source of the alarm and check the current usage level against the usage level stipulated in the license. Either decrease the usage level or apply to Genesys for a new license that covers the increased usage needs.

## CTI Link Failure

This section describes the `CTI Link Failure` predefined alarm condition.

### Configuration

Table 29 describes the default configuration of the `CTI Link Failure` predefined alarm condition.

**Table 29: CTI Link Failure Predefined Alarm Condition**

| Field Name | Value and Description |
|---|---|
| Name | CTI Link Failure |
| Description | Failure of connection between any T-Server and its switch |

**Table 29: CTI Link Failure Predefined Alarm Condition (Continued)**

| Field Name | Value and Description |
|---|---|
| Category | Major |
| Detect Event | 01-20002: CTI Link disconnected |
| Selection Mode | Select by application type T-Server |
| Cancel Event | 01-20001: CTI Link connected |
| Cancel Timeout | 48 hours |
| Reaction Scripts | None |
| Clearance Scripts | None |
| State Enabled | True |

## Detailed Description

Reports that the connection between the specified T-Server and its switch has been lost. Always reported by T-Server and might indicate one of the following:

- The connection was intentionally closed on the switch side (for example, as an automatic defense action).
- The control system of the switch failed.
- A network connectivity problem occurred between the T-Server host and the switch.

## Automatic Recovery Actions

T-Server attempts to reconnect to the CTI link.

## Suggested Maintenance Actions

- Check the condition of the control system of the switch and of its CTI link.
- Check the network connectivity between the control system of the switch and the computer running T-Server.
- If you are using redundant T-Servers to increase solution availability, and if your T-Server is of a pre-7.0 release, consider creating an alarm reaction Script object of the `Switchover` type for this alarm condition. If you associate such a script with the CTI Link Failure Alarm Condition object, a switchover to the backup T-Server is performed when the primary T-Server reports a CTI link failure. The efficiency of this measure depends on the availability of redundant CTI links in your switch.

- If you are using redundant T-Servers of release 7.0 or later and Management Layer 7.1.1 and later, you do not need to configure Switchover alarm reactions. Starting with Management Framework 7.1.1, switchover to backup T-Server in case of CTI Link failure is performed automatically because any T-Server of release 7.0 or later changes its status to Service Unavailable in this scenario.

# Host Inaccessible

This section describes the Host Inaccessible predefined alarm condition.

## Configuration

Table 30 describes the default configuration of the Host Inaccessible predefined alarm condition.

**Table 30: Host Inaccessible Predefined Alarm Condition**

| Field Name | Value and Description |
|---|---|
| Name | Host Inaccessible |
| Description | The Management Layer cannot access a host computer on which Genesys daemon applications run. |
| Category | Major |
| Detect Event | 00-08000: Host [host name] inaccessible. LCA is not listening on port [port number]. |
| Selection Mode | Select by any |
| Cancel Event | 00-08001: Host [host name] operates in normal condition. |
| Cancel Timeout | 48 hours |
| Reaction Scripts | None |
| Clearance Scripts | None |
| State Enabled | True |

## Detailed Description

Reports that the Management Layer cannot contact the Local Control Agent on the host on which Genesys daemon applications are running. Might indicate one of the following (in the order of probability of occurrence in a typical production environment):

- The connection between SCS and the LCA of the specified host failed.
- LCA is not started on the specified host.
- LCA is listening on a port that is different from the one specified in the configuration.
- The LCA of the specified host has terminated or stopped responding.

### Automatic Recovery Actions

By default, this failure is treated by the Management Layer as a failure of every Genesys application running on the given host. For the applications located on the given host that have redundancy, the Management Layer makes their backup applications primary. After that, Solution Control Server makes repeated attempts to restore connection with the LCA of the specified host. Once the connection is restored, the Management Layer attempts to start all applications that were running before the alarm occurred.

### Suggested Maintenance Actions

1. Check the condition of LCA. If LCA terminated or stopped responding, restart LCA. Notify Genesys Technical Support about the LCA failure.
2. Verify the LCA command line parameters and make sure that LCA listens on the same port as the one specified in the Configuration Database.

## Service Unavailable

This section describes the `Service Unavailable` predefined alarm condition.

### Configuration

Table 31 on page 157 describes the default configuration of the `Service Unavailable` predefined alarm condition.

**Table 31: Service Unavailable Predefined Alarm Condition**

| Field Name | Value and Description |
|---|---|
| Name | Service Unavailable |
| Description | A Genesys component is unable to provide service for some internal reasons. |
| Category | Major |

**Table 31: Service Unavailable Predefined Alarm Condition (Continued)**

| Field Name | Value and Description |
|---|---|
| Detect Event | 00-05094: Application is not able to provide service. |
| Selection Mode | Select by any |
| Cancel Event | 00-05093: Application is ready to provide service. |
| Cancel Timeout | 48 hours |
| Reaction Scripts | None |
| Clearance Scripts | None |
| State Enabled | True |

### Detailed Description

Reports that a Genesys component cannot provide service for some internal reasons.

### Automatic Recovery Actions

If a backup application for the specified application is configured, the Management Layer switches operations over to the backup application.

### Suggested Maintenance Actions

This alarm occurs because of internal application reasons. Examine the log of the application that signaled the alarm to determine and eliminate the source of problem.

# Host Unavailable

This section describes the Host Unavailable predefined alarm condition.

### Configuration

Table 32 describes the default configuration of the Host Unavailable predefined alarm condition.

**Table 32: Host Unavailable Predefined Alarm Condition**

| Field Name | Value and Description |
|---|---|
| Name | Host Unavailable |
| Description | A host on which Genesys daemon applications are running is unavailable (turned off). |
| Category | Major |
| Detect Event | 00-08002: Host [host name] unavailable. |
| Selection Mode | Select by any |
| Cancel Event | 00-08001: Host [host name] operates in normal condition. |
| Cancel Timeout | 48 hours |
| Reaction Scripts | None |
| Clearance Scripts | None |
| State Enabled | True |

## Detailed Description

Reports that a host on which Genesys daemon applications are running is unavailable (turned off). Might indicate one of the following (in the order of probability of occurrence in a typical production environment):

- Specified host has failed or has been turned off.
- Network problems prevents SCS from connecting to LCA at the specified host.

## Automatic Recovery Actions

This failure may occur when an SCS attempt to connect to the LCA at the specified host fails. This failure is determined based on the error code returned by the networking subsystem. No automatic recovery actions are performed when this failure occurs.

## Suggested Maintenance Actions

1. Check the condition of the host. If the host failed, take measures to restore its normal operating condition. Once the normal condition is restored, the Management Layer automatically brings up all Genesys applications that are supposed to be running.

2. Check the condition of the network. Make sure that it is possible to reach the host of interest from the host on which SCS is running.

# Host Unreachable

This section describes the `Host Unreachable` predefined alarm condition.

## Configuration

Table 33 describes the default configuration of the `Host Unreachable` predefined alarm condition.

**Table 33: Host Unreachable Predefined Alarm Condition**

| Field Name | Value and Description |
|---|---|
| Name | Host Unreachable |
| Description | The Management Layer cannot reach the host on which Genesys daemon applications are running (no route to the host). |
| Category | Major |
| Detect Event | 00-08003: Host [host name] unreachable. |
| Selection Mode | Select by any |
| Cancel Event | 00-08001: Host [host name] operates in normal condition. |
| Cancel Timeout | 48 hours |
| Reaction Scripts | None |
| Clearance Scripts | None |
| State Enabled | True |

## Detailed Description

Reports that the Management Layer cannot reach the host on which Genesys daemon applications are running (no route to the host). Might indicate the following:

- Network configuration is incorrect: there is no route to the host of interest from the host on which SCS is running.

### Automatic Recovery Actions

This failure may occur when an SCS attempt to connect to the LCA at the specified host fails. This failure is determined based on the error code returned by the networking subsystem. No automatic recovery actions are performed when this failure occurs.

### Suggested Maintenance Actions

1. Check the condition of the network. Make sure that routing is configured correctly in the network and that it is possible to reach the host of interest from the host on which SCS is running.

## Unplanned Solution Status Change

This section describes the `Unplanned Solution Status Change` predefined alarm condition.

### Configuration

Table 34 describes the default configuration of the `Unplanned Solution Status Change` predefined alarm condition.

**Table 34: Unplanned Solution Status Change Alarm Condition**

| Field Name | Value and Description |
|---|---|
| Name | Unplanned Solution Status Change |
| Description | Solution status has changed from Started to Pending without any requests to stop the solution. This may indicate a failure of one of the solution components. |
| Category | Major |
| Detect Event | 43-10385: Solution [solution name] nonplanned change of state from Started to Pending. |
| Selection Mode | Select by any |
| Cancel Event | 43-10370: Solution [solution name] is started. |
| Cancel Timeout | 48 hours |
| Reaction Scripts | None |

**Table 34: Unplanned Solution Status Change Alarm Condition (Continued)**

| Field Name | Value and Description |
|---|---|
| Clearance Scripts | None |
| State Enabled | True |

### Detailed Description

Reports that a solution status has changed from Started to Pending without any requests to stop the solution. Might indicate the following:

- Failure of one or more of the solution components.

### Automatic Recovery Actions

If this alarm occurred because of the failure of one or more of the solution components, the Management Layer performs the same automatic recovery actions for each failed application as described for the `Application Failure` alarm condition.

### Suggested Maintenance Actions

For each failed solution component, perform the same Maintenance Actions as suggested for the `Application Failure` alarm condition.

## Message Server Loss of Database Connection

This section describes the `Message Server Loss of Database Connection` predefined alarm condition.

### Configuration

Table 35 describes the default configuration of the `Message Server Loss of Database Connection` predefined alarm condition.

**Table 35: Message Server Loss of DB Connection Alarm Condition**

| Field Name | Value and Description |
|---|---|
| Name | Message Server Loss of Database Connection |
| Description | Message Server has lost connection to the Centralized Log Database. |

**Table 35: Message Server Loss of DB Connection Alarm Condition (Continued)**

| Field Name | Value and Description |
|---|---|
| Category | Major |
| Detect Event | 00-11051: Connection with DB Cluster lost. |
| Selection Mode | Select by application type Message Server |
| Cancel Event | 00-11050: Connection with DB Cluster established ([host name]:[port number]). |
| Cancel Timeout | 48 hours |
| Reaction Scripts | None |
| Clearance Scripts | None |
| State Enabled | True |

## Detailed Description

Reports that Message Server has lost connection to the Centralized Log Database. Might indicate one of the following (in the order of probability of occurrence in a typical production environment):

- Failure of the DB Server used by Message Server to access the Centralized Log Database.
- Failure of the DBMS that stores the Centralized Log Database.

## Automatic Recovery Actions

If this alarm occurred because of the failure of DB Server used by Message Server to access the Centralized Log Database, the Management Layer performs the same automatic recovery actions for DB Server as described for the `Application Failure` alarm condition.

## Suggested Maintenance Actions

1. In the case of Log DB Server failure, perform the same Maintenance Actions as suggested for the `Application Failure` alarm condition.
2. Otherwise, make sure that the DBMS that stores the Centralized Log Database is operating in normal condition.

Chapter

# 11 Troubleshooting

This chapter contains suggestions on how to identify and handle the most common mistakes made when you are enabling the Management Layer functionality.

This chapter contains the following sections:

## Major Checkpoints

The Management Layer must be configured correctly to function properly. Wizards provide this correct configuration. When you are configuring the Management Layer manually, you can ensure that it operates properly by using the following checklist:

- SQL server is running and configured properly.
- Log DB Server is running.
- Configuration Server is running.
- Solution Control Server (SCS) is running.
- Local Control Agent (LCA) is running with sufficient permissions on each monitored host.
- At least one instance of Message Server is running.
- Message Server, used for centralized logging, has the `db_storage` configuration option set to `true`. (Check the `messages` section on the `Options` tab of the Message Server `Properties` window.)

- The Log Database scripts have been executed successfully.
- The user account that is specified in the `DB Info` information of the Database Access Point `Application` object, and is used for accessing the Log Database, has `Write` permissions configured in the Database Management System (DBMS).
- All monitored applications have the `verbose` configuration option set to a value other than `none`. (Check the `Log` section in the `Application` object's `Options`.)
- All monitored applications have the `network` output type specified. (Check the `Log` section in the `Application` object's `Options`.)
- A connection to Message Server is configured in the `Application` object's `Connections`.
- A connection to Message Server is configured in the SCS `Application` object's `Connections`.
- For Genesys Administrator only:
  - A connection to the correct SCS is configured in the Configuration Manager `Application` object's `Connections`.
  - The same Database Access Point `Application` object is specified in the `Connections` of both Configuration Manager and Message Server `Application` objects.
- For SCI only:
  - A connection to the correct SCS is configured in the SCI `Application` object's `Connections`.
  - The same Database Access Point `Application` object is specified in the `Connections` of both the SCI and Message Server `Application` objects.

**Note:** Refer to the *Framework 8.1 Configuration Options Reference Manual* for configuration option descriptions and information about their valid values.

# Alarming

This section suggests actions to take if you have difficulty enabling Management Layer's alarm-signaling functionality.

## No Active Alarms in Genesys Administrator or SCI

- Check the configuration of the Alarm Condition `Application` object and make sure that the correct `Detect Log Event ID` is specified.
- Make sure that the log message appears in a local log file.

- Make sure that all monitored applications have the network output type. (Check the log section in the Application object's Options.)
- Make sure that the verbose configuration option is set to send log messages of the needed level.
- Check the Message Server log to make sure that Message Server receives log messages.
- Make sure that the correct Message Server is configured in the SCS Application object's Connections.
- Check the Message Server log to make sure that Message Server sends messages to SCS.
- For Genesys Administrator only:
  - Make sure that the correct SCS is configured in the Configuration Manager Application object's Connections.
- For SCI only:
  - Make sure that the correct SCS is configured in the SCI Application object's Connections.
  - Check the SCI log to make sure that SCI receives alarms. To enable SCI log, start SCI from the command prompt using the --debug command-line parameter:

    ```
    sci.exe --debug
    ```

## No Alarm Reactions Executed

- Make sure that the alarm is triggered (see "No Active Alarms in Genesys Administrator or SCI" on page 166).
- Make sure that a Script object of the Alarm Reaction type is specified on the Reaction Scripts tab of the Alarm Condition's Properties.

