SeeBeyond™ eBusiness Integration Suite

e*Gate Integrator Alert and Log File Reference Guide

Release 4.5.2

Version 20020607152339.

Contents

Chapter 3

Chapter 4

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces you to the *e*Gate Integrator Alert and Log File Reference Guide*, its general purpose and scope, and its organization. It also provides sources of related documentation and information.

1.1 User's Guide Purpose and Scope

This user's guide explains generally how to read and use e*Gate Integrator system Alert notifications and log files. It also provides information on how to troubleshoot the SeeBeyond Technology Corporation's™ (SeeBeyond™) e*Gate system as a whole. This manual includes:

- An overview of log files, e*Gate monitoring, and Alert notifications.
- Using the e*Gate Monitor.
- A detailed explanation of how to use log files, the information they contain, and debug levels and flags, including using log files for troubleshooting.
- A comprehensive explanation of the notification process, implementation of notification strategies, and Alert codes and descriptions (error messages) with a reference list as well as troubleshooting information.
- Frequently asked questions on how to troubleshoot general e*Gate system problems.
- A list of status notification descriptions.
- An explanation on how to customize default notification routing.

Important: *Any operation explanations given here are generic, for reference purposes only, and do not necessarily address the specifics of setting up and/or operating individual e*Gate systems.*

This document does not contain information on software installation and system administration procedures (see **"Supporting Documents" on page 15**).

1.2 Intended Audience

The reader of this guide is presumed to be an experienced PC user responsible for helping to set up and/or to maintain a fully functioning e*Gate system. This person must also have moderate to advanced level knowledge of Windows NT/ Windows 2000 or UNIX operations and be thoroughly familiar with Windows-style GUI operations.

1.3 Document Organization

This document is organized topically as follows:

- **Chapter 1 "Introduction"** gives a general preview of this document, its purpose, scope, and organization.
- **Chapter 2 "Overview of Log Files, e*Gate Monitoring, and Alerts"** provides an overview of these features along with a description of Alert notifications and how to activate logging.
- **Chapter 3 "Monitoring e*Gate"** explains how to use the e*Gate Monitor graphical user interface (GUI) to view Alert notifications and the information they provide.
- **Chapter 4 "Using Log Files"** tells you how to locate, understand, and use the log files, including a detailed explanation of how to set debug levels and flags to regulate the amount and type of information recorded in the files; and provides examples of using log files to diagnose frequently encountered problems.
- **Chapter 5 "e*Gate Alert and Notification System"** introduces how the e*Gate system handles monitoring and notification Events. It then explains e*Gate notifications, their codes, and Alert/error messages that appear in the e*Gate Monitor, including potential problems and possible solutions. The tables are organized by components and/or e*Gate functions.
- **Chapter 6 "Frequently Asked Questions"** includes helpful hints, tips for shortcuts, and best practices for troubleshooting the e*Gate system.
- This guide also includes **Appendix A**, which provides a table of e*Gate status Event messages; **Appendix B**, which explains customizing default notifications routing; and **Appendix C**, which provides a list of Java error messages.

1.4 Writing Conventions

The writing conventions listed in this section are observed throughout this document.

Hypertext Links

When you are using this guide online, cross-references are also hypertext links and appear in **blue text** as shown below. Click the **blue text** to jump to the section.

For information on these and related topics, see **"Parameter, Function, and Command Names" on page 15**.

Command Line

Text to be typed at the command line is displayed in a special font as shown below.

```
java -jar ValidationBuilder.jar
```

Variables within a command line are set in the same font and bold italic as shown below.

```
stcregutil -rh host-name -rs schema-name -un user-name
-up password -ef output-directory
```

Code and Samples

Computer code and samples (including printouts) on a separate line or lines are set in Courier as shown below.

```
Configuration for BOB_Promotion
```

However, when these elements (or portions of them) or variables representing several possible elements appear within ordinary text, they are set in *italics* as shown below.

path and *file-name* are the path and file name specified as arguments to **-fr** in the **stcregutil** command line.

Notes and Cautions

Points of particular interest or significance to the reader are introduced with ***Note***, ***Caution***, or ***Important***, and the text is displayed in *italics*, for example:

Note: *The Actions menu is only available when a Properties window is displayed.*

User Input

The names of items in the user interface such as icons or buttons that you click or select appear in **bold** as shown below.

Click **Apply** to save, or **OK** to save and close.

File Names and Paths

When names of files are given in the text, they appear in **bold** as shown below.

Use a text editor to open the **ValidationBuilder.properties** file.

When file paths and drive designations are used, with or without the file name, they appear in **bold** as shown below.

In the **Open** field, type **D:\setup\setup.exe** where **D:** is your CD-ROM drive.

Parameter, Function, and Command Names

When names of parameters, functions, and commands are given in the body of the text, they appear in **bold** as follows:

The default parameter **localhost** is normally only used for testing.

The Monk function **iq-put** places an Event into an IQ.

You can use the **stccb** utility to start the Control Broker.

Additional Conventions

Windows Systems: The e*Gate system is fully compliant with both Windows NT and Windows 2000 platforms. When this document refers to Windows, such statements apply to both Windows platforms.

UNIX and Linux Systems: This guide uses the backslash ("\") as the separator within path names. If you are working on a UNIX or Linux system, please make the appropriate substitutions.

Note: *The e*Gate system is fully compatible with Compaq Tru64 V4.0F and V5.0A.*

1.5 Supporting Documents

The following SeeBeyond documents provide additional information about the Alerts and log files as explained in this guide:

- *Batch e*Way Intelligent Adapter User's Guide*
- *e*Gate Integrator Alert Agent User's Guide*
- *e*Gate Integrator Installation Guide*
- *e*Gate Integrator Intelligent Queue Services Reference Guide*
- *e*Gate Integrator SNMP Agent User's Guide*
- *e*Gate Integrator System Administration and Operations Guide*
- *e*Gate Integrator User's Guide*
- *e*Insight Business Integration Suite Primer*
- *e*Insight Business Integration Suite Deployment Guide*
- *Monk Developer's Reference*
- *Standard e*Way Intelligent Adapters User's Guide*

See the Additional Sources of Information chapter in the *e*Insight™ Business Integration Suite Primer* for a complete list of e*Gate-related documentation. You can also refer to the appropriate Microsoft Windows or UNIX documents, if necessary.

1.6 SeeBeyond Web Site

The SeeBeyond Web site is your best source for up-to-the-minute product news and technical support information. The site's URL is:

http://www.seebeyond.com

Chapter 2

Overview of Log Files, e*Gate Monitoring, and Alerts

This chapter gives an overview of e*Gate monitoring, log files, and Alert notifications, including information on how they operate and steps for activating logging and setting Alerts in the e*Gate system.

Note: *For complete information on the e*Gate system, including setup, configuration, and operation, see the* ***e*Gate Integrator User's Guide****.*

2.1 Log Files, e*Gate Monitoring, and Alert Notifications: Introduction

The e*Gate system provides the following basic troubleshooting tools:

- Log files
- e*Gate monitoring and Alert notifications

This section provides a summary of and introduction to each of these e*Gate features.

2.1.1 Log Files

An important source of information about your e*Gate configuration comes from log files. The e*Gate system's logging facility enables you to trace and store detailed operations information. The following components can generate log files:

- e*Way Intelligent Adapters
- Business Object Brokers (BOBs)
- Intelligent Queue (IQ) Managers
- Control Brokers
- e*Insight Business Process Manager modules

In e*Gate, these executable components are also called modules. Each log file is clearly labeled as belonging to its generating module component. These modules have the ability to create log files that contain whatever type and amount of debugging information you select.

2.1.2 Alert Notifications

The e*Gate system continually issues *monitoring Events* to give you information on how well the overall system is functioning. All major e*Gate components and features issue these Events via internal system operations. The Control Broker converts monitoring Events into *notifications (notification Events)* and sends them to e*Gate monitors.

Note: *In e*Gate, an Event is a package of data, an e*Gate Event is a packet of information that is exchanged with external applications, and an Alert event is a notification of a problem or change of status inside of e*Gate.*

Notifications that indicate problems are called *Alert notifications*. A message readout of Alert notifications appears in the e*Gate Monitor GUI. This feature provides you immediate, easy-to-read information on system problems.

The e*Gate Monitor window also displays readouts of monitoring Event/notification status messages with information on the normal functioning of components. However, the Alert notifications, called *Alerts* in the GUI, contain the actual e*Gate error messages.

Alerts provide important information on system problems and where to start your troubleshooting. You can also monitor status and Alert notifications by other means, such as the command line, faxes, or pagers.

Note: *For more information on Alerts and the monitoring/notification system in e*Gate, see* **Chapter 5**. *For more information on the e*Gate Monitor GUI, see* **Chapter 3**.

This chapter explains

2.2 Understanding Log Files

The components of the e*Gate system carry on a constant invisible "conversation" with each other. They describe their internal states, send questions and return responses, and announce both the results of their efforts and the problems they have encountered in performing their various activities.

Log Files — Provide a way to capture this conversation, so you can analyze the internal workings of the e*Gate system and use this information to solve problems.

2.2.1 Debug Levels and Flags

The e*Gate system records a variety of information in log files according to settings you can control. This logging control feature helps you log only the points of the system's internal conversation that interest you.