## No Alarm Reactions "Send SNMP Trap" Executed

- Make sure that the alarm is triggered (see "No Active Alarms in Genesys Administrator or SCI" on page 166).
- Make sure that a Script object of the Alarm Reaction type is specified in the Alarm Condition object's Reaction Scripts.
- Make sure that the Genesys or a third-party SNMP Master Agent is installed and configured as required. (See Chapter 7 on page 93.)

## No Alarm Reactions "Send E-Mail" Executed

- Make sure that the alarm is triggered (see "No Active Alarms in Genesys Administrator or SCI" on page 166).
- Make sure that a Script object of the Alarm Reaction type is specified in the Alarm Condition object's Reaction Scripts.

- Make sure that the e-mail system is installed and configured as required. (See Chapter 6 on page 89.)
- Make sure that the correct e-mail address is specified for the Alarm Reaction Script object.

# Logging

This section suggests actions to take if you have difficulty enabling Management Layer's logging functionality.

## No Application Logs

- Check the `Log` section in the `Application` object's `Options` and make sure that the `verbose` configuration option is set to a value other than `none`.
- Check the `Log` section in the `Application` object's `Options` and make sure that at least one output type is specified.

## No Log Messages in Genesys Administrator or SCI

- Check the `Log` section in the `Application` object's `Options` and make sure that the `verbose` configuration option is set to a value other than `none`. Then check the value of the configuration option that corresponds to the value of `verbose` and make sure that it is set to `network`.
- Make sure that Message Server is configured in the `Application` object's `Connections`.
- Check the `messages` section in the Message Server `Application` object's `Options` and make sure that the `db_storage` configuration option is set to `true`.
- Make sure that a user with `Write` permission is configured in the DBMS and that the same user account is specified in the `DB Info` of the Database Access Point.
- For Genesys Administrator:
  - Make sure that the same Database Access Point `Application` object is specified on the `Connections` tab of both the Configuration Manager and Message Server `Properties` windows.
- For Configuration Manager:
  - Make sure that the same Database Access Point `Application` object is specified on the `Connections` tab of both the SCI and Message Server `Properties` windows.

# Application Start/Stop

This section suggests actions to take if you have difficulty enabling Management Layer's control functionality.

## Applications Cannot Be Started

- Make sure that LCA is running on the host on which the application is installed.
- Check the SCS log to make sure that connection to the LCA running on the application's host is established.
- Make sure that the command-line parameters are specified in the `Application` object's properties.
- Make sure LCA has sufficient permissions to start an application.
- Make sure that the application is installed on the host specified in the `Application` object's `Start Info` section (in Genesys Administrator) or `Server Info` tab (Configuration Manager).

## Applications Cannot Be Stopped

- Make sure LCA has sufficient permissions to stop an application.

## Connectivity Failure and Microsoft's Media-Sense Feature

In the Microsoft Windows XP operating system, when a host is disconnected from the network, LCA may start second instances of Solution Control Server and Configuration Server even though these components are already running. That in itself is harmless because Framework detects and terminates the additional CS instance and the additional SCS instance never connects to Framework.

The probable cause of this effect is Microsoft's `media-sense` feature, included with Windows XP system, which enables a NIC (Network Interface Card) to detect if a network cable is connected to it. The default setting of this feature (`on`) has these effects:

- If any cable is disconnected, Windows disables the protocols on the adapter, which affects TCP/IP (although loopback of 127.0.0.1 in your HOSTS file still works). This affects applications that require IP connectivity to remain constant; for example, laptops.
- If a network cable is disconnected, Windows also disables the entire network protocol stack, which means that you cannot reach network addresses on your own system.

If you find these effects undesirable, you should disable `media-sense.` Use the following procedure to do this for systems that use TCP/IP.

### Procedure: Disabling Microsoft's media-sense feature

**Purpose:** To disable Microsoft's `media-sense` feature on systems that use TCP/IP so that LCA doesn't start second instances of SCS and Configuration Server.

**Start of procedure**

1. Start the registry editor (`regedit`).
2. Move to `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters`
3. From the Edit menu select `New - DWORD value`.
4. Name the new item `DisableDHCPMediaSense` and press `Enter`.
5. Double click the new value and set it to `1`.
6. Click `OK`
7. Reboot the computer.

**End of procedure**

Point your browser to `http://www.microsoft.com`, and search for `media-sense` to read more information about disabling this feature.

# Alarm Reaction

This section suggests actions to take if you have difficulty enabling alarm reactions of the `Send an e-mail` and `Send an SNMP Trap` types.

## E-Mail

- Make sure the host on which SCS is running has an e-mail system, which is installed, configured, and running correctly.

See also Chapter 6 on page 89.

## SNMP Traps

- Make sure that the SNMP Master Agent `Application` object is configured in the SCS `Application` object's `Connections`.
- Make sure that the host and port parameters are specified correctly on the SNMP Master Agent `Application` object's `Server Info` section (in Genesys Administrator) or `Start Info` tab (in Configuration Manager).

- Make sure that the configuration options are set correctly in the SNMP Master Agent `Application` object's `Options`.

See also Chapter 7 on page 93.

# Distributed SCS Functionality

This section suggests actions to take if you have difficulty enabling `Distributed` mode for Solution Control Servers that control your environment.

Refer to the *Framework 8.1 Deployment Guide* for detailed instructions and information about configuring distributed Solution Control Servers.

## Incorrect Message Server Configuration

- Make sure you have a Message Server dedicated to support communications among Distributed Solution Control Servers. Verify that the `signature` configuration option is set to the `scs_distributed` value in the `MessageServer` section in that Message Server `Application` object's `Options`.

## Incorrect SCS Configuration

- Make sure that the `distributed_mode` configuration option is set to the `ON` value in the `general` section in each configured SCS `Application` object's `Options`.
- Make sure that the Message Server `Application` object that you dedicated to support Distributed SCS communications is specified on each configured SCS `Application` object's `Connections`.

## Incorrect SCS Role Configuration

- If you decide not to have a main Distributed SCS control all unassigned configuration objects, make sure that the `distributed_rights` configuration option is either not configured or not set to the `MAIN` value in the `general` section in each configured SCS `Application` object's `Options`.
- If you decide to have a main Distributed SCS control all unassigned configuration objects, make sure that the `distributed_rights` configuration option in the `general` section is set to:
    - The `MAIN` value in the `Options` of the SCS `Application` object you designate as the main Distributed SCS.
    - The `DEFAULT` value in the `Options` for the rest of the SCS `Application` objects.

## Incorrect Configuration of Controlled Objects

- If you decide not to have a main Distributed SCS control all unassigned configuration objects, make sure that you assign a particular SCS to each `Host, Application,` and `Solution` object:
    - For `Host` objects, specify a Distributed SCS for the `Host` object.
    - For `Application` objects, specify a Distributed SCS for the `Host` object with which the Application is associated.
    - For `Solution` objects, specify a Distributed SCS for the `Solution` object.

Appendix

# A mlcmd Command-Line Utility

This appendix describes how to use the mlcmd command-line utility, with which you can:

- Query the status of hosts, applications, or solutions.
- Start, stop, and gracefully stop applications and solutions.
- Send a custom command to an application.

This appendix contains the following sections:

## Installing the Utility

If you installed Solution Control Server, the mlcmd utility is already installed, and is located in the same folder in which Solution Control Server was installed.

Starting in 8.1.2, you can install the Solution Control Server utilities without installing Solution Control Server itself. If you have not installed the utilities, refer to the section “Installing Solution Control Server Utilities Separately” in the *Framework 8.1 Deployment Guide*. After the utilities are installed, the mlcmd utility is stored in the location you specified during the installation.

# How to Use the Utility

All mlcmd command parameters are made in a single command. The general syntax is as follows:

```
mlcmd <mandatory parameters> <optional parameters> <command
parameter>
```

The user is required to provide all the mandatory parameters and one operation parameter. The parameters are listed in Table 36 on page 174. Starting in release 8.1, you must authenticate yourself with mlcmd by logging in to the utility. If authentication is successful, you can use the utility as part of operations. Use the parameters in Table 36 on page 174.

**Table 36: mlcmd Parameters**

| Parameter | Description |
|---|---|
| **COMMON PARAMETERS** | |
| -help | Prints the version of the utility and its usage. |
| -cshost <*hostname*> | Mandatory. Configuration Server host name. |
| -csport <*portnumber*> | Mandatory. Configuration Server port number. |
| -csuser <*username*> | Mandatory. Username of user. |
| -cspassword <*password*> | Mandatory. The password of the user. |
| -csappname <*application_name*> | Mandatory. Name of Configuration Manager application. |
| -scshost <*SCS_host_name*>[a] | Optional. Solution Control Server host name. |
| -scsport <*SCS_port_name*>[a] | Optional. Solution Control Server port number |
| -timeout <*seconds*> | Optional. Specifies the amount of time (in seconds) that the utility will wait for SCS to perform the requested action. If SCS does not respond within the specified time, the utility will return an error. Default value = 60 seconds. |
| -secure | Optional. Specifies that a secure connection should be used by clients when connecting to SCS. |
| -cert <*certificate*> | Optional. Use only if -secure is used.<br>On Windows, specifies the security certificate thumbprint; on UNIX, specifies the path to the host's security certificate. |
| -key <*key*> | Use only if -secure is used.<br>On UNIX, specifies the path to the file with the private key; not used on Windows. |

### Table 36: mlcmd Parameters (Continued)

| Parameter | Description |
|---|---|
| -ca-cert <*ca-cert*> | Use only if -secure is used.<br>On UNIX, specifies the path to the file with the CA certificate; not used on Windows. |
| **WARNING!** Specify only one of the following parameters and any associated sub-parameters in a single invocation of the utility command. | |
| **COMMAND PARAMETERS** | |
| -getappstatus <*Application Name*> \| <*DBID*> *[-usedbid*] | Requests the status of the application specified by *Application Name* (the default), or the *DBID* if *usedbid* is specified. |
| -getappstatus-runmode <*Application Name*> \| <*DBID*> *[-usedbid*] | Requests the status and runmode of the application specified by *Application Name* (the default), or the *DBID* if *usedbid* is specified. |
| -gethoststatus <*Host Name*> \| <*DBID*> *[-usedbid*] | Requests the status of the host specified by *Host Name* (the default), or the *DBID* if *usedbid* is specified. |
| -getsolstatus <*Solution Name*> \| <*DBID*> *[-usedbid*] | Requests the status of the solution specified by *Solution Name* (the default), or the *DBID* if *usedbid* is specified. |
| -clear-app-alarms <*Application Name*> \| <*DBID*> *[-usedbid*] | Clears all active alarms raised by the Application specified *by the Application Name* (the default), or the *DBID* if *usedbid* is specified. |
| -clear-cond-alarms (-dbid <*Alarm Condition DBID*>) \| (-name "<*Alarm Condition Name*>") | Clears all active alarms raised on behalf of the Alarm Condition with a DBID specified by *Alarm Condition DBID* or with the name specified by *Alarm Condition Name*. |
| -startapp <*Application Name*> \| <*DBID*> *[-usedbid*] | Starts the application with the specified *Application Name* (the default), or the *DBID* if *usedbid* is specified. |
| -stopapp <*Application Name*> \| <*DBID*> *[-usedbid*] | Stops the application with the specified *Application Name* (the default), or the *DBID* if *usedbid* is specified. |
| -switchapp <*Application Name*> \| <*DBID*> *[-usedbid*] | Switches over the application with the specified *Application Name* (the default), or the *DBID* if *usedbid* is specified. |
| -stopapp-graceful <*Application Name*> \| <*DBID*> *[-usedbid*] | Gracefully stops the application with the specified *Application Name* (the default), or the *DBID* if *usedbid* is specified. |
| -startsol <*Solution Name*> \| <*DBID*> *[-usedbid*] | Starts the solution with the specified *Solution Name* (the default), or the *DBID* if *usedbid* is specified. |
| -stopsol <*Solution Name*> \| <*DBID*> *[-usedbid*] | Stops the solution with the specified *Solution Name* (the default), or the *DBID* if *usedbid* is specified. |

### Table 36: mlcmd Parameters (Continued)

| Parameter | Description |
|---|---|
| -stopsol-graceful <*Solution Name*> \| <*DBID*> [*-usedbid*] | Gracefully stops the solution with the specified *Solution Name* (the default), or the *DBID* if *usedbid* is specified. |
| -get-app-performance {<*Application Name*> \| <*DBID*> [*-usedbid*]} [-result <*xml file name*>] | Requests information about a given process of an application with the specified *Application Name* (the default), or the *DBID* if *usedbid* is specified.<br>The information, including CPU Usage for each thread, is stored in an XML file named `<application name>_Performance_<time stamp>.xml` or `<user-defined name>.xml`. |
| **CUSTOM COMMAND PARAMETERS** | |
| -custom-command <*Application Name*> \| <*DBID*> [*-usedbid*] | Sends custom commands to the application with the specified *Application Name* (the default), or the *DBID* if *usedbid* is specified.<br>Must be used with one or both of the -custom-subcommand and -custom-data parameters. |
| -custom-subcommand <*subcommand*><br>Where <*subcommand*> is a number<br>-custom-data <*path*><br>Where <*path*> is the path to a data file | Use only with the -custom-command option.<br>Both of these options specify the content of a custom data packet as follows:<br>• If the -custom-subcommand option is used, the <subcommand> value is converted to a 4 B value (network byte order) and added to the custom data packet.<br>• If the -custom-data option is used, the contents of the file given by <path> is added to the custom data packet.<br>The custom data packet is then sent to the application with a custom command. |
| -custom-response <*path*> | Use only with the -custom-command option.<br>Specifies the path to the file in which the response to the custom command is to be stored. If this parameter is not used, the response is discarded. |
| -custom-print-response | Use only with the -custom-command option.<br>Specifies that the first 4 B of the custom response must be converted from network byte order into a decimal format and sent to the standard output. |

a. If `-scshost` and `-scsport` are not specified, and if authentication is successful, mlcmd will retrieve the SCS connection parameters from Configuration Server. If there is more than one SCS, the utility will select the first SCS it finds, and retrieve the host and port information from the application object for that SCS.

# Utility Output

For all parameters, the utility returns a numeric code when it has finished, regardless of whether execution was successful. This code can then be used in downstream processing as necessary. A full list of return codes is given in the section “Return Codes” on page 177.

If any errors occur when processing this utility, a log message is generated and sent to `stderr`. Output is never sent to `stdout` unless the `-help` or `-custom-print-response` parameters are specified.

The parameter `get-app-performance` enables you to output CPU Usage data in an xml file. See “XML Data Output for Information Query” on page 179

## Return Codes

A zero-value (0) or a positive two-digit return code indicates that processing was completed successfully. If the command included one of the parameters used to retrieve the status of a `host`, `application`, or `solution`, the return code indicates the status. See Tables 37 and 38.

A negative return code, or on UNIX, a positive value in the range of 249 to 255, indicates that processing did not complete successfully. See Table 39 on page 179.

### Table 37: Success Codes Returned by mlcmd Utility

| Parameter Used | Code | Description |
|---|---|---|
| -gethoststatus | 0 | Host status is UNKNOWN. |
| | 1 | Host status is DISCONNECTED. |
| | 2 | Host status is RUNNING. |
| | 3 | Host status is UNAVAILABLE. |
| | 4 | Host status is UNREACHABLE. |

**Table 37: Success Codes Returned by mlcmd Utility (Continued)**

| Parameter Used | Code | Description |
|---|---|---|
| -getappstatus | 0 | Application status is UNKNOWN. |
| | 1 | Application status is STOPPED. |
| | 2 | Application status is STOP_TRANSITION. |
| | 3 | Application status is STOP_PENDING. |
| | 4 | Application status is START_TRANSITION. |
| | 5 | Application status is START_PENDING. |
| | 6 | Application status is RUNNING. |
| | 7 | Application status is INITIALIZING. |
| | 8 | Application status is SERVICE_UNAVAILABLE. |
| | 9 | Application status is SUSPENDING. |
| | 10 | Application status is SUSPENDED. |
| -getsolstatus | 0 | Solution status is UNKNOWN. |
| | 1 | Solution status is STOPPED. |
| | 2 | Solution status is STOP_PENDING. |
| | 3 | Solution status is START_PENDING |
| | 4 | Solution status is STOP_TRANSITION. |
| | 5 | Solution status is START_TRANSITION. |
| | 6 | Solution status is RUN_PENDING. |
| | 7 | Solution status is RUNNING. |
| | 8 | Solution status is STARTED. |
| -getappstatus-runmode | 0 | Execution completed successfully. |
| | The code returned by -getappstatus-runmode is a combination of the application status and the runmode. See Table 38 to separate the two elements. | |
| All others not mentioned previously | 0 | Execution completed successfully. |

### Table 38: Codes Returned Using -getappstatus-runmode Parameter

| Runmode | Application Status (see -get_appstatus in Table 37 above) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 (PRIMARY) | 0[a] | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 (BACKUP) | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 |
| 2 (EXIT) | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 |

[a] Execution completed successfully.

### Table 39: Error Codes Returned by mlcmd Utility

| Code | | Description |
|---|---|---|
| All but UNIX | UNIX | |
| -1 | 255 | Cannot connect to SCS. |
| -2 | 254 | Unexpected disconnection from SCS. |
| -3 | 253 | Timeout expired. |
| -4 | 252 | No specified object found. |
| -5 | 251 | Management Layer cannot execute the operation for some reason. |
| -6 | 250 | Incorrect parameters or unsuccessful authentication. |
| -7 | 249 | Internal utility error. |

# XML Data Output for Information Query

When you use the `get-app-performance` command, the results are stored in an XML file named as follows:

- Default: `<application name>_Performance_<timestamp>.xml`
- User specifies name in command: `<User-specified name>.xml`

The format of the record in the XML file is as follows:

```
<AppName> <!·The Application name-->
   <getprocessinfo>
      <process>
         <pid>94236</pid>
         <execname>MessageServer.exe</execname>
         <vmsize>4243456</vmsize>
         <pctcpu>0.00</pctcpu>
         <priority>normal</priority>
         <cmdline>cmdline</cmdline>

         <threads> <!-- The section can be absent if the application
            is running on a host with old version of LCA -->

            <thread>
               <tid>1st thread ID</tid>
                  <pctcpu>1st thread CPU usage</pctcpu>
            </thread>

            <thread>
               <tid>2nd thread ID</tid>
                  <pctcpu>2nd thread CPU usage</pctcpu>
            </thread>

            ...

            <thread>
               <tid>Nth thread ID</tid>
                  <pctcpu>Nth thread CPU usage</pctcpu>
            </thread>
         </threads>
      </process>
   <getprocessinfo>
   <ProcSizeInBytes/>
</AppName>
```

Appendix

# B logmsg Command-Line Utility

The logmsg utility enables you to configure log messages to be sent to a specific Message Server on behalf of the specified application. For example, if a specific situation occurs, you can call the utility to send a log message to a specific Message Server for information purposes or to perform an appropriate action. You can also use this utility to generate and clear Management Layer alarms based on conditions external to the Management Layer. This appendix describes the logmsg utility, and how to use it.

This appendix contains the following sections:

## Overview

The logmsg utility was developed to serve the need of customer scripts to raise alarms in the Management Layer. The utility is packaged with Solution Control Server, and uses existing Management Layer functionality to generate and clear Management Layer alarms based on conditions external to the Management Layer. When an external alarm condition is detected, such as by a script, the logmsg utility is called to send a log message with the corresponding Log Event ID. SCS then triggers an active alarm specified by the Alarm Condition identified by the Detect Log Event ID.

The logmsg utility is used by the stuck calls scripts (see "Using the gstuckcalls Utility" on page 86). However, you can also use the utility independently of the stuck calls scripts, as described in this Appendix.

# Installing the Utility

If you installed Solution Control Server, the logmsg utility is already installed, and is located in the same folder in which Solution Control Server was installed.

Starting in 8.1.2, you can install the Solution Control Server utilities without installing Solution Control Server itself. If you have not installed the utilities, use the procedure “Installing Solution Control Server Utilities” in the *Framework 8.1 Deployment Guide*. After the utilities are installed, the logmsg utility is stored in the location you specified during the installation.

# How to Use the Utility

The logmsg utility is located in the same folder in which Solution Control Server was installed.

All logmsg command parameters are made in a single command. The general syntax is as follows:

```
logmsg -h <host> -p <port> -lmid <ID> -aname <name> -atype <type>
-adbid <dbid> -hname <host>[-urgent <level>] -lmtext <text>
[-lmattfile <filename>]
```

All parameters are provided in Table 40, and are mandatory unless stated otherwise in the table.

**Table 40: logmsg Utility Parameters**

| Parameter | Description |
|---|---|
| -h <host> | The name of the Host on which is running the Message Server to which you are sending the log message. |
| -p <port> | The Message Server port through which logmsg communicates with Message Server. |
| -lmid <ID> | The ID of the log event to be sent. |
| -aname <name> | The name of the application on behalf of which the log message is being sent. |
| -atype <type> | The type of the application on behalf of which the log message is being sent. |
| -adbid <dbid> | The DBID of the application on behalf of which the log message is being sent. |

**Table 40: logmsg Utility Parameters (Continued)**

| Parameter | Description |
|---|---|
| -hname <host> | The name of the host on which the application is running. |
| -urgent <level> | The urgency level of the log message, as follows:<br>0 - NONE<br>1 - DEBUG<br>2 - INFORMATION<br>3 - INTERACTION<br>4 - ERROR<br>5 - ALARM<br>This parameter is optional. |
| -lmtext <text> | The text of the log message. |
| -lmattfile <filename> | The name of the file containing the log message attribute pairs, with each pair consisting of an <attribute_name> and an <attribute_value>. This parameter is optional. |

# Configuring Alarms for Events External to the Management Layer

You can also use the logmsg utility to generate and clear Management Layer alarms based on conditions external to the Management Layer. For example, you may configure an alarm to be triggered in response to a stuck call.

When an external alarm condition is detected, such as by a script, call the `logmsg` utility to send a log message with the corresponding Log Event ID. SCS then triggers an active alarm specified by the Alarm Condition identified by the Detect Log Event ID.

Before you use this utility for this purpose, you must configure one `Alarm Condition` object for each condition for which an alarm is to be triggered. Each condition must specify a Detect Log Event ID and a Cancel Log Event ID.

**Note:** Detect and Cancel Log Events created in this situation must have an event ID in the range of `9600–9800`; this range is reserved for custom alarms.

Genesys

Appendix

# C Log Event Specifications

The *Framework Combined Log Events Help* file contains descriptions of all log events generated by Genesys components.

This Appendix contains the following sections:

## Overview

The *Framework Combined Log Events Help* file is built into the software for Solution Control Interface (SCI). If you have not installed SCI, you can also download it from the Genesys Technical Support website.

This Help file is continually updated as new and updated Genesys components are released. To obtain the latest version, download it from the Genesys Technical Support website.

## Downloading from the Technical Support Website

To obtain the most recent version of the *Framework Combined Log Events Help* file, do the following:

1. From the Technical Support Website, download the appropriate release of the *Framework Combined Log Events Help* file to your local computer. The Help file is called `fr_logevents-help.zip`.

2. On your local machine, extract the contents of this file to a new or empty folder.
3. In this folder, open the file `fr_logevents-help.chm`.

# Accessing from Solution Control Interface

If you have Solution Control Interface installed, you can access the *Framework Combined Log Events Help* file separately, using any of the following methods:

- Open the file from the directory in which SCI is installed. See "Accessing on your File System" on page 187.
- Open the file in SCI using the Alarm Condition Wizard. See *Framework 8.1 Solution Control Interface Help*, or use the procedure "Solution Control Interface—Viewing the Combined Log Events Help file using the Alarm Condition Wizard" below.
- Open the file in SCI using the Centralized Log Context Menu. See *Framework 8.1 Solution Control Interface Help*, or use the procedure "Solution Control Interface—Viewing the Combined Log Events Help file using the Centralized Log Context Menu" on page 187.

## Procedure: Solution Control Interface—Viewing the Combined Log Events Help file using the Alarm Condition Wizard

**Purpose:** To view the *Framework Combined Log Events Help* file using the Alarm Condition Wizard in SCI.

### Prerequisites

- SCI is installed and running.

### Start of procedure

1. Log in to SCI.
2. If the toolbar is not displayed, select `Toolbar` from the `View` menu.
3. Select `Run a Wizard` in the toolbar. The `Select Wizard` window dialog box opens.
4. Expand the `Configuration` folder and select `Alarm Condition Wizard`.
5. Click `OK`. The `Select Wizard` window dialog box closes, and the Alarm Condition Wizard starts.
6. On the Welcome screen, click `Next`.

7. On the `Alarm Detection Method` screen, select `Log Event Detection Mode` and click `Next`.
8. On the `Alarm Detection screen`, click `More Information`. The *Framework Combined Log Events Help* file opens.
9. As required, navigate through the Help file, and search for specific log events using the navigation and search tools provided in the Help file.

**End of procedure**

### Procedure: Solution Control Interface—Viewing the Combined Log Events Help file using the Centralized Log Context Menu

**Purpose:** To view the *Framework Combined Log Events Help* file using the Centralized Log Context Menu in SCI. This opens the Help file to the specification for a selected log record.

**Prerequisites**

- SCI is installed and running.

**Start of procedure**

1. Log in to SCI.
2. Open the Centralized Log view.
3. Right-click on a log record. The *Framework Combined Log Events Help* file opens to the specification for the selected log record.
4. As required, navigate through the Help file, and search for other log events using the navigation and search tools provided in the Help file.

**End of procedure**

# Accessing on your File System

If you have Solution Control Interface installed on your computer, you can find the *Framework Combined Log Events Help* file in the folder in which you installed SCI, such as the default folder:

`C:\Program Files\GCTI\Solution Control Interface.`

Navigate to this folder, and double-click on the file named:

`fr_logevents-help.chm.`

# Accessing from Genesys Administrator

The *Framework Combined Log Events Help* file is not included with Genesys Administrator software. To access the file, download the file from the Technical Support website as described above, and use the file separately.