Note: *As you use more debug flags and more detailed logging levels, the resulting logs will contain more information. Recording greater amounts of log information, however, can cause your system to run more slowly.*

Each system Event that can be logged has the following basic properties:

- A *debug level* that describes the general nature of a system Event, for example, whether the Event is fatal, nonfatal error-related, or only informational.
- One or more *debug flags* that describe the source of a system Event. For example, **CB** Events originate from the Control Broker, **EWY** Events originate from e*Ways, **MNK** Events describe the workings of Monk scripts, and so on. Many components produce both verbose-mode and normal- (nonverbose) mode system Events.

When you set up logging for a component in the Enterprise Manager, you specify the debug level and the flags of the system Events you want to log. The resulting log file will then contain a record of only those system Events whose level and flags match the level and flags you have specified.

2.2.2 Handling Log Files

The e*Gate system provides no special tools to view or archive log files. You can use any methodology appropriate for your installation.

The amount of information written to a log file affects its size, and tracing that information slows system performance. You can reduce the impact of logging on both hard disk space and system memory by changing the debug level and flags. You can perform these operations using the Enterprise Manager (see the next section).

2.2.3 Activating Logging

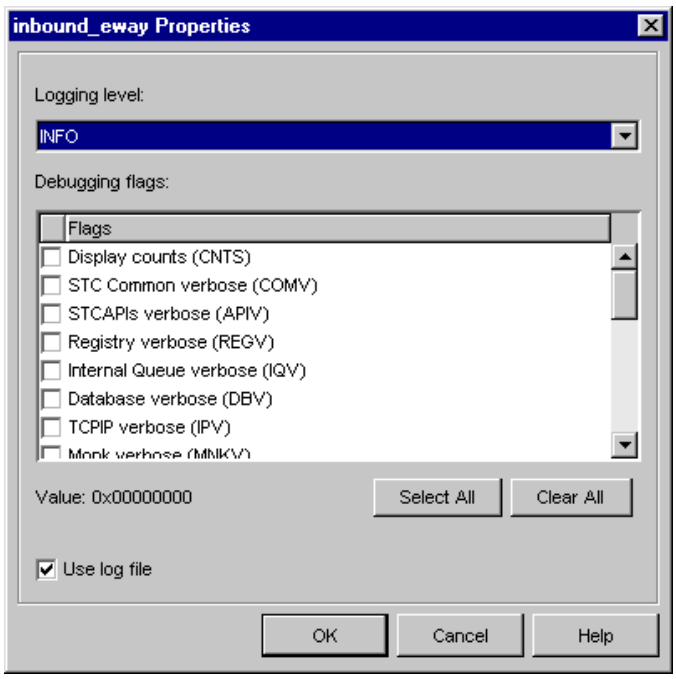
You activate logging and set logging levels using features in the Enterprise Manager as explained in this section.

To configure a module to generate a log file

1. Select the **Components** tab, then use the Navigator to select the module for which you want to generate a log file.
2. On the toolbar, click  .
3. Select the **Advanced** tab, then click **Log**.

   The **Logging** dialog box for the current component appears (see **Figure 1 on page 20**).

Figure 1 Setting logging options



4 Choose the logging options you want to apply. Exactly which flags you choose depends on which topics you want to observe in each module (see **"Understanding Log Files" on page 41** for more information). You have the following additional options:

- Click **Select All** to enable all logging options.
- You must select a logging level as well as debugging flags to see anything more than basic information in the log file. Click **Help** if you need more information about logging options.

5 Click the **Use log file** check box.

6 Click **OK** to save your selected options and close the property sheet.

Note: *When you make changes to the Registry through the Enterprise Manager without modules running, the Registry caches all the changes and sends update messages to all e*Ways after they start up. When the e*Ways receive the messages, they reload all configurations. This operation may take some time. Be sure to allow all modules to run for a few minutes, then restart them.*

See **Chapter 4** for a complete explanation of log files, their information, how to read them, and how to use them for troubleshooting.

2.3 Understanding e*Gate Monitoring and Alert Notifications

The e*Gate monitoring/notification system provides the following methods to check the status of your e*Gate system:

- **Interactive Monitoring** uses client applications to display real-time status information on the e*Gate system and enables you to start and stop e*Gate components. e*Gate includes two interactive monitors: the e*Gate Monitor GUI feature and a command-line monitor, **stccmd**.

Note: *See the* **"Viewing Alerts Using Command-line Monitoring" on page 34** *for information on the* ***stccmd*** *command.*

- **Non-interactive Monitoring** forwards Alert and status information through delivery channels, including e-Mail and printing, but does not provide any means to control e*Gate components. The non-interactive notification system also provides an escalation system for unresolved problems and failures to make sure that all notifications are properly delivered.

Note: *See the* ***e*Gate Integrator User's Guide*** *for information on non-interactive monitoring.*

You can use the e*Gate Integrator Alert and SNMP Agent features for non-interactive monitoring. For details, see the appropriate SeeBeyond user's guides that explain these features.

2.3.1 How Real-time Monitoring Works

In the e*Gate real-time monitoring system, components send monitoring Events to the Control Broker. These monitoring Events include a code and description plus additional information such as the time of Event occurrence and names of possibly affected components.

Role of the Control Broker

The e*Gate monitoring/notification system depends heavily on the Control Broker, both as a source of information and as an intermediary for commands issued to the various e*Gate components. Monitoring e*Gate depends on the following operations:

- You must have the Control Broker running before you can use any e*Gate monitoring or notification feature.
- When the Control Broker starts running, it first binds a port (this action is configurable via a range in the appropriate **Control Broker** property sheet), then the monitor connects to it.

Caution: *Because of the Control Broker's importance within the e*Gate monitoring/ notification system, SeeBeyond recommends that system failures involving this component must be addressed as quickly as possible. Both the host and the Control Broker must be active before the e*Gate Monitor can connect to them.*

The Control Broker uses a Collaboration script to convert monitoring Events into notifications (notification Events), which contain not only the data from the monitoring Event but a range of recipient information, such as e-Mail addresses.

Note: *To monitor e*Gate through a fire wall, use VPN or a similar product.*

Role of notifications

Notifications have the following basic properties you need to know:

- You can configure the Collaboration script for notification-routing to apply recipient information based on monitoring-Event properties. For example, you can notify one set of users regarding fatal errors and others regarding non-fatal errors, or route notifications via e-Mail, based on the component issuing the monitoring Event.
- Notifications go directly from the Control Broker to *monitors*, which are simply applications that display notifications. Non-interactive monitors merely display information or route that information through delivery channels such as e-Mail, while interactive monitors, for example, the e*Gate Monitor, also enable you to send commands to e*Gate components and to mark which notifications have been resolved.
- The Control Broker can also execute command scripts (for example, to launch batch files, shell scripts, or executable files), either in addition to or instead of sending notifications to monitors.

The monitoring/notification system is explained in detail in the *e*Gate Integrator System Administration and Operations Guide*. This explanation includes information on the conversion script that directs notifications to their final recipients.

2.3.2 Status and Alert Notifications

The e*Gate Monitor GUI is your primary source for viewing system notifications in the form of on-screen messages. The two types of notifications displayed by the e*Gate Monitor are:

- **Status** messages that provide routine information of interest on the operation of e*Gate and do not display errors.
- **Alerts** messages that give you error and problem information you can use for troubleshooting purposes.

As notifications (via notification Events) appear in the e*Gate Monitor window, each one contains the following data segments:

- Code
- Description

The Control Broker takes these segments directly from the monitoring Event it uses to generate the corresponding notification (Event). An example of a notification code and description follows:

```
10113020: IQ Manager Down Controlled
```

The previous example comes from an Alert notification, but status notifications have the same format. Alert notifications (called Alerts in the GUI) are the relevant e*Gate error messages you can use to troubleshoot the entire system.

See **Chapter 5** for a complete explanation of notifications, including codes and descriptions (as displayed in the e*Gate Monitor), Alerts, and what Alerts mean to you. This chapter also provides comprehensive troubleshooting information.

2.3.3 Readable Time and Date Stamps

The date and time stamp of a generated notification is in hexadecimal format. This is the code that is used in **Notification.tsc** files. Having a date and time stamp on Alert occurrences in a readable format is very important, especially when you construct information about the Alert in preparation to sending it to others.

To create readable time and date stamps

Use the Monk code below to set up your system to convert the hexadecimal date and time stamp into a human-readable date and time stamp.

1 Load the **stc_monkext.dll** one time (for example):

```
(if (not (defined? ext_loaded))
 (begin
  (load-extension "stc_monkext.dll")
  (define ext_loaded #t)
 )
 (begin
  ;; already loaded extension file
 )
)
```

2 Then:

```
(define alert_dt 0)
(define alert_tm 0)
(define alert_dttm 0)

(set! alert_dt (big-endian->integer (hexdump->string (get ~<YOUR_DATE_NODE>)) 4))
(set! alert_tm (big-endian->integer (hexdump->string (get ~<YOUR_TIME_NODE>)) 4))
(set! alert_dttm (clock-timestamp2string alert_dt alert_tm))
```

Chapter 3

Monitoring e*Gate

This chapter introduces e*Gate monitoring, including using the e*Gate Monitor to view Alert and status messages, along with viewing Alerts using command-line monitoring.