Appendix

# D Genesys MIB File

This appendix presents the content of the Genesys 7 MIB (Management Information Base) file used for the SNMP (Simple Network Management Protocol) support built into the latest release of Management Layer. See also Appendix E on page 243.

```
GENESYS-SML-MIB-G71 DEFINITIONS ::= BEGIN

--- Genesys Solutions Management Layer MIB v7.1

IMPORTS
  DisplayString,
  RowStatus,
  TEXTUAL-CONVENTION  FROM SNMPv2-TC
  MODULE-IDENTITY,
  OBJECT-TYPE,
  NOTIFICATION-TYPE,
  enterprises,
  Unsigned32,
  Integer32,
  TimeTicks           FROM SNMPv2-SMI
  MODULE-COMPLIANCE,
  NOTIFICATION-GROUP,
  OBJECT-GROUP        FROM SNMPv2-CONF;

---
--- MIB High Level description
---

genesys MODULE-IDENTITY
  LAST-UPDATED "200905181300Z"
  ORGANIZATION "Genesys Telecommunications Labs, Inc. An Alcatel Company"
  CONTACT-INFO
     "Postal: 2001 Junipero Serra Blvd
              Daly City, CA 94014
```

```
            US

            Tel:    +1 888 GENESYS
            Fax:    +1 415 437 1260

     email:  support@genesyslab.com"

  DESCRIPTION
    "Collection of managed objects that provides for remote
     management of Genesys Call Center server applications."

  REVISION "200303281500Z"
  DESCRIPTION
    "Initial version."

  REVISION "200304141830Z"
  DESCRIPTION
    "Syntax of some of the managed objects changed from DisplayString to an enumerated
     INTEGER."

  REVISION "200304160630Z"
  DESCRIPTION
    "Included Integer32 into imports.

    Reverted tsDtaType from INTEGER to DisplayString.

    Changed names of enumerated values to begin with lowercase letter.

    Added 'Unknown' values to some enumerators.

    Added 'debug' as possible value for logVerbose object.

    Revised some description clauses to improve managed
    objects specification."

  REVISION "200304230630Z"
  DESCRIPTION
    "Changed definition of gsCtrlRefreshStatus object - the value
     dataRefreshCanceled is removed, dataNotReady is used instead;
     description of the object is changed accordingly.

     Changed definition of gServerStatus object. Syntax changed to
     enumerated INTEGER and description is refined accordingly.

     Removed objects gServerStartup, gServerShutdown, and gServerGracefulShutdown.
     Added object gServerCommand instead that provides functionality
     formerly performed by removed objects.

     Moved object gsCmndDeleteClient from gsInfoEntry to gServersEntry
     and renamed it to gServerDeleteClient."
  REVISION "200305120630Z"
  DESCRIPTION
    "Changed dataRefreshCanceled to dataNotReady in description of gsCtrlRowStatus
object."

  REVISION "200305150630Z"
  DESCRIPTION
    "Changed MAX-ACCESS of gsCtrlRefreshStatus and gsCtrlLastRefreshed to read-only."

  REVISION "200309100000Z"
  DESCRIPTION
    "Added gsCtrlAutomaticRefresh object to the gServerControlTable table."

  REVISION "200311140700Z"
  DESCRIPTION
    "Descriptions of gsAlarmLogText and gsAlarmCategory objects were changed."

  REVISION "200409291800Z"
  DESCRIPTION
    "Added new tables tsCallFilterTable and tsCallInfoTable.
     Consequently extended TableID with new values.

     Added two new variables to the gsMLAlarm's variables-bind
     gsAlarmAppHostName and gsAlarmGUID."

  REVISION "200802271200Z"
  DESCRIPTION
    "Added new extension point apps"

  REVISION "200803141900Z"
  DESCRIPTION
    "Added statuses statusSuspending and statusSuspended"

  REVISION "200905181300Z"
  DESCRIPTION
    "Added new table tsLinkStatsTable"

  ::= { enterprises 1729 }


--- Textual Conventions

TableID ::= TEXTUAL-CONVENTION
  STATUS current
  DESCRIPTION
    "An integer value representing a conceptual data table defined in this MIB module.
     Along with ServerDBID used as an index into the gsControlTable."
  SYNTAX INTEGER
  {
    gsLogTable(1),
    gsInfoTable(2),
    gsClientTable(3),
    gsPollingTable(4),
    tsInfoTable(5),
    tsCallTable(6),
    tsDtaTable(7),
    tsLinkTable(8),
    tsCallFilterTable(9),
    tsCallInfoTable(10),
    tsLinkStatsTable(11)
   }

ServerDBID ::= TEXTUAL-CONVENTION
  STATUS current
  DESCRIPTION
    "An unique identifier for a Genesys managed server as assigned
    by the Configuration Server."
  SYNTAX Unsigned32

HostDBID ::= TEXTUAL-CONVENTION
  STATUS current
  DESCRIPTION
    "An unique identifier for a host as assigned
     by the Configuration Server."
  SYNTAX Unsigned32

SolutionDBID ::= TEXTUAL-CONVENTION
  STATUS current
  DESCRIPTION
    "An unique identifier for a solution as assigned
    by the Configuration Server."
  SYNTAX Unsigned32



-- #############################################################################
-- Object Identifiers
-- #############################################################################

servers OBJECT IDENTIFIER
  ::= { genesys 100 }

hosts OBJECT IDENTIFIER
  ::= { genesys 101 }

solutions OBJECT IDENTIFIER
  ::= { genesys 102 }

notifications OBJECT IDENTIFIER
  ::= { genesys 104 }

apps OBJECT IDENTIFIER
```

```
  ::= { genesys 200 }

genericServer OBJECT IDENTIFIER
  ::= { servers 1 }

specificServer OBJECT IDENTIFIER
  ::= { servers 2 }

tServer OBJECT IDENTIFIER
  ::= { specificServer 1 }

--ursServer OBJECT IDENTIFIER
--  ::= { specificServer x }

--sipServer OBJECT IDENTIFIER
--  ::= { specificServer x }



-- ############################################################################
-- Genesys alarms
-- ############################################################################

gsAlarm NOTIFICATION-TYPE
  OBJECTS { gsServersLastAlarm }
  STATUS  current
  DESCRIPTION
    "Notification message reporting that a Genesys server
     has encountered an alarm situation."
  ::= {notifications 1}

gsMLAlarm NOTIFICATION-TYPE
  OBJECTS
  {
    gsAlarmId,
    gsAlarmLogText,
    gsAlarmMessagesIds,
    gsAlarmApplicationName,
    gsAlarmApplicationType,
    gsAlarmAppHostName,
    gsAlarmCategory,
    gsAlarmGUID
  }

  STATUS current
  DESCRIPTION
    "An alarm message from Management Layer to NMS that a pre-configured alarm condition
     has been detected.
     The details of this alarm are described by the variable list attached to this
     notification message."
  ::= {notifications 2}
```

```
gsServerUpTrap NOTIFICATION-TYPE
  OBJECTS { gsServersLastTrap }
  STATUS  current
  DESCRIPTION
    "Notification message reports that a Genesys server has been started up."
  ::=  {notifications 3}

gsServerDownTrap NOTIFICATION-TYPE
  OBJECTS { gsServersLastTrap }
  STATUS  current
  DESCRIPTION
    "Notification message reports that a Genesys server
    has ceased operation or has failed to restart."
  ::= {notifications 4}

gsPollingSignal NOTIFICATION-TYPE
  OBJECTS { gsPollingLastTrap }
  STATUS  current
  DESCRIPTION
    "Notification message reporting that a polling signal has been
    received from a server.
    The notification message has the following format:
    <server name>:<polling ID>."
  ::=  {notifications 5}

tsLinkStatusTrap NOTIFICATION-TYPE
  OBJECTS { tsLastChangedLinkStatus }
  STATUS current
  DESCRIPTION
    "Notification message reporting that TServer CTI link status has been changed."
  ::= { notifications 6 }


gsCleanupTimeout  OBJECT-TYPE
  SYNTAX       Unsigned32
  MAX-ACCESS   read-write
  STATUS       current
  DESCRIPTION
    "The time in minutes the agent should keep rows in the
    gsControlTable and consequently in related data tables if there were no
    requests to objects of this row or corresponding rows from data
    table(s) and no automatic refresh is activated for this particular row.
    After that time the agent should automatically delete unattended rows.

    Value 0 of this object specifies that MIB clean up should not be performed.
    NOTE: this automatic clean up mechanism applies to the following
    server data tables:

      gsLogTable
      gsInfoTable
      gsClientTable
      tsInfoTable
      tsCallTable
      tsDtaTable
      tsLinkTable
      tsCallFilterTable
      tsCallInfoTable
      tsLinkStatsTable

    NOTE: this automatic clean up mechanism does not apply to the gsPollingTable.
    The gsPollingTable is used to activate (and deactivate) servers heart-bit
notification
    messages (propagated as TRAP messages) and one cannot assume this was the intent of
    the management station to stop receiving such notifications. Note also that one of
the
    columnar objects of this table, gsPollingLastTrap, is used as a variable in the
variable binding
    list of the mentioned notification."
  DEFVAL { 60 }
  ::= { genericServer 1 }


-- ###########################################################################
-- Genesys servers Table (list of servers)
-- ###########################################################################

gServersTable OBJECT-TYPE
  SYNTAX SEQUENCE OF GServersEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "This table populates the servers that are currently registered with the
     Genesys configuration layer.

     This table has several purposes:

     (a) Provides general information about servers working environment
     (b) Allows to start or stop an individual server
     (c) Provides information about DBID numbers assigned to servers.
     These DBID numbers are needed when creating entries in control table.
     Each entry in control table is set to monitor a particular
     server. A DBID number is used to identify that server.

     Note that since server configuration objects can be added or deleted
     to or from configuration database throughout a lifetime of
     the agent, a management station might want to walk this table
     frequently in order to keep the servers list updated."

  ::= { genericServer 2 }

gServersEntry OBJECT-TYPE
```

```
  SYNTAX      GServersEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "A conceptual row in the gServersTable.
     Each deamon application object in configuration database has an entry in
     this table."
  INDEX { gServerId }
  ::= { gServersTable 1 }

GServersEntry ::= SEQUENCE
{
  gServerId                ServerDBID,
  gServerName              DisplayString,
  gServerStatus            INTEGER,
  gServerType              DisplayString,
  gServerVersion           DisplayString,
  gServerWorkDir           DisplayString,
  gServerCommandLine       DisplayString,
  gServerPID               Integer32,
  gServerCommand           INTEGER,
  gServerDeleteClient      Unsigned32
}

gServerId OBJECT-TYPE
  SYNTAX      ServerDBID
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "An index that uniquely identifies a server in the gServersTable.
    Server DBIDs are generated by the Configuration Server when new
    server configuration objects are created and added to the configuration database
    and are unique within a given configuration environment.
    DBID numbers of deleted servers are not reused again but only across power
    cycles of the Configuration Server."
  ::= { gServersEntry 1 }

gServerName OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "A human-readable description of the server as assigned by a user
    that created this server in configuration database.
    This object is used to provide a mapping of a server name to a
    DBID number. The index portion of object identifier represents the
    DBID assigned to this server: gServerName.<index> = <value>
    where index == DBID and value == server name.
    By walking all instances of this object a manager
    can retrieve an updated list of all currently configured servers,
    their names and DBIDs."
  ::= { gServersEntry 2 }

gServerStatus OBJECT-TYPE
  SYNTAX INTEGER
  {
    statusUnknown(1),
    statusStopped(2),
    statusPending(3),
    statusRunning(4),
    statusInitializing(5),
    statusServiceUnavailable(6),
    statusSuspending(7),
    statusSuspended(8)
  }
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Indicates server status:

      - statusUnknown: Indicates that the Management Layer is unable to provide reliable
       information about the current server status. It does not necessarily mean that
       the server is unable to perform its function.

      - statusStopped: Indicates that a server is installed and configured in the
system,
      but has not started. This status indicates that the server either has not been
      activated or has failed.

      - statusPending: Indicates one of the following:

        (a) The server is in the process of being activated. This status only exists
        for the interval between the instruction to start the server and the actual
        readiness of the server to perform its function. Typically, the Pending stage
        involves starting the server, reading configuration data from the Configuration
        Layer, checking this data for integrity and completeness, and establishing
        connections with all the resources according to the given configuration data.

        (b) The server is in the process of being shut down. This status only exists
        for the interval between the instruction to stop the server and its actual
        termination. Typically, the Pending stage involves some server-specific
        wrap-up functions, closure of all open connections, termination, and detection
        of the termination by Local Control Agent.

     - statusRunning: Assigned from the moment a server is completely initialized; that
is,
     when the server has read and checked its configuration and established connections
      with all the required resources. This status does not necessarily mean that
      the server is actually performing its function.

      - statusInitializing: Server is starting and performing initialization. Server is
      connected to LCA. On initialization phase server is not able to provide service.
      - statusServiceUnavailable: Server is running but unable to provide service due
      to unavailability of some resource.

      - statusSuspending: Server is running and handling current requests, but no new
      requests are accepted.

      - statusSuspended: Server is running, all requests are completely handled,
      no new requests are accepted. This status means that server can be stopped
      without any information/request loss."

  ::= { gServersEntry 3 }

gServerType OBJECT-TYPE
  SYNTAX      DisplayString (SIZE(1..256))
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Indicates server type.
     Displayed text is based on mapping of server type index
     to human readable representation as provided by Configuration Server.
     For example: T-Server"
  ::= { gServersEntry 4 }

gServerVersion OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Indicates server version as specified in Configuration Server."
  ::= { gServersEntry 5 }

gServerWorkDir OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Indicates server working directory as specified in Configuration Server."
  ::= { gServersEntry 6 }

gServerCommandLine OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Indicates full command-line used to start this server as specified
    in Configuration Server.
    For example: scs -host enigma -port 4135 -app TServer_Meridian."
  ::= { gServersEntry 7 }

gServerPID OBJECT-TYPE
```

```
  SYNTAX      Integer32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Indicates process ID assigned to this server by operating system.
     It is positive integer for running server, -1 for stopped server
     or 0 if PID is unknown. The latter may happen, for instance, when
     LCA is not running on the server's host."
  ::= { gServersEntry 8 }

gServerCommand OBJECT-TYPE
  SYNTAX INTEGER
  {
    start(1),
    shutDown(2),
    shutDownGracefully(3)
  }
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "This object is used to start, shutdown or gracefully shutdown server with dbid
equal
     to this object instance id.

     When this object is set value the agent either attempts to perform corresponding
     operation, or rejects it if the operation is impossible or does not make sense,
     e.g. if value start(1) is set for server that the agent knows to be running.
     In any case the agent returns to NMS corresponding result (success or failure)
     immediatly.

     Result 'success' does not necessarily means success of the operation. The NMS
     should check the value of object gServerStatus to determine whether
     the operation succeeded or failed."

  ::= { gServersEntry 9 }

gServerDeleteClient OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Sends 'delete client' command to this server.
    To delete a client enter its socket number."
  ::= { gServersEntry 10 }



-- ####################################################################################
-- Servers Control Table
-- ####################################################################################
```

```
gServerControlTable OBJECT-TYPE
  SYNTAX SEQUENCE OF GServerControlEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    "Control table containing a set of parameters to set up and
     control data collection from Genesys server(s).
     This table facilitates the monitoring of multiple Genesys servers.

     The following data tables defined in this MIB module are controlled by the
     gsControlTable:

     - gsLogTable
     - gsInfoTable
     - gsClientTable
     - gsPollingTable
     - tsInfoTable
     - tsCallTable
     - tsDtaTable
     - tsLinkTable
     - tsCallFilterTable
     - tsCallInfoTable
     - tsLinkStatsTable

     Entries in the above tables are created on behalf of an entry in
     the gServerControlTable.
     If a control row is destroyed, then corresponding row in the respective data table
     is destroyed too.

     Some tables defined in this MIB module may just be used to perform
     some management function on a server (e.g.,gsPollingTable).
     When a management station creates a row in the control table for such a table, the
     agent will create a row in the corresponding table thus making it ready to be used
to
     perform a management function for a selected server.
     This way, we are allowed to create multiple instances of a management object that
can
     be used to perform a management function for multiple servers.
     For example, in case of gsPollingTable a manager will be able to perform reconfig
command
     for multiple Stat Servers."

  ::= { genericServer 3 }

gServerControlEntry OBJECT-TYPE
  SYNTAX GServerControlEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    "A conceptual row in the gServerControlTable."
  INDEX { gsCtrlServerID, gsCtrlTableID }
  ::= {gServerControlTable 1 }

GServerControlEntry ::= SEQUENCE
{
  gsCtrlServerID         ServerDBID,
  gsCtrlTableID          TableID,
  gsCtrlRefreshStatus    INTEGER,
  gsCtrlLastRefreshed    TimeTicks,
  gsCtrlAutomaticRefresh Unsigned32,
  gsCtrlRowStatus        RowStatus
}

gsCtrlServerID OBJECT-TYPE
  SYNTAX ServerDBID
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    "A server that is to be managed by this control row."
  ::= { gServerControlEntry 1 }

gsCtrlTableID OBJECT-TYPE
  SYNTAX TableID
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    " A particular data table to be refreshed by this control row."
  ::= { gServerControlEntry 2 }

gsCtrlRefreshStatus OBJECT-TYPE
  SYNTAX INTEGER
  {
    dataNotReady(1),
    dataRefreshInProgress(2),
    dataReady(3),
    mgmtIsNotAvailable(4),
    dataRefreshFailed(5)
  }
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Indicates refresh status of corresponding data table as specified by gsCtrlTableID.

         dataNotReady - indicates that data table refresh has not been attempted yet
                (the row was just created with createAndWait RowStatus value),
               or data table was not refreshed because the user has set gsCtrlRowStatus
                to notInService while refresh was in progress
        dataRefreshInProgress - indicates that data table refresh has been started but
not finished yet
         dataReady - indicates that data table has been successfully refreshed
         mgmtIsNotAvailable - indicates that data table refresh failed for one of the
following reasons:
```

```
                (a) no mgmt port is configured for this server
                (b) this server does not provide for mgmt instrumentation
                (c) this server does not provide for requested data (like if config
server is requested for tsCallsList)
                (d) LCA is not started (or terminated) on host where this server
                is running
        dataRefreshFailed - indicates that refresh failed for any reason other then
described above;
                this situation may typically arise when mgmt connection could not be
established
                or was lost during refresh"
  DEFVAL {1}
  ::= { gServerControlEntry 3 }

gsCtrlLastRefreshed OBJECT-TYPE
  SYNTAX TimeTicks
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    "The time in hundredths of seconds since the row was last successfully refreshed
    (gsCtrlRefreshStatus was set to dataReady) or 0 for the rest of refresh states"
  ::= { gServerControlEntry 4 }

gsCtrlAutomaticRefresh OBJECT-TYPE
  SYNTAX Unsigned32
  MAX-ACCESS read-create
  STATUS current
  DESCRIPTION
    "The time in seconds the agent should perform automatic refresh for
    a data table managed by this control row.
    Any value other then zero activates the automatic refresh
    mechanism, otherwise automatic refresh is disabled (default)."
  DEFVAL {0}
  ::= { gServerControlEntry 5 }

gsCtrlRowStatus OBJECT-TYPE
  SYNTAX RowStatus
  MAX-ACCESS read-create
  STATUS current
  DESCRIPTION
    "The status of this conceptual control entry.

     The agent may change the value of this object to 'notInService' if a
     connection to a managed server has been lost or some internal error has
     occurred. The manager also sets this object to 'active' in order to
     refresh table(s).

     The following is a detailed description of the object's functions
     and data refresh mechanism:

    (a) Creating new conceptual row.
```

```
       Management station should set value ·createAndWait'
       for gsCtrlRowStatus with instance id specifying managed server and refreshed
       table. Agent MUST create a row and set gsCtrlRowStatus to ·notReady' and return
       noError. No data refresh is performed.

       Alternatively, management station may set value ·createAndGo'. In this case
       the agent MUST create a row, set gsCtrlRowStatus to ·active', returnnoError
       and immediatly start data refresh.

      (b) Refreshing data tables.

      In order to refresh the table specified by row instance id the manager should set
      value of object gsCtrlRowStatus 'active'. In response, the agent MUST set
      gsCtrlRefreshStatus to dataRefreshInProgress, start data table refresh, and
      return noError.

      Upon successful completion of data table refresh the agent MUST set
      gsCtrlRefreshStatus to dataReady.

      In some cases data retrieval may not be completed successfully. In
      such case, the agent MUST set gsCtrlRefreshStatus to a value indicating the
      reason of failure and delete all corresponding rows from the refreshed table.

      (c) Canceling data collection in progress.

      If the agent receives request to set gsCtrlRowStatus to ·notInService' when
      data collection is in progress, it MUST interrupt data collection, delete
      corresponding rows from data table and set gsCtrlRefresh to dataNotReady.

      (d) Cleaning data tables upon management station request.

      If the agent receives request to set gsCtrlRowStatus to ·destroy' it MUST
      interrupt data collection in progress (if any), delete the row from
      gServerControlTable and all corresponding rows from data table.

      (e) Automated cleaning of data tables.

      The agent MUST keep track of the times when the last SNMP request to
      gServerControlTable row or corresponding data table rows occur for each
      gServerControlTable row. If no requests occur during threshold time
      interval specified by gsCleanupTimeout object, corresponding rows MUST
      be deleted from gServerControlTable and data tables. That would save memory
      in case when management station *forgets* to clean data tables.

      (f) Data tables should always contain snapshots of corresponding information and
      the agent MUST NOT update them the way other then described above. The only
      exception is gsPollingTable that MUST be automatically updated in response to
      notifications from managed servers."

    ::= { gServerControlEntry 6 }
```

```
-- ####################################################################################
-- Generic Server - Info Table
-- ####################################################################################

gsInfoTable OBJECT-TYPE
  SYNTAX SEQUENCE OF GsInfoEntry
  MAX-ACCESS  not-accessible
  STATUS current
  DESCRIPTION
    " Collection of statistics about server's client. As well it contains object
      to send delete client command to a server."
  ::= { genericServer 4 }

gsInfoEntry OBJECT-TYPE
  SYNTAX      GsInfoEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    " "
  INDEX { gsCtrlServerID }
  ::= { gsInfoTable 1 }

GsInfoEntry ::= SEQUENCE
{
  gsClientsExistNum       Unsigned32,
  gsClientsTotalNum       Integer32,
  gsServerConfigFile      DisplayString
}

gsClientsExistNum OBJECT-TYPE
  SYNTAX Unsigned32
  MAX-ACCESS read-only
  STATUS current
  DESCRIPTION
    "Total number of clients that can be handled by a server."
  ::= { gsInfoEntry 1 }

gsClientsTotalNum OBJECT-TYPE
  SYNTAX      Integer32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "The total number of clients currently connected to server."
  ::= { gsInfoEntry 2 }

gsServerConfigFile OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Indicates config file name that was used to start a server, if any."
  ::= { gsInfoEntry 3 }

-- ####################################################################################
-- Generic Server - Log Table
-- ####################################################################################

gsLogTable OBJECT-TYPE
  SYNTAX SEQUENCE OF GsLogEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    " "
  ::= { genericServer 5 }

gsLogEntry OBJECT-TYPE
  SYNTAX     GsLogEntry
  MAX-ACCESS not-accessible
  STATUS     current
  DESCRIPTION
    " "
  INDEX { gsCtrlServerID }
  ::= { gsLogTable 1 }

GsLogEntry ::= SEQUENCE
{
  logVerbose       DisplayString,
  logTrace         DisplayString,
  logStandard      DisplayString,
  logDebug         DisplayString,
  logAll           DisplayString,
  logBuffering     DisplayString,
  logSegment       DisplayString,
  logExpire        DisplayString,
  logMessageFile   DisplayString,
  logMessageFormat DisplayString,
  logTimeFormat    DisplayString,
  logTimeConvert   DisplayString
}

logVerbose OBJECT-TYPE
  SYNTAX     DisplayString (SIZE(1..10))
  MAX-ACCESS read-write
  STATUS     current
  DESCRIPTION
    "Specifies logging verbose mode. Determines the minimum level
     of log events to be generated by a server.
     Activates or deactivates server event log feature according to
     the verbose mode specified.
     Possible settings: one of the following keywords:
```

```
     none
     debug
     trace
     standard
     all

     The SET request takes effect immediately."
  ::= { gsLogEntry 1 }

logTrace OBJECT-TYPE
  SYNTAX     DisplayString
  MAX-ACCESS read-write
  STATUS     current
  DESCRIPTION
    "Specifies the log outputs that log events of the
     Trace level are to be sent to.
     Possible settings: any combination of the following key words
     separated by space:

     stdout
     stderr
     filename
     network

     The SET request takes effect immediately."
  ::= { gsLogEntry 2 }

logStandard OBJECT-TYPE
  SYNTAX     DisplayString
  MAX-ACCESS read-write
  STATUS     current
  DESCRIPTION
    "Specifies the log outputs that log events of the Standard
     level are to be sent to.
     Possible settings: any combination of the following keywords
     separated by space:

     stdout
     stderr
     filename
     network

     Note: to avoid network congestion log events of Standard level
     can be logged locally only.
     The SET request takes effect immediately."
  ::= { gsLogEntry 3 }

logDebug OBJECT-TYPE
  SYNTAX      DisplayString (SIZE (0..22))
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Specifies the log outputs that log events of the Debug
    level are to be sent to.
    Possible settings: any combination of the following keywords
    separated by space:

      stdout
      stderr
      filename

    Note: to avoid network congestion log events of Debug level
    can be logged locally only.
    The SET request takes effect immediately."
  ::= { gsLogEntry 4 }

logAll OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Specifies the log outputs that log events of all levels are to be sent to.
     Possible settings: any combination of one or more of
     the following keywords separated by space:

     stdout
     stderr
     filename
     network

     Note: to avoid network congestion log events of Debug level
     can be logged locally only.
     The SET request takes effect immediately."
  ::= { gsLogEntry 5 }

logBuffering OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Turns on/off operating system file buffering.
     To set this object use the following keywords:
       true
       false
     The SET request takes effect immediately."
  ::= { gsLogEntry 6 }

logSegment OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Specifies log file segmentation mode.
    When currently opened log segment exceeds the size set by this
    option, then current segment is closed and a new empty
    log segment is created.
    Possible settings: one of the following keywords:

      false
      <number> KB
      <number> MB
      <number> hr

    Setting object to a value other than false will activate log
    segmentation feature according to the mode specified.
    The SET request takes effect immediately."
  ::= { gsLogEntry 7 }

logExpire OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Specifies log file expiration mode.
    When old file is expired, new file is created while the old one is removed.
    Possible settings: one of the following keywords:

      false
      <number> file
      <number> day

    Setting object to a value other than false will activate log
    expiration feature according to the expiration method specified.
    The SET request takes effect immediately."
  ::= { gsLogEntry 8 }

logMessageFile OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Specifies the file name for application-specific log events.
    Valid filename is of the following format:

      <filename>.lms

    The SET request takes effect after an application is restarted.
    NOTE: object intended for use by Genesys personnel only."
  ::= { gsLogEntry 9 }


logMessageFormat OBJECT-TYPE
```

```
  SYNTAX      DisplayString (SIZE(1..10))
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Specifies the format of log record headers that an
     application uses when writing logs in the log file. Using compressed
     log record headers improves application performance and reduces the
     log file size.
     Possible settings: one of the following keywords:

     full
     short

     The SET request takes effect immediately."
  ::= { gsLogEntry 10 }

logTimeFormat OBJECT-TYPE
  SYNTAX      DisplayString (SIZE(1..10))
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Specifies how to represent in a log file the time when
     an application generates log records.
     Possible settings: one of the following keywords:

    locale
    iso8601

    The SET request takes effect immediately."
  ::= { gsLogEntry 11 }

logTimeConvert OBJECT-TYPE
  SYNTAX      DisplayString (SIZE(1..10))
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Specifies in which system an application calculates the log record time
     when generating a log file. The time is converted from the time in seconds
     since the Epoch (00:00:00 UTC,January 1, 1970).
     Possible settings: one of the following keywords:

    local
    utc

    The SET request takes effect immediately."
  ::= { gsLogEntry 12 }


-- ###################################################################################
-- Generic Server Polling Table
-- ###################################################################################
```

```
gsPollingTable OBJECT-TYPE
  SYNTAX SEQUENCE OF GsPollingEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    " A set of parameters to control the server's polling feature. As well as
      contains an object to watch the progress of the polling signals from
      a particular server."
  ::= { genericServer 6 }

gsPollingEntry OBJECT-TYPE
  SYNTAX GsPollingEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    " "
  INDEX { gsCtrlServerID }
  ::= { gsPollingTable 1 }

GsPollingEntry ::= SEQUENCE
{
  gsPollingStatus      DisplayString,
  gsPollingInterval    Unsigned32,
  gsPollingID          Unsigned32,
  gsPollingLastTrap    DisplayString
}

gsPollingStatus OBJECT-TYPE
  SYNTAX      DisplayString (SIZE(1..10))
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Activates or deactivates server polling feature, which
     sends NMS periodical signals describing server
     operational status. Possible settings on/off."
  ::= { gsPollingEntry 1 }

gsPollingInterval OBJECT-TYPE
  SYNTAX      Unsigned32 (0..120)
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Specifies interval in seconds between polling signals from server."
  ::= { gsPollingEntry 2 }

gsPollingID OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
```

```