3.1 Overview of e*Gate Monitoring

This chapter provides the following information on e*Gate monitoring:

- Using the e*Gate Monitor GUI to view Alert and status messages interactively
- Marking Alerts after viewing them
- Creating automatic notification resolutions
- Reconnecting to the Control Broker
- Limiting and increasing the number of displayed notifications
- Viewing Alerts using command-line monitoring

This chapter explains

3.2 Setting a Port for the Monitor When Using a Firewall

The port that the Monitor uses when a third party firewall is installed can be set prior to installing e*Gate as it is a user configurable parameter. The range of selectable ports for the Monitor is 4000-4500 (4000, 4001, 4002, ... 4500). However, the Monitor cannot be used until the user has e*Gate running and opens the **Control Broker Properties** dialog box to specifically set the ports for the Monitor.

To set the port:

1. Select the **Components** tab in the Navigator pane.
2. Select the **Control Broker**.

3 Open the **Control Broker <*hostname*_cb> Properties** dialog box.

4 On the **Advanced** tab, click **Port**.

5 On the **Port Properties** dialog box, set the **Lower Port** and the **Upper Port**.

Note: *For more information on the Monitor and Control Brokers, see the* ***e*Gate Integrator User's Guide****.*

3.3 Using the e*Gate Monitor

This section explains how to use the e*Gate Monitor GUI to view Alerts interactively.

To launch the e*Gate Monitor

1 Click **Start**.

2 Choose **Programs**, then point to **SeeBeyond™ eBusiness Integration Suite**.

3 Click **e*Gate Monitor**.

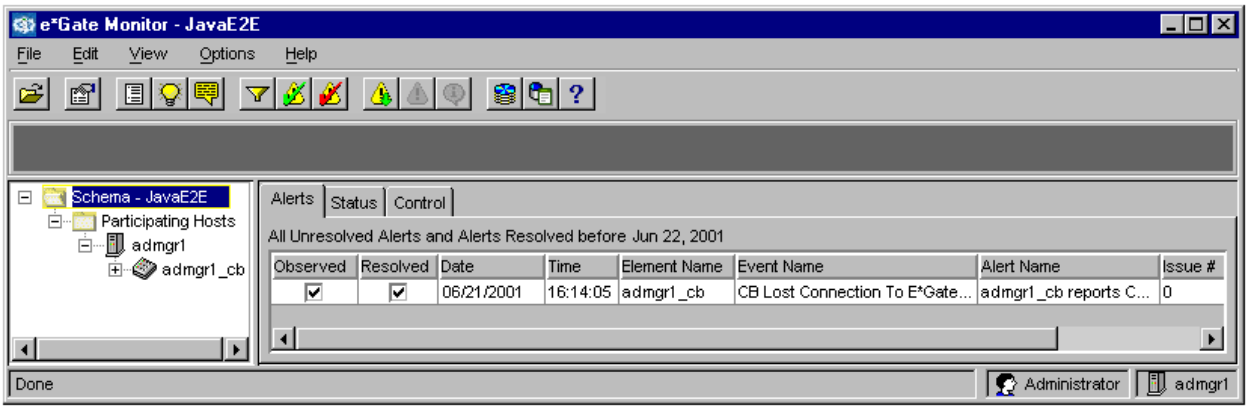
4 When the **e*Gate Monitor Login** dialog box appears, choose a **Server** from the drop-down list, enter a **Login ID**, type your **Password**, then click **Log In**.

5 On the **Open Schema on Registry Host <*host name*>**, select the schema you want to monitor and click **OK**.

Note: *Select* ***only*** *the schema that is currently supporting an* ***active*** *Control Broker. If you open a schema that has an inactive (non-running) Control Broker that has the same name as an active (running) Control Broker, you will get inaccurate and undesirable results.*

Figure 2 illustrates the e*Gate Monitor window.

Figure 2 e*Gate Monitor window



Like the Enterprise Manager window, the e*Gate Monitor window is divided into two panes. The left pane, the Navigator pane, operates exactly like the same pane within the Enterprise Manager. The colors of the components show their status at a glance:

- Normal component symbols, matching the colors that appear in the Enterprise Manager, indicate that a component is functioning normally.
- Red component symbols indicate that a component is either not functioning or not communicating with the Control Broker.
- Gray component symbols indicate that the e*Gate Monitor is not connected to an element's Control Broker.

The Message pane (right) provides the following tabs:

- **Alerts** are display messages describing Events that warrant attention and resolution, such as warnings regarding components that have stopped functioning or system parameters that have exceeded preset limits.
- **Status** displays messages concerning conditions that do not represent problems to be solved, such as news that components are operating normally.
- **Control** presents a console with which you can send commands (such as startup or shutdown) to the e*Gate component that is selected in the Navigator. Using the Control tab is discussed in *e*Gate Integrator User's Guide*.

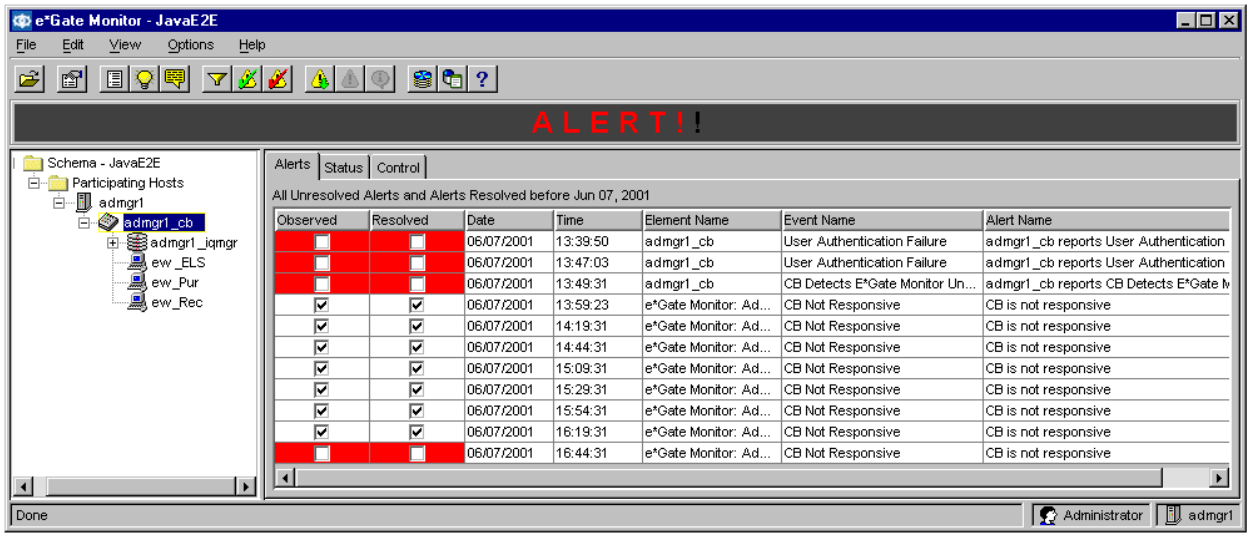
3.3.1 Viewing Alert and Status Messages

To view Alert or status messages

1. In the Navigator, select the component you wish to monitor. The component you select determines which messages you view.
   - Select the **Schema** folder to view all messages for all components within the schema.
   - Select a **Control Broker** to view all the messages from all the components that are supervised by that Control Broker.
   - Select a component to view any messages that pertain to it. Only messages for that component will display.
2. Click the **Alerts** or **Status** tab.

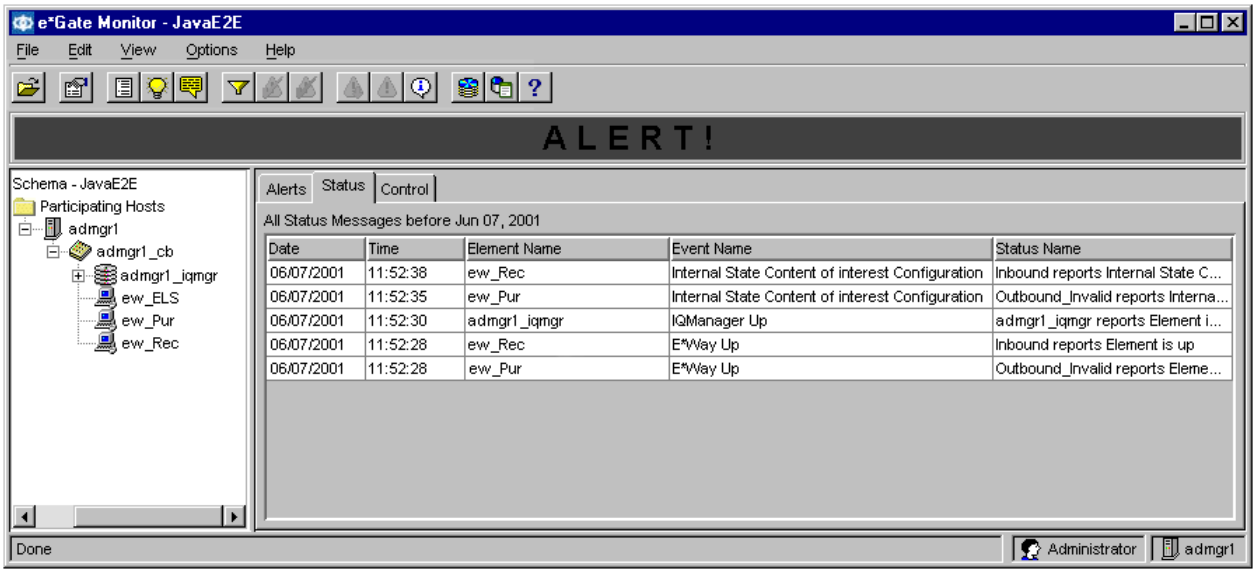
Figure 3 shows typical notification messages on the **Alert** tab and **Figure 4 on page 27** shows typical notification messages on the **Status** tab.