    "Event ID assigned to each polling signal, used by NMS for monitoring purposes."
  DEFVAL {1}
  ::= { gsPollingEntry 3 }

gsPollingLastTrap OBJECT-TYPE
  SYNTAX      DisplayString (SIZE(1..64))
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Last ID assigned to polling signal by the agent."
  ::= { gsPollingEntry 4 }

-- ##################################################################################
-- Generic Server Client Table
-- ##################################################################################

gsClientTable OBJECT-TYPE
  SYNTAX SEQUENCE OF GsClientEntry
  MAX-ACCESS not-accessible
  STATUS     current
  DESCRIPTION
    " "
  ::= { genericServer 7 }

gsClientEntry OBJECT-TYPE
  SYNTAX      GsClientEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    " "
  INDEX { gsCtrlServerID, gsClientSocket }
  ::= { gsClientTable 1 }

GsClientEntry ::= SEQUENCE
{
  gsClientSocket      Unsigned32,
  gsClientAppName     DisplayString,
  gsClientAuthorized  Integer32,
  gsClientType        INTEGER,
  gsClientGotEvents   Unsigned32,
  gsClientSentReqs    Unsigned32
}

gsClientSocket OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "Specifies socket number through which client connected to server."
  ::= { gsClientEntry 1 }
```

```
gsClientAppName OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies client app name."
  ::= { gsClientEntry 2 }

gsClientAuthorized OBJECT-TYPE
  SYNTAX      Integer32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies level of client authorization."
  ::= { gsClientEntry 3 }

gsClientType OBJECT-TYPE
  SYNTAX INTEGER
  {
    typeTClient(0),
    typeMClient(1),
    typeEXRClient(2),
    typeUnknown(3)
  }
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Indicates client type."
  ::= { gsClientEntry 4 }

gsClientGotEvents OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies number of events received by clients."
  ::= { gsClientEntry 5 }

gsClientSentReqs OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies number of requests sent by client."
  ::= { gsClientEntry 6 }


-- ############################################################################
-- Generic server alarm objects
-- ############################################################################
```

```
gsAlarmObjects OBJECT IDENTIFIER
  ::= { genericServer 10 }

gsServersLastAlarm OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies the last alarm message sent to NMS.
     Note: object provides for backward compatibility in Genesys v5."
  ::= { gsAlarmObjects 11 }

gsServersLastTrap OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies the last server-status (server up or down) trap sent to NMS.
     Note: object provides for backward compatibility in Genesys v5."
  ::= { gsAlarmObjects 12 }

gsAlarmId OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  accessible-for-notify
  STATUS      current
  DESCRIPTION
    "Uniquely identifies an alarm condition upon
    which Management Layer can react with sending of SNMP
    Trap. The alarm condition is configured in
    the Configuration Server and assigned with an unique
    configuration object ID (smlAlarmId) which is
    unique within the Genesys Configuration Database."
  ::= { gsAlarmObjects 13 }

gsAlarmLogText OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  accessible-for-notify
  STATUS      current
  DESCRIPTION
    "A textual description of an alarm event.
     For example: Connection to [server type] [servername] at host [hostname], port
[port number] lost."
  ::= { gsAlarmObjects 14 }

gsAlarmMessagesIds OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  accessible-for-notify
  STATUS      current
  DESCRIPTION
    "Uniquely identifies log event (or a set of log events) upon which an
    active alarm condition has been created (or deleted).
    Currently, only one log event can create or destroy an Alarm condition.
    For example, 4003."
  ::= { gsAlarmObjects 15 }

gsAlarmApplicationName OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  accessible-for-notify
  STATUS      current
  DESCRIPTION
    "Application name which is the source of this alarm. For example: tserver."
  ::= { gsAlarmObjects 16 }

gsAlarmApplicationType OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  accessible-for-notify
  STATUS      current
  DESCRIPTION
    "Application type which is the source of this alarm. For example: G3tserver."
  ::= { gsAlarmObjects 17 }

gsAlarmCategory OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  accessible-for-notify
  STATUS      current
  DESCRIPTION
    "A category assigned to this alarm. The following
     categories are defined: Critical, Major, Minor, Clearance.
     Category Clearance indicates that this notification is sent in response to an
     active alarm being canceled."
  ::= { gsAlarmObjects 18 }


gsAlarmGUID OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  accessible-for-notify
  STATUS      current
  DESCRIPTION
    " Uniquely identifies Alarm Clearance Trap with Alarm Creation Trap."
  ::= { gsAlarmObjects 19 }


gsAlarmAppHostName OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  accessible-for-notify
  STATUS      current
  DESCRIPTION
    "Host name the application that generated the alarm is running on."
  ::= { gsAlarmObjects 20 }
```

```
-- ###################################################################################
-- TServer Info Table
-- ###################################################################################
tsInfoTable OBJECT-TYPE
  SYNTAX SEQUENCE OF TsInfoEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    "This table contains some statistical data about TServer."
  ::= { tServer 1 }

tsInfoEntry OBJECT-TYPE
  SYNTAX     TsInfoEntry
  MAX-ACCESS not-accessible
  STATUS     current
  DESCRIPTION
    " "
  INDEX { gsCtrlServerID }
  ::= { tsInfoTable 1 }

TsInfoEntry ::= SEQUENCE
{
  tsCallsExistNum          Unsigned32,
  tsCallsTotalNum          Unsigned32,
  tsLinksCommand           DisplayString,
  tsLastChangedLinkStatus  DisplayString
}

tsCallsExistNum OBJECT-TYPE
  SYNTAX     Unsigned32
  MAX-ACCESS read-only
  STATUS     current
  DESCRIPTION
    "Specifies number of calls being handled by TServer."
  ::= { tsInfoEntry 1 }

tsCallsTotalNum OBJECT-TYPE
  SYNTAX     Unsigned32
  MAX-ACCESS read-only
  STATUS     current
  DESCRIPTION
    "Specifies total number of calls handled by TServer."
  ::= { tsInfoEntry 2 }

tsLinksCommand OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Specifies command which will be sent to T-Server."
  ::= { tsInfoEntry 3 }
```

```
tsLastChangedLinkStatus OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Last changed link status sent from T-Server."
  ::= { tsInfoEntry 4 }


-- ######################################################################################
-- T-Server Calls Table
-- ######################################################################################

tsCallTable OBJECT-TYPE
  SYNTAX SEQUENCE OF TsCallEntry
  MAX-ACCESS not-accessible
  STATUS     current
  DESCRIPTION
    " A collection of telephony calls data for one or more
      T-Server (for which an entry has been created in the
      tsControlTable). "
  ::= { tServer 2 }

tsCallEntry OBJECT-TYPE
  SYNTAX     TsCallEntry
  MAX-ACCESS not-accessible
  STATUS     current
  DESCRIPTION
    "A conceptual row in the tsCalls table.

     The gsCtrlServerID identifies a managed TServer, and along with the TableID (6),
     identifies a row in the gServerControlTable on which behalf this entry was created.

     The tsCallInstanceID is the call instance ID assigned by the management library.

     An example of the indexing of this entry is
     tsCallType.101.1508, where 101 is the server ID and 1508 is the call instance ID."

  INDEX {gsCtrlServerID, callInstanceID }
  ::= { tsCallTable 1 }

TsCallEntry ::= SEQUENCE
{
  callInstanceID           Unsigned32,
  callConnID               DisplayString,
  callState                INTEGER,
  callCallID               Unsigned32,
  callType                 INTEGER,
  callReferenceID          Unsigned32,
  callTimeStamp            TimeTicks,
  callDNIS                  DisplayString,
  callANI                   DisplayString,
  callNumParties            Unsigned32,
  callPartiesList           DisplayString,
  callCustomerID            DisplayString,
  callFirstTransferLocation DisplayString,
  callFirstTransferDN       DisplayString,
  callLastTransferLocation  DisplayString,
  callLastTransferDN        DisplayString
}

callInstanceID OBJECT-TYPE
  SYNTAX     Unsigned32
  MAX-ACCESS not-accessible
  STATUS     current
  DESCRIPTION
    "Specifies number of instance of call."
  ::= { tsCallEntry 1 }

callConnID OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
     "Specifies ID assigned to call by TServer."
  ::= { tsCallEntry 2 }

callState OBJECT-TYPE
  SYNTAX INTEGER
  {
    stateOk(0),
    stateTransferred(1),
    stateConferenced(2),
    stateGeneralError(3),
    stateSystemError(4),
    stateRemoteRelease(5),
    stateBusy(6),
    stateNoAnswer(7),
    stateSitDetected(8),
    stateAnsweringMachineDetected(9),
    stateAllTrunksBusy(10),
    stateSitInvalidnum(11),
    stateSitVacant(12),
    stateSitIntercept(13),
    stateSitUnknown(14),
    stateSitNocircuit(15),
    stateSitReorder(16),
    stateFaxDetected(17),
    stateQueueFull(18),
    stateCleared(19),
    stateOverflowed(20),
    stateAbandoned(21),
    stateRedirected(22),
    stateForwarded(23),
    stateConsult(24),
    statePickedup(25),
    stateDropped(26),
    stateDroppednoanswer(27),
    stateUnknown(28)
}

  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies call state."
  ::= { tsCallEntry 3 }

callCallID OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies ID assigned to call by switch."
  ::= { tsCallEntry 4 }

callType OBJECT-TYPE
  SYNTAX INTEGER
  {
    typeUnknown(0),
    typeInternal(1),
    typeInbound(2),
    typeOutbound(3),
    typeConsult(4)
   }
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies call type."
  ::= { tsCallEntry 5 }

callReferenceID OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies call reference ID."
  ::= { tsCallEntry 6 }

callTimeStamp OBJECT-TYPE
  SYNTAX      TimeTicks
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies the timestamp when the call was created, in seconds starting from January
1, 1970."
  ::= { tsCallEntry 7 }

callDNIS OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies DNIS field in call structure."
  ::= { tsCallEntry 8 }

callANI OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies ANI field in call structure."
  ::= { tsCallEntry 9 }

callNumParties OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies number of parties currently participating in call."
  ::= { tsCallEntry 10 }

callPartiesList OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies list of parties involved in call."
  ::= { tsCallEntry 11 }

callCustomerID OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies customer id field in call structure."
  ::= { tsCallEntry 12 }

callFirstTransferLocation OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies location of T-Server which first transfer has been made from."
  ::= { tsCallEntry 13 }

callFirstTransferDN OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies DN on remote T-Server which first transfer has been made from."
  ::= { tsCallEntry 14 }

callLastTransferLocation OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies location of T-Server which last transfer has been made from."
  ::= { tsCallEntry 15 }

callLastTransferDN OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies DN on remote T-Server which last transfer has been made from."
  ::= { tsCallEntry 16 }


-- ##############################################################################
-- T-Server Dta Table
-- ##############################################################################

tsDtaTable OBJECT-TYPE
  SYNTAX SEQUENCE OF TsDtaEntry
  MAX-ACCESS not-accessible
  STATUScurrent
  DESCRIPTION
    "A collection of telephony calls data for one or more
     T-Server (for which an entry has been created in the
     tsControlTable). "
  ::= { tServer 3 }

tsDtaEntry OBJECT-TYPE
  SYNTAX     TsDtaEntry
  MAX-ACCESS not-accessible
  STATUS     current
  DESCRIPTION
    "A conceptual row in the tsDtaTable.
     The gsCtrlServerID identifies the manager on whose behalf this entry was created.
     The tsDtaInstanceID value in the index identifies the call instance
     assigned by the management library to this Dta instance.
     An example of the indexing of this entry is
    tsDtaDigits.1.134, where 1 is the index to to control table and 134 is the instance
of dta."
  INDEX {gsCtrlServerID, tsDtaInstanceID }
  ::= { tsDtaTable 1 }

TsDtaEntry ::= SEQUENCE
{
  tsDtaInstanceID Unsigned32,
  tsDtaDigits     DisplayString,
  tsDtaMode       INTEGER,
  tsDtaState      INTEGER,
  tsDtaType       DisplayString
}

tsDtaInstanceID OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "Specifies instance field in dta structure."
  ::= { tsDtaEntry 1 }

tsDtaDigits OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies digits field dta structure."
  ::= { tsDtaEntry 2 }

tsDtaMode OBJECT-TYPE
  SYNTAX INTEGER
  {
    modeShared(0),
    modePrivate(1),
    modeMonitor(2),
    modeUnknown(3)
  }
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies mode field in dta structure."
  ::= { tsDtaEntry 3 }

tsDtaState OBJECT-TYPE
  SYNTAX INTEGER
  {
    stateIdle(0),
    stateNonIdle(1)
  }
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies state field in dta structure."
  ::= { tsDtaEntry 4 }

tsDtaType OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies type field in dta structure."
  ::= { tsDtaEntry 5 }


-- ############################################
-- T-Server Link Table
-- ############################################

tsLinkTable OBJECT-TYPE
  SYNTAX SEQUENCE OF TsLinkEntry
  MAX-ACCESS not-accessible
  STATUScurrent
  DESCRIPTION
    " A collection of CTI link data."
 ::= { tServer 4 }

tsLinkEntry OBJECT-TYPE
  SYNTAX TsLinkEntry
  MAX-ACCESS not-accessible
  STATUS    current
  DESCRIPTION
    "A conceptual row in the tsLink table.
     The gsCtrlServerID identifies the manager on whose behalf this entry was created.
     The tsLinkID value in the index identifies the call instance
     assigned by the management library.
     An example of the indexing of this entry is
     tsLinkProtocol.9.2, where 9 is the index to the control table and 2 is the link
id."
  INDEX {gsCtrlServerID, tsLinkID}
  ::= { tsLinkTable 1 }

TsLinkEntry ::= SEQUENCE
{
  tsLinkID             Unsigned32,
  tsLinkName           DisplayString,
  tsLinkStatus         INTEGER,
  tsLinkProtocol       INTEGER,
  tsLinkSocket         Unsigned32,
  tsLinkPID            Unsigned32,
  tsLinkDelay          Unsigned32,
  tsLinkPort           Unsigned32,
  tsLinkAddress         DisplayString,
  tsLinkX25LocalAddress DisplayString,
  tsLinkMode            Unsigned32,
  tsLinkX25Device       DisplayString,
  tsLinkDTEClass        DisplayString,
  tsLinkTemplate        DisplayString
}

tsLinkID OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "Specifies link identificator."
  ::= { tsLinkEntry 1 }

tsLinkName OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies link name."
  ::= { tsLinkEntry 2 }

tsLinkStatus OBJECT-TYPE
  SYNTAX INTEGER
  {
    statusNone(0),
    statusIdle(1),
    statusUp(2),
    statusDown(3),
    statusUnknown(4)
  }
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies link current status."
  ::= { tsLinkEntry 3 }

tsLinkProtocol OBJECT-TYPE
  SYNTAX INTEGER
  {
    protoX25(0),
    protoTCPIP(1),
    protoX25server(2),
    protoTCPIPdebug(3),
    protoUnknown(4)
  }
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
```

```
    "Specifies protocol type."
  ::= { tsLinkEntry 4 }

tsLinkSocket OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
  "Specifies socket number through which server connected to link."
  ::= { tsLinkEntry 5 }

tsLinkPID OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies link process ID."
  ::= { tsLinkEntry 6 }

tsLinkDelay OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies delay."
  ::= { tsLinkEntry 7 }

tsLinkPort OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies link physical port number."
  ::= { tsLinkEntry 8 }

tsLinkAddress OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies link address."
  ::= { tsLinkEntry 9 }

tsLinkX25LocalAddress OBJECT-TYPE
  SYNTAX     DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies local address for x25 connection."
  ::= { tsLinkEntry 10 }
```

```
tsLinkMode OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies mode."
  ::= { tsLinkEntry 11 }

tsLinkX25Device OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies x25 device."
  ::= { tsLinkEntry 12 }