Figure 3 Notification messages on the Alert tab



- The **Element Name** field lists the name of the element from which a monitoring Event is initiated.
- The **Event Name** field lists the name of the monitoring Event from which the Alert notification is generated.
- The **Alert Name** field lists that actual Alert notification. Created by the Control Broker, it is a combination of the information found in the **Element Name** and **Event Name** fields.

Figure 4 Notification messages on the Status tab



To view troubleshooting tips about an Alert's possible causes and remedies

1 Select an Alert.

2 On the toolbar, click  .

To display additional details about an Alert or Status message

1 Select an Alert or status message.

2 On the toolbar, click  .

The Details screen also shows you the list of recipients to whom a notification regarding a status or Alert message may be sent, for every escalation level, that has been specified in the notification routing script. See **Chapter 8** for more information on notifications.

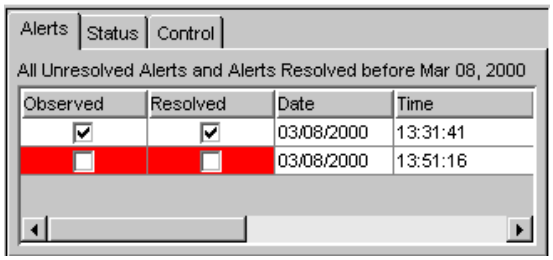
If you are using the e*Gate Integrator Alert Agent, this may include e-mail addresses. Note that this list does not indicate the recipients to whom a message *has been* sent, but to whom a message *may be* sent, if the message is escalated sufficiently. See the *e*Gate Integrator Alert Agent User's Guide* for more information about how the escalation system works.

Complete instructions for selecting, displaying, and filtering the displayed messages are contained in the e*Gate Monitor's Help system, as is a reference for all the status and error codes the e*Gate system can process.

3.3.2 Marking Alerts 'Observed' and 'Resolved'

The left-most columns on the Alerts tab are check boxes for marking Alerts *observed* and *resolved*. These enable you to track your progress as you address each Alert (see Figure 5).

Figure 5 "Observed" and "Resolved" check boxes



- Marking an Alert *observed* means that you have seen the notification. When all Alerts have been marked observed, the "Alert" sign at the top of the window will stop flashing. The "observed" box only affects your own monitor; the "observed" state is not reported to other users.
- Marking an Alert *resolved* indicates that you have addressed and remedied the issue causing the Alert. Unresolved Alerts are automatically escalated for as many levels of escalation as have been defined (see **"Escalation" on page 72** for additional information about escalation). When you mark an Alert "resolved," that change is reported to other monitors. If your site uses an escalation system, be sure to mark Alerts as resolved as soon as you have resolved them.

Marking an Alert "resolved" automatically marks it as "observed."

Note: *Since status messages require no resolution or escalation, there are no comparable check boxes on the Status tab.*

To mark an Alert as observed or resolved

- Check the appropriate box corresponding to the desired Alert message. Once you have made a mark, you cannot remove it.

To mark all Alerts as observed

- On the toolbar, click  .

To mark all Alerts as resolved

- On the toolbar, click  .

Alerts that have been resolved by one user appear as "resolved" automatically on other user's monitors. However, if a given user is logged in to more than one monitor, changes made in one instance of the monitor will not be automatically broadcast to other instances of the same user's monitor.

3.3.3 Automatic Notification Resolution

In addition to marking notifications "resolved" manually (as described in **"Marking Alerts 'Observed' and 'Resolved'" on page 28**), the Control Broker can automatically resolve certain notifications. For example, "Component down" Monitoring Events can be resolved by the same component sending a "Component up" Monitoring Event. When this happens, the Control Broker automatically resolves the appropriate notifications, and sends this information to any logged-in interactive monitors.

The list of Monitoring Events that can resolve another Monitoring Event are included within the Event's header. See the *e*Gate Integrator System Administration and Operation Guide* or the Resolution tab of the Event Details dialog box for more information on the EventHeader nodes.

To view the Event codes for Events that can resolve a selected Monitoring Event

1. In the e*Gate Monitor, select the **Alerts** tab.
2. Select an **Event**.
3. On the toolbar, click  .

   The **Issues & Recipient Information** dialog box appears.
4. At the top of the dialog box, select the **Event** tab.
5. In the middle section of the dialog box, select the **Resolution** tab.

Any codes for Events that can resolve the currently selected Event will appear in the **Can Be Resolved By** box. If no Event codes appear in the list, the selected Event cannot be resolved automatically.

3.3.4 Reconnecting to the Control Broker

If the e*Gate Monitor loses its connection to the Control Broker (either because of a problem with a network connection or because the Control Broker has shut down), you must reconnect the monitor to the Control Broker before you can continue monitoring components supervised by that Control Broker.

In the Navigator, disconnected Control Brokers are indicated by  .

To reconnect to a Control Broker

1. In the Navigator, right-click the disconnected **Control Broker**.
2. On the popup menu, click **Connect**.

If this procedure does not work, do the following:

1. From the **File** menu, select **Login**.
2. Log in (again) to the current Registry host.
3. Open the schema you want to monitor.

Note: *You cannot reconnect to a Control Broker marked "never started." Such Control Brokers must be started using other means (for example, the -sm or -sa command-line options). See "Control Broker:stccb" in the* ***e*Gate Integrator System Administration and Operations Guide*** *for more information.*

3.3.5 Entering Comments

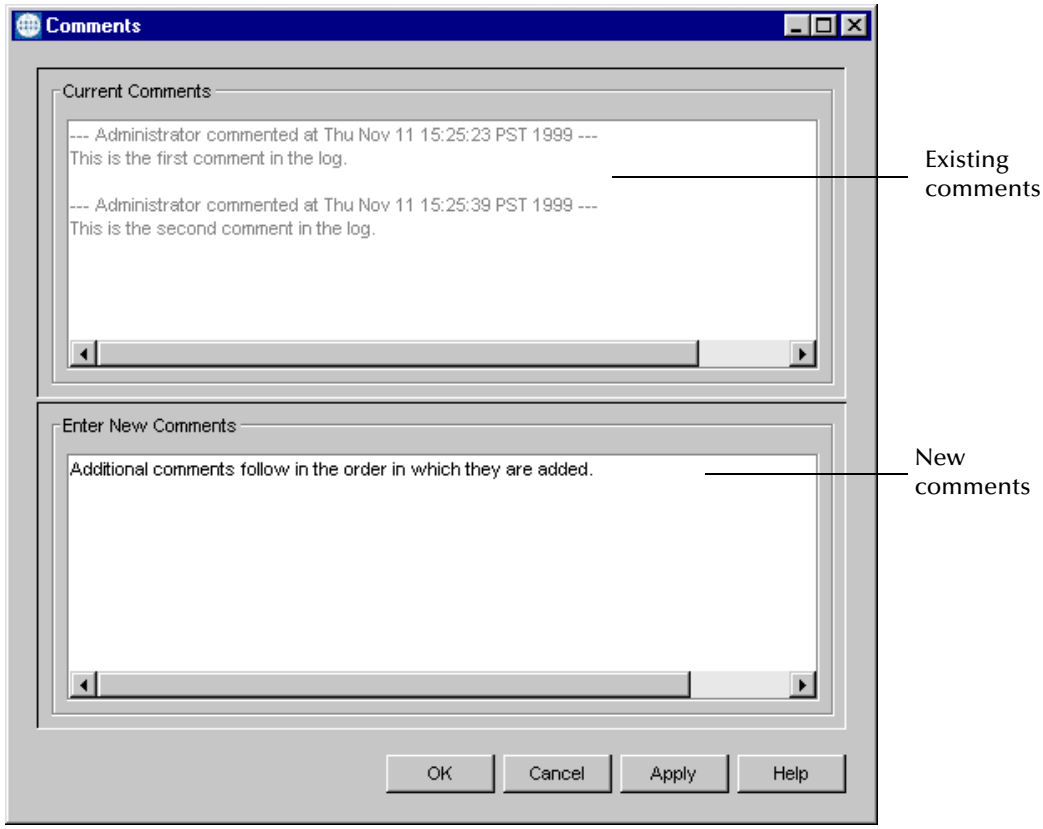
Comments enable system administrators to log their own activity while redressing Alert conditions. Each time you enter a comment for an Alert, that comment is appended to that Alert's "comment log," which can then be read by other users whose monitors receive that Alert.

To enter comments

1. Select the **Alert** for which you want to enter comments.
2. On the toolbar, click  .

   The Comments dialog box appears (see **Figure 6 on page 31**).

Figure 6 Entering Comments



Note: *You may also access this feature using the* ***Comments*** *option under the* ***View*** *menu. In addition, access this option (with* ***Details*** *and* ***Notification Tips****) under a pop-up menu. Access the pop-up by right-clicking on an Alert/status notification message, using the mouse.*

3 Enter any appropriate comments and click **OK**.

Note: *We recommend that anyone using the Monitor should enter comments as soon as they begin to address an Alert. This not only enables you to track your work, but provides real-time information to others about what steps are being taken to remedy any given situation.*

Because Comments are "signed" automatically by the user who submits them (and for other reasons—see the previous section), we recommend that each monitoring user log into an e*Gate Monitor under his own user name.

To view comments

- Repeat the **procedure on page 30** to enter comments and view the log of comments (as shown in Figure 6).

To determine which Events have associated comments

1 On the toolbar, click  .

2 Select the **Comments** check box.

3 Click **OK**.

A "Comments" column will display in the Alerts tab, listing the user who last added a comment and the time at which that column was added.

Customizing Columns on Alerts and Status Tabs

To change the size of a column

Drag the border between elements within the column heading.

To sort the display based on a selected column

Click the column heading.

To select which columns should be displayed

1 On the toolbar, click  .

2 Select the columns you want displayed and click **OK**.

Filtering notifications

You can use the filter to select the date and time after which to display resolved notifications (on the **Alert** tab) or status messages (on the **Status** tab).

Note: *You can only filter **resolved** notifications. You cannot filter unresolved notifications, but you can limit the number that are displayed. See the next section for more information.*

To set the date filter

1 Select the tab (**Alert** or **Status**) on which you want to apply the filter.

2 On the toolbar, click  .

The **filter** dialog box appears.

3 Use the controls in the **filter** dialog box to select the date and time after which to display resolved notifications (on the Alert tab) or status messages (on the Status tab).

4 Click **OK**.

There is no filter that you can apply to *unresolved* Alerts.

Right-click Shortcut Menus

The e*Gate Monitor supports right-click shortcut menus within the Navigator pane. Any command that can be issued via the **Control** tab can be issued using a shortcut menu.

To use a right-click shortcut menu

1. In the Navigator, right-click on any component.
2. From the popup menu, click the command you wish to issue.

3.3.6 Limiting the Number of Displayed Notifications

If you need to conserve system memory, you can lower the total number of notifications the e*Gate Monitor will display. The Monitor displays the most recent notifications up to the limit that you set.

You can set limits separately for the following notifications:

- Unresolved Alerts
- Resolved Alerts
- Status notifications

Once the limit has been reached, older notifications are replaced by newer ones in "first in, first out" order. If the e*Gate Monitor is currently displaying its maximum number of notifications, you may notice a drop in system performance if a large number of new notifications are processed at once.

To set a notification limit

1. Launch the e*Gate Monitor and select any schema. (The limits you set affect all schema.)
2. On the Monitor's toolbar, click  or select **Edit > Properties** from the Menu bar.
3. When the **Table Properties Configuration** dialog box opens, select the **Maximum Notifications** tab.
4. Change the values to desired values and click **OK**.

3.3.7 Increasing the Number of Displayed Notifications

Normally, the Monitor can only display a maximum of 600 notifications without causing "out of memory" errors. If the 600-notification limit is too low, you can increase the amount of heap memory for the Monitor by modifying the shortcut that launches the Monitor.

Add the command flag **-mxNNm** to the shortcut, where **NN** is the amount of heap memory. An example is shown below (the central section of the command line is not shown):

C:\EGate\client [... many items follow...] \1.1\bin\jrew.exe -mx64m"

The value in the example above (64) increases the heap memory to 64 MB and enables you to display approximately 5500 notifications using the sample **Notification.tsc**. A heap-memory limit of 40 MB enables you to display approximately 3400 notifications. The actual maximum vary depending on the complexity of the Collaboration used for your notification routing.

See the Windows Help system for more information about modifying shortcuts, or the JRE documentation for more information about heap memory.

3.4 Viewing Alerts Using Command-line Monitoring

The **stccmd** command utility provides comparable monitoring and control functions as the e*Gate Monitor in a command-line application. You can use this interactive monitoring feature at any time you want.

Note: *SeeBeyond recommends that you use the e*Gate Monitor GUI feature as your primary monitoring tool.*

3.4.1 Using stccmd Interactively

To launch stccmd

- At the shell prompt, type the following command (on a single command line):

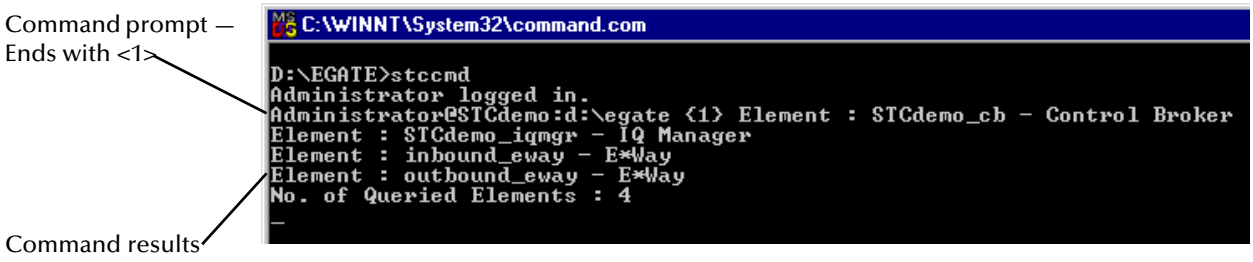
```
stccmd -rh RHost -rs Schema -cb CBName -un User -up Passwd
```

where:

RHost is the name of the e*Gate Registry Host,
Schema is the name of a schema,
CBName is the name of a Control Broker within that schema, and ***User*** and ***Password*** are valid e*Gate login information.

The **stccmd** output differs slightly between Windows and UNIX systems. Figure 7 shows the Windows output.

Figure 7 Appearance of stccmd under Windows

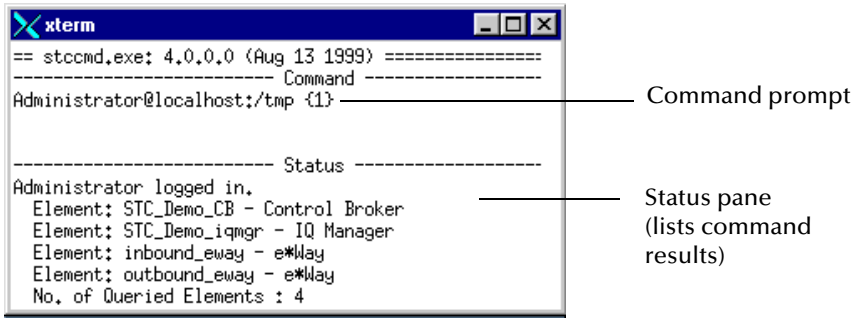


Under Windows, the command output is not always clearly separated from the command line and can often appear immediately after the command prompt, as seen in the figure above. To redisplay the command prompt, press **Enter**.

Note: *On Windows systems, we recommend you use* **stccmd** *for non-interactive (batch) operations only; see* **"Using stccmd Non-interactively" on page 38** *for more information. The graphic e*Gate Monitor provides superior functionality for interactive monitoring on Windows systems.*

Figure 8 illustrates **stccmd** under UNIX.

Figure 8 Appearance of stccmd under UNIX



On UNIX systems, the **stccmd** utility uses a two-pane display. The top pane provides the command prompt; the bottom displays the command results.

On both operating systems, the command prompt takes the following format:

```
username@parthost:/directory{N}
```

where:

username is the name of the e*Gate user running the utility,
parthost is the name of the Participating Host being monitored,
directory is the path from which the **stccmd** utility was started, and
N is the history number of the command (the first command is number 1, the second command is number 2, and so on).

To issue commands

Under Windows, you need sufficient privilege to add services to the Windows Registry (see the Help system for the Windows utility **regedt32** for more information about registry security). Under UNIX, there are no privilege restrictions.

- Type the command and any optional arguments (available commands are listed in Table 1). The command will appear automatically in the Command pane; the results will appear at the cursor location (Windows systems) or in the Status pane (UNIX systems).

To display a list of commands

- At the **stccmd** prompt, type **help** or **?** to display a page of options. To display the next page, press the space bar at the "More" prompt. To exit the Help display, type **q**.

Table 1 lists the valid **stccmd** command arguments. Optional arguments are listed in [square brackets].

Table 1 Command Arguments for stccmd

Command	Arguments [ ] = optional	Purpose
!	numeral	Re-execute command number *numeral*.
! !		Repeat last command.

Table 1 Command Arguments for stccmd (Continued)

Command	Arguments [ ] = optional	Purpose
?		Display list of commands.
activate	component-name	Activate a suspended component. See **Note on page 37**.
attachiq	IQ-name	Attach a detached Intelligent Queue (IQ).
cls		Clear the status window.
detachiq	IQ-name	Detach an IQ (stop the IQ from receiving published data).
exit		Exit **stccmd**.
getres	[-b *begindate*] [-e *enddate*] (dates in format mm/dd/yyyy)	Show all resolved notifications [within specified date range].
getstatus	[-b *begindate*] [-e *enddate*] (dates in format mm/dd/yyyy)	Show all status notifications [within specified date range].
getunres	[-b *begindate*] [-e *enddate*] (dates in format mm/dd/yyyy)	Show all unresolved notifications [within specified date range].
help		Display a list of commands.
list	One of the following: all (default) monitors *or* -m alertors *or* -a control *or* -c -n all (default for -n) -n -b *begindate* -e *enddate* (dates in format mm/dd/yyyy) -n +r -n -r -n -i *number* -n -*element-name*	List the following: All elements Monitors Elements that can send alerts Control Brokers All notifications Notifications within date range Resolved notifications Unresolved notifications Notification *number* Notifications for *element*
quit		Exit **stccmd**.
reload	e*Way-name	Reload the configuration for e*Way-*name*. See the **Note on page 37**.
resolve	notification-number	Mark a notification as resolved (see **"Marking Alerts 'Observed' and 'Resolved'" on page 28**).
sequence	*component-name* [*number*]	Show *component's* message sequence number (or change it to *number*). See the **Note on page 37**.
shell	command-string	Run *command-string* in an external shell.
shutdown	component-name	Shutdown *component*.
shutdownall		Shutdown all components supervised by the specified Control Broker. Does not shut down the Control Broker.