tsLinkDTEClass OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies DTE class for x25 connection."
  ::= { tsLinkEntry 13 }

tsLinkTemplate OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specifies template."
    ::= { tsLinkEntry 14 }


-- #############################################
-- T-Server Call Filter Table
-- #############################################

tsCallFilterTable OBJECT-TYPE
  SYNTAX SEQUENCE OF TsCallFilterEntry
  MAX-ACCESS not-accessible
  STATUS     current
  DESCRIPTION
    " A collection of CallFilter data."
 ::= { tServer 5 }

tsCallFilterEntry OBJECT-TYPE
  SYNTAX TsCallFilterEntry
  MAX-ACCESS not-accessible
  STATUS     current
  DESCRIPTION
    "A conceptual row in the tsCallFilter table.
```

```
     tsCallFilterTable and tsCallInfoTable introduced to facilitate the discovery of
stuck calls.

     gsCtrlServerID identifies a managed TServer, and along with TableID (9),
     identifies a row in the gServerControlTable on which behalf this entry was
     created.

     An example of the indexing of this entry is
     fltCallUpdatedBefore.101, where 101 is the server ID.

     This table is used to set filtering criteria for the tsCallInfoTabe and provides
     the interface for clearing call by Connection ID."
  INDEX {gsCtrlServerID }
  ::= { tsCallFilterTable 1 }

TsCallFilterEntry ::= SEQUENCE
{
  fltCallCreatedBefore  Unsigned32,
  fltCallCreatedAfter   Unsigned32,
  fltCallUpdatedBefore  Unsigned32,
  fltCallUpdatedAfter   Unsigned32,
  fltClearCallByConnId  DisplayString
}

fltCallCreatedBefore OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Filters calls reported via tsCallInfoTable to those created more than
     specified number of seconds before the request to get information.
     Zero value means filter is not used."
  ::= { tsCallFilterEntry 1 }

fltCallCreatedAfter OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Filters reported calls to those created less than specified number of
     seconds before the request."
  ::= { tsCallFilterEntry 2 }

fltCallUpdatedBefore OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Filters calls reported to those updated more than specified number of
     seconds before the request to get information."
  ::= { tsCallFilterEntry 3 }

fltCallUpdatedAfter OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Filters calls reported to those updated less than specified number of
     seconds before the request to get information."
  ::= { tsCallFilterEntry 4 }

fltClearCallByConnId OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "Connection ID (converted to string by connid_to_str function) of the call
     to be cleared."
  ::= { tsCallFilterEntry 5 }

-- #############################################
-- T-Server Call Info Table
-- #############################################

tsCallInfoTable OBJECT-TYPE
  SYNTAX SEQUENCE OF TsCallInfoEntry
  MAX-ACCESS not-accessible
  STATUS     current
  DESCRIPTION
    " A collection of CallInfo data."
 ::= { tServer 6 }

tsCallInfoEntry OBJECT-TYPE
  SYNTAX     TsCallInfoEntry
  MAX-ACCESS not-accessible
  STATUS     current
  DESCRIPTION
    "A conceptual row in the tsCallFilter table.

     gsCtrlServerID identifies a managed TServer, and along with the TableID of this
     table (10), identifies a row in the gServerControlTable on which behalf this entry
was
     created.

     The tsCallFilterID value in the index identifies the call instance
     assigned by the management library.

     An example of the indexing of this entry is
     callLastUpdatedTimemark.101.5510, where 101 is the server ID and 5510 is the call
instance
     ID as assigned by the management library.
     This table stores the latest snapshot of active calls from a given T-Server.
     Contains a set of attributes that facilitates the discovery of stuck calls."
  INDEX { gsCtrlServerID, callInfoInstanceID }
  ::= { tsCallInfoTable 1 }

TsCallInfoEntry ::= SEQUENCE
{
  callInfoInstanceID            Unsigned32,
  callInfoConnID                DisplayString,
  callInfoType                  INTEGER,
  callInfoCreationTimestamp     Unsigned32,
  callInfoLastUpdatedTimestamp  Unsigned32,
  callInfoInternalParties       DisplayString
}

callInfoInstanceID OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "Call instance ID."
  ::= { tsCallInfoEntry 1 }

callInfoConnID OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
     "Specifies ID assigned to call by TServer."
  ::= { tsCallInfoEntry 2 }

callInfoType OBJECT-TYPE
  SYNTAX INTEGER
  {
    typeUnknown(0),
    typeInternal(1),
    typeInbound(2),
    typeOutbound(3),
    typeConsult(4)
  }
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Specified call type."
  ::= { tsCallInfoEntry 3 }

callInfoCreationTimestamp OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Call creation timestamp - seconds since Epoch as returned by time()
     system function.
     The epoch on most UNIX and POSIX systems is 1970-01-01 00:00:00 UTC."
  ::= { tsCallInfoEntry 4 }

callInfoLastUpdatedTimestamp OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Call info last updated timestamp - seconds since Epoch as returned by
     time() system function.
     The epoch on most UNIX and POSIX systems is 1970-01-01 00:00:00 UTC."
  ::= { tsCallInfoEntry 5 }

callInfoInternalParties OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Internal DN where the call is believed to be located; in case of multiple
     internal parties, all of them are merged into one string, separated by the
     '\n' (new line) character."
  ::= { tsCallInfoEntry 6 }

-- #############################################
-- T-Server Link Stats Table
-- #############################################

tsLinkStatsTable OBJECT-TYPE
  SYNTAX SEQUENCE OF TsLinkStatsEntry
  MAX-ACCESS not-accessible
  STATUS     current
  DESCRIPTION
    " Info about current statistics on TServer link"
  ::= { tServer 7 }

tsLinkStatsEntry OBJECT-TYPE
  SYNTAX     TsLinkStatsEntry
  MAX-ACCESS not-accessible
  STATUS     current
  DESCRIPTION
"A conceptual row in the tsLinkStatsTable.

gsCtrlServerID identifies a managed TServer, and along with the TableID of this table
(11), identifies a row in the gServerControlTable on which behalf this entry was
created.

An example of the indexing of this entry is numberOfRequests.101, where 101 is the
server ID.
```

```
Table contains link send/receive messages statistics."

  INDEX { gsCtrlServerID }
  ::= { tsLinkStatsTable 1 }

TsLinkStatsEntry ::= SEQUENCE
{
linkId                Unsigned32,
timeElapsedSec        Unsigned32,
numberMessagesTx      Unsigned32,
rateMessagesTx        Unsigned32,
numberMessagesRx      Unsigned32,
rateMessagesRx        Unsigned32
}

linkId OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    " The id of the link "
  ::= { tsLinkStatsEntry 1 }

timeElapsedSec OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    " The time (seconds) that have elapsed since the last statistics read."
  ::= { tsLinkStatsEntry 2 }

numberMessagesTx OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    " The number of CTI messages sent by T-Server over this link since last statistics
read."
  ::= { tsLinkStatsEntry 3 }

rateMessagesTx OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    " Literally cti-msg-tx / time-elapsed-sec."
  ::= { tsLinkStatsEntry 4 }

numberMessagesRx OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    " The number of CTI messages received since the last statistics read."
  ::= { tsLinkStatsEntry 5 }

rateMessagesRx OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    " Literally cti-msg-rx / time-elapsed-sec."
  ::= { tsLinkStatsEntry 6 }

-- ################################################################################
-- Hosts
-- ################################################################################

hostsTable OBJECT-TYPE
  SYNTAX SEQUENCE OF HostsEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    " "
  ::= { hosts 1 }

hostsEntry OBJECT-TYPE
  SYNTAX      HostsEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    " "
  INDEX { hostId }
  ::= { hostsTable 1 }

HostsEntry ::= SEQUENCE
{
  hostId             HostDBID,
  hostName           DisplayString,
  hostStatus         DisplayString,
  hostIPAddress      IpAddress,
  hostOSType         DisplayString,
  hostLCAPort        Unsigned32
}

hostId OBJECT-TYPE
  SYNTAX      HostDBID
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    " "
  ::= { hostsEntry 1 }

hostName OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    " "
  ::= { hostsEntry 2 }


hostStatus OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    " "
  ::= { hostsEntry 3 }

hostIPAddress OBJECT-TYPE
  SYNTAX      IpAddress
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    " "
  ::= { hostsEntry 4 }

hostOSType OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    " "
  ::= { hostsEntry 5 }

hostLCAPort OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    " "
  ::= { hostsEntry 6 }



-- ##########################################################################
-- Solutions
-- ##########################################################################

solutionsTable OBJECT-TYPE
  SYNTAX SEQUENCE OF SolutionsEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    " "
  ::= { solutions 1 }

solutionsEntry OBJECT-TYPE
  SYNTAX      SolutionsEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    " "
  INDEX { solutionId }
  ::= { solutionsTable 1 }

SolutionsEntry ::= SEQUENCE
{
  solutionId              SolutionDBID,
  solutionName            DisplayString,
  solutionType            DisplayString,
  solutionStatus          DisplayString,
  solutionControlServer   DisplayString
}

solutionId OBJECT-TYPE
  SYNTAX      SolutionDBID
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    " "
  ::= { solutionsEntry 1 }

solutionName OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    " "
  ::= { solutionsEntry 2 }

solutionType OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    " "
  ::= { solutionsEntry 3 }

solutionStatus OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    " "
  ::= { solutionsEntry 4 }

solutionControlServer OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Name of the Solution Control Server application that controls this solution."
  ::= { solutionsEntry 5 }

--
-- Solution components
--

solutionsComponentsTable OBJECT-TYPE
  SYNTAX SEQUENCE OF SolutionsComponentsEntry
  MAX-ACCESS not-accessible
  STATUS current
  DESCRIPTION
    " "
  ::= { solutions 2 }

solutionsComponentsEntry OBJECT-TYPE
  SYNTAX      SolutionsComponentsEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    " "
  INDEX { solutionId, componentId }
  ::= { solutionsComponentsTable 1 }

SolutionsComponentsEntry ::= SEQUENCE
{
  componentId             ServerDBID,
  componentName           DisplayString
}

componentId OBJECT-TYPE
  SYNTAX      ServerDBID
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "Uniquely identifies a server that is part of this particular Solution."
  ::= { solutionsComponentsEntry 1 }

componentName OBJECT-TYPE
  SYNTAX      DisplayString
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Name of this components."
  ::= { solutionsComponentsEntry 2 }


--
######################################################################################
-- Conformance Information
--
######################################################################################

genesysmibConformance  OBJECT IDENTIFIER ::= { genesys 110 }
genesysmibGroups       OBJECT IDENTIFIER ::= { genesysmibConformance 1 }
genesysmibCompliances  OBJECT IDENTIFIER ::= { genesysmibConformance 2 }

-- Compliance Statements

genesysmibCompliance MODULE-COMPLIANCE
  STATUS current
  DESCRIPTION
    "The compliance statement for SNMPv2 entities."
  MODULE -- this module
  MANDATORY-GROUPS
  {
    gServersListGroup,
    genericServerControlGroup,
    genericServerInfoGroup,
    genericServerLogGroup,
    genericServerPollingGroup,
    genericServerClientGroup,
    tServerInfoGroup,
    tServerCallGroup,
    tServerDtaGroup,
    tServerLinkGroup,
    genericAlarmObjectGroup,
    specificAlarmObjectGroup,
    hostsGroup,
    solutionsGroup,
    solutionsComponentsGroup,
    notificationGroup,
    tServerCallFilterGroup,
    tServerCallInfoGroup
  }
  ::= { genesysmibCompliances 1 }

-- Units of Conformance

gServersListGroup OBJECT-GROUP
  OBJECTS
  {
    gServerName,
    gServerStatus,
    gServerType,
    gServerVersion,
    gServerWorkDir,
    gServerCommandLine,
    gServerPID, gServerCommand,
    gServerDeleteClient
  }
  STATUS current
  DESCRIPTION
    "A collection of objects providing information about Genesys managed servers."
  ::= { genesysmibGroups 1 }

genericServerControlGroup OBJECT-GROUP
  OBJECTS
  {
    gsCleanupTimeout,
    gsCtrlRefreshStatus,
    gsCtrlLastRefreshed,
    gsCtrlAutomaticRefresh,
    gsCtrlRowStatus
  }
  STATUS current
  DESCRIPTION
    " "
  ::= { genesysmibGroups 2 }

genericServerInfoGroup OBJECT-GROUP
  OBJECTS
  {
    gsClientsExistNum,
    gsClientsTotalNum,
    gsServerConfigFile
  }
  STATUS current
  DESCRIPTION
    " "
  ::= { genesysmibGroups 3 }

genericServerLogGroup OBJECT-GROUP
  OBJECTS
  {
    logVerbose,
    logTrace,
    logStandard,
    logDebug,
    logAll,
    logBuffering,
    logSegment,
    logExpire,
    logMessageFile,
    logMessageFormat,
    logTimeFormat,
    logTimeConvert
  }
  STATUS current
  DESCRIPTION
    " "
  ::= { genesysmibGroups 4 }


genericServerPollingGroup OBJECT-GROUP
  OBJECTS
  {
    gsPollingStatus,
    gsPollingInterval,
    gsPollingID,
    gsPollingLastTrap
  }
  STATUS current
  DESCRIPTION
    " "
  ::= { genesysmibGroups 5 }

genericServerClientGroup OBJECT-GROUP
  OBJECTS
  {
    gsClientAppName,
    gsClientAuthorized,
    gsClientType,
    gsClientGotEvents,
    gsClientSentReqs
  }
  STATUS current
  DESCRIPTION
    " "
  ::= { genesysmibGroups 6 }


tServerInfoGroup OBJECT-GROUP
  OBJECTS
  {
    tsCallsExistNum,
    tsCallsTotalNum,
    tsLinksCommand,
    tsLastChangedLinkStatus
  }
  STATUS current
  DESCRIPTION
    " "
  ::= { genesysmibGroups 7 }
```

```
tServerCallGroup OBJECT-GROUP
  OBJECTS
  {
    callConnID,
    callState,
    callCallID,
    callType,
    callReferenceID,
    callTimeStamp,
    callDNIS,
    callANI,
    callNumParties,
    callPartiesList,
    callCustomerID,
    callFirstTransferLocation,
    callFirstTransferDN,
    callLastTransferLocation,
    callLastTransferDN
  }
  STATUS current
  DESCRIPTION
    " "
  ::= { genesysmibGroups 8 }

tServerDtaGroup OBJECT-GROUP
  OBJECTS
  {
    tsDtaDigits,
    tsDtaMode,
    tsDtaState,
    tsDtaType
  }
  STATUS current
  DESCRIPTION
    " "
  ::= { genesysmibGroups 9 }

tServerLinkGroup OBJECT-GROUP
  OBJECTS
  {
    tsLinkName,
    tsLinkStatus,
    tsLinkProtocol,
    tsLinkSocket,
    tsLinkPID,
    tsLinkDelay,
    tsLinkPort,
    tsLinkAddress,
    tsLinkX25LocalAddress,
    tsLinkMode,
    tsLinkX25Device,
    tsLinkDTEClass,
    tsLinkTemplate
  }
  STATUS current
  DESCRIPTION
    " "
  ::= { genesysmibGroups 10 }


genericAlarmObjectGroup OBJECT-GROUP
  OBJECTS
  {
    gsAlarmId,
    gsAlarmLogText,
    gsAlarmMessagesIds,
    gsAlarmApplicationName,
    gsAlarmApplicationType,
    gsAlarmCategory,
    gsServersLastAlarm,
    gsServersLastTrap,
    gsAlarmAppHostName,
    gsAlarmGUID
  }
  STATUS current
  DESCRIPTION
    " "
  ::= { genesysmibGroups 11 }

specificAlarmObjectGroup OBJECT-GROUP
  OBJECTS
  {
    tsLastChangedLinkStatus
  }
  STATUS current
  DESCRIPTION
    " "
  ::= { genesysmibGroups 12 }

hostsGroup OBJECT-GROUP
  OBJECTS
  {
    hostName,
    hostStatus,
    hostIPAddress,
    hostOSType,
    hostLCAPort
  }
  STATUS current
  DESCRIPTION
    " "
  ::= { genesysmibGroups 13 }

solutionsGroup OBJECT-GROUP
  OBJECTS
  {
    solutionName,
    solutionType,
    solutionStatus,
    solutionControlServer
  }
  STATUS current
  DESCRIPTION
    " "
  ::= { genesysmibGroups 14 }

solutionsComponentsGroup OBJECT-GROUP
  OBJECTS
  {
    componentName
  }
  STATUS current
  DESCRIPTION
    " "
  ::= { genesysmibGroups 15 }

notificationGroup NOTIFICATION-GROUP
  NOTIFICATIONS
  {
    gsAlarm,
    gsMLAlarm,
    gsServerUpTrap,
    gsServerDownTrap,
    gsPollingSignal,
    tsLinkStatusTrap
  }
  STATUS current
  DESCRIPTION
    "List of notifications generated by ML SNMP agent."
  ::= { genesysmibGroups  16 }

tServerCallFilterGroup OBJECT-GROUP
  OBJECTS
  {
    fltCallCreatedBefore,
    fltCallCreatedAfter,
    fltCallUpdatedBefore,
    fltCallUpdatedAfter,
    fltClearCallByConnId
  }
  STATUS current
  DESCRIPTION
```

```
    " "
  ::= { genesysmibGroups 17 }

tServerCallInfoGroup OBJECT-GROUP
  OBJECTS
  {
    callInfoConnID,
    callInfoType,
    callInfoCreationTimestamp,
    callInfoLastUpdatedTimestamp,
    callInfoInternalParties
  }
  STATUS current
  DESCRIPTION
    " "
  ::= { genesysmibGroups 18 }

END
```

Genesys

Appendix

# E Changes in MIB Files

This appendix lists the differences in Genesys Management Information Base (MIB) files between releases. It contains the following sections:

## Changes from 8.0 to 8.1

There are no changes in the Genesys MIB file in release 8.1.

## Changes from 7.6 to 8.0

In release 8.0, the Genesys MIB file is modified as follows:

- `apps OBJECT IDENTIFIER ::= { genesys 200 }` is added to provide an extension point for applications. This enables an application, such as Genesys Voice Platform, to register additional MIB regions that are controlled by the application itself (not by Solution Control Server).
- The `gServerStatus` object is extended with the following new status levels:
    - `statusSuspending(7)`
    - `statusSuspended(8)`
- The table `tsLinkStatsTable` is added to provide current statistics about the T-Server Link. See "The tsLinkStatsTable Table" on page 110.

# Changes from 7.5 to 7.6

There are no changes in the Genesys MIB file in release 7.6.

# Changes from 7.1 to 7.5

There are no changes in the Genesys MIB file in release 7.5.

# Changes from 7.0 to 7.1

Release 7.1 supports the new stuck calls functionality with new tables and an extended SNMP trap.

Two new MIB tables were added to manage stuck calls via SNMP: `tsCallFilterTable` and `tsCallInfoTable`.

- `tsCallFilterTable` filters calls to reduce network traffic and increase application performance when monitoring T-Servers via SNMP to identify stuck calls.

  `tsCallFilterTable` also provides the interface for clearing a call by the call's Connection ID. See Chapter 11, "The tsCallFilterTable" on page 108.
- `tsCallInfoTable supports this function` with information about active calls filtered by conditions set in `tsCallFilterTable`. See Chapter 11, "The tsCallInfoTable Table" on page 109.

**Note:** The new filters do not apply in any way to the old `tsCallTable`.

The SNMP trap gsMLAlarm was extended.

- Starting with 7.1.0, this trap includes the host name and the Alarm GUID in the trap's variable-binding list. These are new attributes added to the set of existing attributes.
- You can use the Alarm GUID to match alarm clearance traps to alarm creation traps.

Appendix

# F Using Startup Files

Use the procedures in this section if your application requires separate start and stop scripts. Refer to “How to Use Startup Files” on page 56 for more information about these files.

## Procedure: Genesys Administrator—Modifying an application to use a startup file

**Purpose:** To enable an application to use a startup file.

### Prerequisites

- Configuration Layer components are installed and running.
- The `Application` object exists for the application that is use a startup file.
- You are logged in to Genesys Administrator.

### Start of procedure

1. Open the `Configuration` tab of the `Application` object.
2. In the `Server Info` section, modify the following properties:
   - `Command Line` property—Instead of the application executable file or startup file name, specify the command prompt name (`cmd.exe`) with the full path to it. For example:

     `D:\Windows\system32\cmd.exe`
   - `Command Line Arguments` property—Instead of the Configuration Server parameters and application name, specify the startup file name (`startServer.bat`) with the full path to it, preceded with the `/c` command line parameter. For example:

     `/c D:\GCTI\MessageServer\startServer.bat`

3. Click `Save` to save the configuration changes.

### End of procedure

### Next Steps

- Modify the startup file if required. Use the procedure "Modifying a startup file" on page 247.

## Procedure: Configuration Manager—Modifying an application to use a startup file

**Purpose:** To enable an application to use a startup file.

### Prerequisites

- Configuration Layer components are installed and running.
- The `Application` object exists for the application that is use a startup file.
- You are logged in to Configuration Manager.

### Start of procedure

1. Open the `Properties` dialog box of the `Application` object, and select the `Start Info` tab.
2. Modify the following properties:
   - `Command Line`—Instead of the application executable file or startup file name, specify the command prompt name (`cmd.exe`) with the full path to it. For example:
     `D:\Windows\system32\cmd.exe`
   - `Command Line Arguments`—Instead of the Configuration Server parameters and application name, specify the startup file name (`startServer.bat`) with the full path to it, preceded with the `/c` command line parameter. For example:
     `/c D:\GCTI\MessageServer\startServer.bat`
3. Click `OK` to save the configuration changes.

### End of procedure

### Next Steps

- Modify the startup file if required. Use the procedure "Modifying a startup file" on page 247.

## Procedure: Modifying a startup file

**Purpose:** To enable Management Layer to run applications from their startup files.

### Prerequisites

- The applications must be configured to use startup files. Use one of the procedures "Genesys Administrator—Modifying an application to use a startup file" on page 245 or "Configuration Manager—Modifying an application to use a startup file" on page 246, as appropriate.

### Start of procedure

1. Using the text-editor of your choice, open one of the following files that are located in the applications' directories:
   - on UNIX—`run.sh`
   - on Windows—`startServer.bat`
2. If necessary, modify the existing line, or insert a new line specifying the application's executable file name with the full path to it, followed by this sequence of symbols:

   ```
   %1 %2 %3 %4 %5 %6
   ```

   For example, for a Message Server application installed to a `GCTI` directory on the `D` drive, the content of a startup file to use with the Management Layer looks like this:

   ```
   D:\GCTI\MessageServer\MessageServer.exe %1 %2 %3 %4 %5 %6
   ```
3. Save the file.

### End of procedure

# Related Documentation Resources

The following resources provide additional information that is relevant to this software. Consult these additional resources as necessary.

## Framework

- *Framework 8.0 Architecture Help,* which will help you view the place of the Management Layer in the Framework architecture.
- *Framework 8.1 Deployment Guide,* which will help you configure, install, start, and stop the Management Layer components.
- *Framework 8.1 Configuration Options Reference Manual,* which will provide you with descriptions of configuration options for the Management Layer components.
- *Framework 8.1 Genesys Administrator Help,* which will help you use Genesys Administrator.
- *Framework 8.1 Configuration Manager Help,* which will help you use Configuration Manager.
- *Framework 8.0 Solution Control Interface Help,* which will help you use Solution Control Interface.
- Release Notes and Product Advisories for this product, which are available on the Genesys Technical Support website at `http://genesyslab.com/support`.

## Genesys

- *Genesys 8.1 Security Deployment Guide*, which describes the security features available in Genesys software, and provides detailed instructions for deploying these features.

- *Genesys Technical Publications Glossary,* which ships on the Genesys Documentation Library DVD and which provides a comprehensive list of the Genesys and computer-telephony integration (CTI) terminology and acronyms used in this document.
- *Genesys Migration Guide*, which ships on the Genesys Documentation Library DVD, and which provides documented migration strategies for Genesys product releases. Contact Genesys Technical Support for more information.

Information about supported hardware and third-party software is available on the Genesys Technical Support website in the following documents:

- *Genesys Supported Operating Environment Reference Guide*
- *Genesys Supported Media Interfaces Reference Manual*

Consult the following additional resources as necessary:

- *Genesys Hardware Sizing Guide,* which provides information about Genesys hardware sizing guidelines for the Genesys 8.x releases.
- *Genesys Interoperability Guide,* which provides information on the compatibility of Genesys products with various Configuration Layer Environments; Interoperability of Reporting Templates and Solutions; and Gplus Adapters Interoperability.
- *Genesys Licensing Guide,* which introduces you to the concepts, terminology, and procedures that are relevant to the Genesys licensing system.
- *Genesys Database Sizing Estimator 8.x Worksheets,* which provides a range of expected database sizes for various Genesys products.

For additional system-wide planning tools and information, see the release-specific listings of System Level Documents on the Genesys Technical Support website, accessible from the `system level documents by release` tab in the Knowledge Base `Browse Documents` Section.

Genesys product documentation is available on the:

- Genesys Technical Support website at `http://genesyslab.com/support`.
- Genesys Documentation Library DVD, which you can order by e-mail from Genesys Order Management at `orderman@genesyslab.com`.

# Document Conventions

This document uses certain stylistic and typographical conventions—introduced here—that serve as shorthands for particular kinds of information.

## Document Version Number

A version number appears at the bottom of the inside front cover of this document. Version numbers change as new information is added to this document. Here is a sample version number:

81fr_us-ml_03-2011_v8.1.001.00

You will need this number when you are talking with Genesys Technical Support about this product.

## Screen Captures Used in This Document

Screen captures from the product graphical user interface (GUI), as used in this document, may sometimes contain minor spelling, capitalization, or grammatical errors. The text accompanying and explaining the screen captures corrects such errors *except* when such a correction would prevent you from installing, configuring, or successfully using the product. For example, if the name of an option contains a usage error, the name would be presented exactly as it appears in the product GUI; the error would not be corrected in any accompanying text.

## Type Styles

Table 41 describes and illustrates the type conventions that are used in this document.

### Table 41: Type Styles

| Type Style | Used For | Examples |
|---|---|---|
| Italic | • Document titles<br>• Emphasis<br>• Definitions of (or first references to) unfamiliar terms<br>• Mathematical variables<br>Also used to indicate placeholder text within code samples or commands, in the special case where angle brackets are a required part of the syntax (see the note about angle brackets on page 252). | Please consult the *Genesys Migration Guide* for more information.<br>Do *not* use this value for this option.<br>A *customary and usual* practice is one that is widely accepted and used within a particular industry or profession.<br>The formula, $x + 1 = 7$ where $x$ stands for ... |
| Monospace font<br>(Looks like `teletype` or `typewriter text`) | All programming identifiers and GUI elements. This convention includes:<br>• The *names* of directories, files, folders, configuration objects, paths, scripts, dialog boxes, options, fields, text and list boxes, operational modes, all buttons (including radio buttons), check boxes, commands, tabs, CTI events, and error messages.<br>• The values of options.<br>• Logical arguments and command syntax.<br>• Code samples.<br>Also used for any text that users must manually enter during a configuration or installation procedure, or on a command line. | Select the `Show variables on screen` check box.<br>In the `Operand` text box, enter your formula.<br>Click `OK` to exit the `Properties` dialog box.<br>T-Server distributes the error messages in `EventError` events.<br>If you select `true` for the `inbound-bsns-calls` option, all established inbound calls on a local agent are considered business calls.<br>Enter `exit` on the command line. |
| Square brackets ([ ]) | A particular parameter or value that is optional within a logical argument, a command, or some programming syntax. That is, the presence of the parameter or value is not required to resolve the argument, command, or block of code. The user decides whether to include this optional information. | `smcp_server -host [/flags]` |
| Angle brackets (< >) | A placeholder for a value that the user must specify. This might be a DN or a port number specific to your enterprise.<br>**Note:** In some cases, angle brackets are required characters in code syntax (for example, in XML schemas). In these cases, italic text is used for placeholder values. | `smcp_server -host <confighost>` |

# Index