Table 1 Command Arguments for stccmd (Continued)

Command	Arguments [ ] = optional	Purpose
start	component-name	Start *component*.
status	component-name	Display status of *component*.
suspend	component-name	Suspend *component's* execution. See the **Note on page 37**.
version	component-name	Display the version number of *component*.

Notes

1. Most e*Way Intelligent Adapters support the **suspend** and **activate** commands. The **suspend** command effectively takes a component "off line" but does not end the executing process. When the command is received, the component finishes processing any in-process Events, then goes into a "wait" state (e*Ways will also close any open connections to external systems). The **activate** command brings the component back on-line. Events cannot be published to suspended components, but will remain in their source location (another IQ or an external system) until the subscribing component is reactivated or another subscriber obtains them.
2. The **reload** command only affects one or two types of e*Ways for which you must explicitly reload configuration changes (the e*Way's user's guide will instruct you if this is necessary). Most types of e*Way automatically reload their configuration whenever the configuration is changed, and for them, this command has no effect.
3. The **sequence** command only affects e*Ways.

To list all components

- Type **list**.

To start a component

- Type **start** ***component-name***.

To shut down a component

- Type **shutdown** ***component-name***.

Note: *Components that are configured as "Start automatically" in their properties dialog box* ***Startup*** *tab will not automatically restart after you shut them down manually. However, they will restart on schedule if a schedule has been defined and calls for a scheduled startup.*

If you shut down a component that has been configured to start automatically, it will restart automatically if you use the Enterprise Manager to apply any configuration changes to the component.

3.4.2 Using stccmd Non-interactively

You can also use **stccmd** to issue e*Gate commands from the command line without launching **stccmd** in its interactive "windowed" mode. For example, you might use this capability to issue commands from command scripts or cron jobs.

To issue a command non-interactively

- At the shell prompt, type the following:

```
stccmd -rh RHost -rs Schema -cb CBName -un User
    -up Passwd -cmd "command"
```

where:

RHost is the name of the e*Gate Registry Host,
Schema is the name of a schema,
CBName is the name of a Control Broker within that schema,
User and ***Passwd*** are valid e*Gate login information, and
command is a valid **stccmd** command (including any necessary arguments).

You can only issue one command at a time.

Chapter 4

Using Log Files

This chapter explains how to use, view, and understand log files, including debug levels and flags. It also includes helpful information on how to do basic troubleshooting using log files, including examples.

4.1 Overview of Log Files

The e*Gate system creates a log file for each executable component, that is, a component that requires an **.exe** file for its operation. In e*Gate, the executable components are:

- e*Way Intelligent Adapters
- Business Object Brokers (BOBs)
- Intelligent Queue (IQ) Managers
- Control Brokers
- e*Insight Business Process Manager modules

Note: *In e*Gate, executable components are also called* ***modules****.*

The rest of this chapter tells you how to use log files, including viewing and understanding them, as well as basic e*Gate troubleshooting techniques possible employing these files. Troubleshooting explanations include commented examples.

This chapter explains

4.2 Log File Properties

Log files have the following basic properties you need to know:

- **Location:** They are created in the **\egate\client\logs** directory on the Participating Host (on the same drive on which e*Gate is installed) executing the module.
- **Names:** They are named for the module that creates them (for example, the e*Way **ewPOS_IN** creates a log file called **ewPOS_IN.log**).
- **Handling:** Log file archiving and management takes place outside of the e*Gate environment.

Log files are stored in a directory defined within the **.egate.store** file. The directory **\eGate\client\logs** is the default and can be changed by editing this file.

4.2.1 Log File Maintenance

You can archive, delete, copy, or rename log files at any time (for example, using an **at** or **cron** job). Keep in mind that while e*Gate is writing information to log files, it is also using additional system resources.

Impact on performance

Log files can have the following impact on system performance:

- As more log files are created, they take up more hard disk space. Be sure to delete log files on a regular basis, as quickly as possible, once you are finished with them. Otherwise, they can rapidly consume large amounts of volume.
- Writing information to log files and tracing that information consumes system resources, including memory and disk space. Handling over-large amounts of logging slows system performance. You can reduce the impact of logging on your system by:
  - Using less detailed debug levels, for example, INFO instead of TRACE
  - Reducing the number of debug flags in use

For an explanation of how to change logging options (debug levels and flags) using the Enterprise Manager, see the **procedure on page 19**.

Note: *If available disk space is a concern, you can set a disk-usage threshold at which e*Gate will automatically send you a warning Alert. See the* ***e*Gate Integrator System Administration and Operations Guide*** *for more information.*

4.2.2 Viewing Log Files

Use a text editor to view a module's log file. You can view the log while the module is still running. However, depending on the editor, you may need to reread the file to refresh your view of the log data.

Note: *You cannot get log-file updates on the fly.*

To view an e*Gate log file

1. Navigate to the directory **\eGate\client\logs**.
2. Open a log file with a text editor (such as **Notepad** or **vi**).

If you are viewing the log file of a running module:

- You must periodically refresh the copy of the file in the editor. You cannot view log data in automatically updated real-time format unless your editor supports this function.
- Alternatively, you can reopen the file at any time to view the most recently added data, or on UNIX systems, use the shell command **tail -f**.
- You can also use command-line arguments to turn on logging; see the *e*Gate Integrator System Administration and Operations Guide* for more details about command-line applications and arguments.

Using the display command

You can also use the Monk command **display** to send a file to the specified output port. The port is optional. If not present, the system defaults to the standard output port. To perform this command, use the following syntax:

display ***object*** **[*port*]**

where:

object is the object to display at the output port and ***port*** is the handle to the open port (optional).

4.3 Understanding Log Files

This section explains how to read and understand log files, including debug flags. It also explains how to view debug tracing.

4.3.1 Log File Structure

All log entries are appended to the log file. e*Gate log files are normally closed except when an entry is being written. If you receive an error when you attempt to access the log file, simply try again.

Each time an e*Gate module starts, it writes a timestamp header to the log file, such as the one shown below:

```
------------------------------------------------------------
[15-Sep-1999 15:29:45.585] START LOG FOR: eway_in on STC1 pid: 222
------------------------------------------------------------
```

Each line in the log file is divided into columns. Typical e*Gate log file entries look like those shown in Table 2. This example shows short lines for ease of reading. Most entries are much longer.

Table 2 Typical Log File Entries

Timestamp	Debug Flag	Log Level	Thread No.	Source File and Source Line No.	Actual Log Entry
15:29:46.497	APP	I	233	(svcmain.cxx:124):	Service: Started
15:31:46.800	COM	I	233	(tracelog.cxx:313):	*** Trace Mask Changed
15:31:46.850	REGV	T	233	(msgfuncs.cxx:396):	I_SendMessage(): sent

Table 3 explains the columns in the last log-file line in Table 2.

Table 3 Log-file Data Analyzed by Column

Entry	Description	Purpose
15:31:46.850	Timestamp	Time at which the Event occurred (to the millisecond). Timestamps use the local time of the component generating the entry.
REGV	Debug flag or topic	Debug flag that caused this system Event to be logged.
T	Logging level (severity code)	Logging level of this system Event (see **"Debug Levels" on page 43**).
233	Thread number	A pointer that identifies a node's position within a log; used to facilitate the traversal of the log's tree structure.
(msgfuncs.cxx:396)	Source file and source line number	Refers to the position in the code that is communicating with the current module.
I_SendMessage(): sent	Event text/message	Most of this information is for SeeBeyond use when the customer cannot solve the problem. If this occurs, you can forward this information to SeeBeyond Customer Support.

You can use log file information for some basic troubleshooting purposes. See **"Troubleshooting with Log Files" on page 47** for details.

4.3.2 Setting Levels and Debug Flags

You can control the amount and types of information that log files contain by setting debug levels and flags. Use the Enterprise Manager to change these settings as explained under **"Activating Logging" on page 19**. These settings operate as follows:

- **Debug Levels** set the amount and type of information logged for the selected module.
- **Debug Flags** set the area within the module wherein logging is desired.

Speed or maintainability are the key for deciding upon the type of information that will be tracked. If less information is logged, the process completes faster and takes up less hard-disk space than if a large amount of information was requested.

For example, if the information you require changes quickly and is not necessary to maintain for any length of time (like stock quotes), the INFO (I) debug level and display counts (CNTS) debug flag might log all the information you require.

However, if you need to know more detail for a specific module, you might select TRACE for the debug level and internal queue verbose (IQV), configuration (CFG), and e*Way verbose (EWYV) for the debug flags.

Debug Levels

Table 4 lists the e*Gate debug levels. These levels can be set for each module within the Enterprise Manager (see the **procedure on page 19**).

Note: *If debug flags are changed from within the e*Gate Monitor, the default level is set to TRACE.*

Table 4 Debug Levels

Level	Name	Description
T	TRACE	Extensive details of internal module operations (logs significant amounts of information). All application program interface (API) calls are reported.
D	DEBUG	Details of internal module operations. A significant amount of detail is reported.
I	INFO	Basic status information, such as service/application startup or parameter changes.
W	WARNING	Problems that will not cause operation or application failure; faults that were satisfactorily handled. This is an unexpected condition and is an attention request.
E	ERROR	Problems that may cause a given operation to fail, but not serious enough to cause the module itself to halt.
F	FATAL	Errors that halt a module's execution.
N	NONE	No information is recorded.

An additional level of logged Events, A (for API), is automatically entered in the log file if levels W, E, or F are selected, but you cannot select this logging level in the Enterprise Manager.

Note: *DEBUG and TRACE logging creates significant numbers of entries in log files and can also impact system performance.*

Debug Flags

The topics (or processes) logged to a module's log file are dictated by the debug flags selected in the properties for that module. Table 5 lists and explains the e*Gate debug flags.

Table 5 List of Debug Flags

Debug Flag	Description	Use To Troubleshoot or Log
API	SeeBeyond API: Messages pertaining to uses of SeeBeyond API calls (very large category)	e*Gate internal functions.
CB	Control Broker: Messages sent by the Control Broker to other modules, or describing Control Broker internal operations	Problems with the Control Broker. Debugging information does not only deal with problems; all activities performed within the realm of the Debug flag will be reported.
CFG	Configuration	Module configuration changes.
COL	Collaboration Services	SeeBeyond predefined DLLs and customer defined DLLs. That is, logging that pertains to Collaborations (Subscriptions/ Publications, Event Types) will be reported.
COM	SeeBeyond Common: Messages generated by common modules (very large category)	e*Gate internal functions.
CNTS	Display counts	Data throughput. **Note**: The CNTS debug flag must be turned on before the **Status** display in the Monitor will show how many Events a given e*Way or BOB has processed. If the debug level is also set to TRACE, a count will be written to the log file for every 100 Events processed.
DB	Database	Interactions with external databases.
EWY	e*Way	e*Way operations.
IP	TCP/IP: Details of IP functions performed as e*Gate components communicate with each other	Network problems, or Internet protocol (IP) connectivity issues. Debugging information does not only deal with problems; all activities performed within the realm of the Debug flag will be reported.
IQ	Internal Queue: IQ operations	Problems with IQs or the IQ Manager, or whether Events are being inserted into IQs. Debugging information does not only deal with problems; all activities performed within the realm of the Debug flag will be reported.
MNK	Monk	Collaboration logic or usage.

Table 5 List of Debug Flags (Continued)

Debug Flag	Description	Use To Troubleshoot or Log
MSG	Message: Messages describing the handling of Events. Includes only message-header information	Problems with the way Collaborations are processing Events. Debugging information does not only deal with problems; all activities performed within the realm of the Debug flag will be reported.
MSGP	Message parse	How Event nodes are being parsed within a Collaboration script.
REG	Registry: Messages describing interactions with the e*Gate Registry	Modules are having problems communicating with the Registry, or knowing whether information is being properly uploaded/downloaded.

Exactly which flags you choose will depend upon which topics you wish to observe in each module.

Verbose mode

Verbose-mode debug flags display additional lower-level details than basic flags. Because of the increased amount of logged information, using these flags usually causes a significant increase in the size of the log file and can decrease the quality of overall system performance.

As a result, the verbose-mode Alert Events generated by these debug flags illustrate much more detail or lower-level details than normal Events. For example, an Alert Event with an **IP** flag could simply read "Not Open," a system Event with an **IPV** flag could read "Connection reset on socket 300."

Note: *See* **Chapter 5** *for more information on Alert Events.*

The following verbose-mode debug flags are available:

APIV	IQV	IPV
COLV	MSGV(*)	MNKV
COMV	CBV	REGV
EWYV	DBV	

* = Logs complete Event content as well as Event header

To choose all the verbose-mode debug flags

- Click **Select All**.

Note: *SeeBeyond recommends that you first use the basic debug flags listed (see Table 5) before proceeding to verbose-mode flags. For the best system performance, use as few verbose-mode flags as possible.*

Examples

The following example shows a log file for a BOB with the INFO debug level set and no debug flags.

```
--------------------------------------------------------------------------------
[07-May-2000 18:35:11.830] *** START LOG FOR: bobPROMOTION on akhosravy pid: 281 ***
--------------------------------------------------------------------------------
18:35:11.830 COM  I 259 (tracelog.cxx:373): *** Trace Mask Changed From 0x00000000-
0x00000000 To 0x00000000-0x000000F7 - log file is on ***
18:35:12.611 API  I 259 (configload.cxx:126): Configuration for bobPROMOTION
18:35:12.621 API  I 259 (configload.cxx:129):      Workslices: 1
18:35:12.621 API  I 259 (configload.cxx:136):      Collab: crFIND_PROMO (1)
18:35:12.621 API  I 259 (configload.cxx:150):          Input Topics: 2
18:35:12.641 API  I 259 (configload.cxx:610):              Msg: etNEW_POS_DATA - IQ:
iqPOS_DATA2 - Cntrl: Yes Def: No
18:35:12.641 API  I 259 (configload.cxx:610):              Msg: etNEW_POS_DATA - IQ:
iqPOS_DATA - Cntrl: Yes Def: No
18:35:12.641 API  I 259 (configload.cxx:163):          Output Topics: 2
18:35:12.651 API  I 259 (configload.cxx:610):              Msg: etALL_POS_DATA - IQ:
iqFIND_PROMO - Cntrl: No Def: Yes
18:35:12.651 API  I 259 (configload.cxx:610):              Msg: etPROMO_DATA - IQ:
iqFIND_PROMO - Cntrl: No Def: No
18:35:12.651 EWY  I 259 (svcmain.cxx:88): ServiceMain(): Started
18:35:14.083 API  I 342 (recovery.cxx:575): Minor sequence number for IQ handle to queue
iqFIND_PROMO starts at 28
18:35:14.093 API  I 342 (recovery.cxx:575): Minor sequence number for IQ handle to queue
iqFIND_PROMO starts at 6
18:36:24.484 API  I 259 (eventutils.cxx:88): Got CB Shutdown request
--------------------------------------------------------------------------------
[07-May-2000 18:36:27.618] *** END LOG FOR: bobPROMOTION on akhosravy pid: 281 ***
--------------------------------------------------------------------------------
```

The following example shows a portion of a log file for an e*Way with all debug flags set.

```
12:44:00.581 API  I 317 (configload.cxx:126): Configuration for ewPOS_IN2
12:44:00.581 API  I 317 (configload.cxx:129):      Workslices: 1
12:44:00.591 API  I 317 (configload.cxx:136):      Collab: crTRANSFORM_INPUT (1)
12:44:00.591 API  I 317 (configload.cxx:150):          Input Topics: 1
12:44:00.591 IPV  D 317 (iosocketclient.cxx:863): SendAppMessage: msg command: rENM, data: 112
12:44:00.601 IPV  D 317 (iosocketclient.cxx:1121): WaitForAppMessage: msg len: 24 code: rENM compressed: NO
12:44:00.601 IPV  D 317 (iosocketclient.cxx:863): SendAppMessage: msg command: rFHD, data: 16
12:44:00.601 IPV  D 317 (iosocketclient.cxx:1121): WaitForAppMessage: msg len: 112 code: rFHD compressed: NO
12:44:00.601 API  I 317 (configload.cxx:610):              Msg: etRAW_POS_DATA2 - IQ: <none> - Cntrl: Yes Def:
No
12:44:00.601 IPV  D 317 (iosocketclient.cxx:863): SendAppMessage: msg command: rCHD, data: 8
12:44:00.601 IPV  D 317 (iosocketclient.cxx:1121): WaitForAppMessage: msg len: 24 code: gACK compressed: NO
12:44:00.601 API  I 317 (configload.cxx:163):          Output Topics: 1
12:44:00.611 IPV  D 317 (iosocketclient.cxx:863): SendAppMessage: msg command: rENM, data: 112
12:44:00.611 IPV  D 317 (iosocketclient.cxx:1121): WaitForAppMessage: msg len: 24 code: rENM compressed: NO
12:44:00.611 IPV  D 317 (iosocketclient.cxx:863): SendAppMessage: msg command: rFHD, data: 16
12:44:00.611 IPV  D 317 (iosocketclient.cxx:1121): WaitForAppMessage: msg len: 112 code: rFHD compressed: NO
12:44:00.611 IPV  D 317 (iosocketclient.cxx:863): SendAppMessage: msg command: rENM, data: 112
12:44:00.611 IPV  D 317 (iosocketclient.cxx:1121): WaitForAppMessage: msg len: 24 code: rENM compressed: NO
12:44:00.611 IPV  D 317 (iosocketclient.cxx:863): SendAppMessage: msg command: rFHD, data: 16
12:44:00.631 IPV  D 317 (iosocketclient.cxx:1121): WaitForAppMessage: msg len: 112 code: rFHD compressed: NO
12:44:00.631 API  I 317 (configload.cxx:610):              Msg: etNEW_POS_DATA - IQ: iqPOS_DATA2 - Cntrl: No
Def: Yes
12:44:00.631 IPV  D 317 (iosocketclient.cxx:863): SendAppMessage: msg command: rCHD, data: 8
```

Note the differences in information that exist between the two examples.

After initial testing

After initial testing, use debug flags selectively. Turning on all flags can result in unnecessary file input/output, and will generally result in huge files that contain significantly more information than necessary for troubleshooting a given product (see **Chapter 5** for more information).

Note: *For additional information on API and command-line related topics, see the* ***e*Gate Integrator System Administration and Operations Guide****.*

4.4 Troubleshooting with Log Files

This section provides an overview of basic e*Gate troubleshooting, which you can do using log files. It also includes examples of actual log files that correspond to each problem and/or area of interest.

4.4.1 General Troubleshooting

You can use the following text strings to search for problems and/or events indicated by log files:

- **>>>>** — This string is always an "attention" signal within a log file.
- **MONKEXCEPT**. Monk errors from Collaborations will begin with this text.
- **ROLLBACK**. The failed publication of an Event is indicated by this string.
- **COMMIT**. A successful publication has resulted when this text appears.

Leading and trailing spaces surround the individual letters (for example, **F**). You must have already set the appropriate debug flag to display any of the text strings in the previous list.

Note: *The most common error most first-time e*Gate developers will find in a log file is "Unable to load module configuration." This error means that you have created an e*Gate module, but not assigned both an executable file and a configuration file.*

4.4.2 Troubleshooting by Debug Flag

This section gives you directions on how to record specific types of most frequently needed information through the use of debug flags and includes examples.

API and APIV debug flags

The API and APIV (API Verbose) debug flags cause the system to report configuration information and related errors in log files. See the examples below.

API Debug Flag

```
07-May-2000 12:58:30.972] *** START LOG FOR: bobPROMOTION on akhosravy pid: 260 ***
-------------------------------------------------------------------------------
12:58:30.972 COM  I 324 (tracelog.cxx:373): *** Trace Mask Changed From 0x00000000-0x00000000 To 0xFF360FFB-
0x00000000 - log file is on ***
12:58:31.082 API  I 324 (configload.cxx:126): Configuration for bobPROMOTION
12:58:31.082 API  I 324 (configload.cxx:129):      Workslices: 1
12:58:31.092 API  I 324 (configload.cxx:136):      Collab: crFIND_PROMO (1)
12:58:31.112 API  I 324 (configload.cxx:150):          Input Topics: 2
12:58:31.122 API  I 324 (configload.cxx:610):              Msg: etNEW_POS_DATA - IQ: iqPOS_DATA2 - Cntrl: Yes
Def: No
12:58:31.142 API  I 324 (configload.cxx:610):              Msg: etNEW_POS_DATA - IQ: iqPOS_DATA - Cntrl: Yes
Def: No
12:58:31.152 API  I 324 (configload.cxx:163):          Output Topics: 2
12:58:31.162 API  I 324 (configload.cxx:610):              Msg: etALL_POS_DATA - IQ: iqFIND_PROMO - Cntrl:


FAILED:InsufficientData:product_description:ON:1:1:::-1:(-1):">"        -> "".
12:59:20.103 API  T 394 (translate.cxx:315): MonkTransLoadFailed E:0x20000050 (incorrect format)
12:59:20.103 API  A 394 (workitems.cxx:1757): TransFailed E:0x20000050 (incorrect format)
12:59:20.103 IQV  D 394 (iqmark.cxx:72): MsgMark: queue: iqPOS_DATA
12:59:20.113 MSG  T 394 (iqmark.cxx:194): Event Track (Mark: end): IQ [iqPOS_DATA] Msg [etNEW_POS_DATA] Mark
[0x00000000]
12:59:20.113 MSG  T 394 (workitems.cxx:560): *ROLLBACK* inbound message [etNEW_POS_DATA] seq major: 0 minor: 16
priority: 0 enqueue: 05/07/2000 11:58:32.574
12:59:20.113 APIV D 324 (defeventproc.cxx:91): DefaultEventProc: STCE_INMESSAGE_ROLLBACK (An inbound message has
been rolled back)
```

APIV Debug Flag

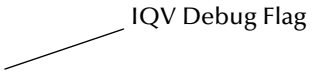
CFG and EWY debug flags

The CFG and EWY debug flags report e*Way activity. See the example below.

```
12:41:17.967 CFG  D 342 (ScCfgLdr.c:1892): loaded STRING cfg param: [General Settings:AllowIncoming=YES]
12:41:17.977 CFG  D 342 (ScCfgLdr.c:1892): loaded STRING cfg param: [General Settings:AllowOutgoing=NO]
12:41:17.977 CFG  D 342 (ScCfgLdr.c:1892): loaded STRING cfg param: [General Settings:PerformanceTesting=NO]
12:41:17.977 CFG  D 342 (ScCfgLdr.c:1892): loaded STRING cfg param: [Outbound (send)
settings:OutputDirectory=C:\DATA]
12:41:17.977 CFG  D 342 (ScCfgLdr.c:1892): loaded STRING cfg param: [Outbound (send)
settings:OutputFileName=output%d.dat]
12:41:17.977 CFG  D 342 (ScCfgLdr.c:1892): loaded STRING cfg param: [Outbound (send) settings:AddEOL=YES]
12:41:17.977 CFG  D 342 (ScCfgLdr.c:1892): loaded STRING cfg param: [Outbound (send)
settings:MultipleRecordsPerFile=YES]
12:41:17.977 CFG  D 342 (ScCfgLdr.c:1858): loaded DWORD cfg param: [Outbound (send)
settings:MaxRecordsPerFile=10000]
12:41:17.977 CFG  D 342 (ScCfgLdr.c:1892): loaded STRING cfg param: [Poller (inbound)
settings:PollDirectory=d:\eGateExercises\temp\input2]
12:41:17.977 CFG  D 342 (ScCfgLdr.c:1892): loaded STRING cfg param: [Poller (inbound)
settings:InputFileExtension=]
12:41:17.977 CFG  D 342 (ScCfgLdr.c:1892): loaded STRING cfg param: [Poller (inbound)
settings:InputFileMask=*.txt]
12:41:17.977 CFG  D 342 (ScCfgLdr.c:1858): loaded DWORD cfg param: [Poller (inbound)
settings:PollMilliseconds=1000]
12:41:17.987 CFG  D 342 (ScCfgLdr.c:1892): loaded STRING cfg param: [Poller (inbound)
settings:MultipleRecordsPerFile=YES]
12:41:17.987 CFG  D 342 (ScCfgLdr.c:1892): loaded STRING cfg param: [Poller (inbound) settings:RemoveEOL=YES]
12:41:17.987 CFG  D 342 (ScCfgLdr.c:1858): loaded DWORD cfg param: [Poller (inbound)
settings:MaxBytesPerLine=4096]
12:41:17.987 CFG  D 342 (ScCfgLdr.c:1892): loaded STRING cfg param: [Poller (inbound)
settings:BytesPerLineIsFixed=NO]
12:41:17.987 CFG  D 342 (ScCfgLdr.c:1858): loaded DWORD cfg param: [Performance Testing:LogEveryXEvents=100]
12:41:17.987 CFG  D 342 (ScCfgLdr.c:1858): loaded DWORD cfg param: [Performance Testing:InboundDuplicates=1]
12:41:17.987 EWY  I 342 (svcmain.cxx:135): Service: Started
```

IQ and IQV debug flags

The IQ and IQV (IQ Verbose) debug flags log interactions with an IQ. This information includes hexidecimal and ASCII representations of each Event published. See the example below.




```
12:44:12.217 IQV  D 261 (iqput.cxx:71): MsgPut: queue: iqPOS_DATA2
12:44:12.217 IQV  D 261 (iqput.cxx:113): sending to iq manager Data Follows (bytes 81):
      30 31 2A 30 31 7E 30 32 2A 38 30 30 31 2A 33 30  |  01*01~02*8001*30
      30 31 2A 31 34 7E 30 33 2A 6E 6F 2A 31 32 33 61  |  01*14~03*no*123a
      62 63 3E 32 30 2E 30 30 3E 31 2A 63 61 73 68 2A  |  bc>20.00>1*cash*
      38 30 2E 30 30 7E 30 34 2A 31 32 33 61 62 63 3E  |  80.00~04*123abc>
      54 65 6E 6E 69 73 20 53 68 6F 65 73 7E 30 35 2A  |  Tennis Shoes~05*
      35                                               |  5
```

The above example uses the hexadecimal system. Representing binary numbers, 16 digits (0 through 9) and letters (A through F/they can be upper or lower case) are used to represent decimal numbers 0 through 15. The hexadecimal system allows computers to express binary numbers in which a byte contains eight binary digits; one hexadecimal digit can represent four binary digits while two hexadecimal digits can represent eight binary digits (a byte).

Table 6 compares decimal (base 10), hexadecimal (base 16), and binary (base 2) equivalents.

Table 6 Decimal/Hexadecimal/Binary Conversion

Decimal	Hexadecimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Using Table 6, you are able to convert binary into hexadecimal, and vice versa.

Note: *Hexadecimal numbers are usually preceded by $, &, or 0x, and followed by H, to prevent confusion with decimal numbers.*

The sample below includes the word "failed," but it does not mean there has been an error. This word indicates that the system failed to find an available Event in a given IQ.

```
12:59:31.719 IQV  A 394 (iqget.cxx:214): NegativeAck E:0x20000020 (item not found)
12:59:31.719 IQV  A 394 (iq.cxx:942): IQMsgGetFailed E:0x20000020 (item not found)
```

Did not find Event

MNK and MNKV debug flags

The MNK and MNKV (MNK Verbose) debug flags are particularly useful in Collaborations executing Monk Collaborations. Be sure to select these flags when using the Monk display command to write to log files within Collaborations. See the example.

```
12:44:12.177 MSGP I 261 (monklog:406): MAPPED:
12:44:12.177 MSGP I 261 (monklog:406): etdPOS2[0]:ON
12:44:12.207 MSGP I 261 (monklog:406): :1:1:
12:44:12.207 MSGP I 261 (monklog:406): :
12:44:12.207 MSGP I 261 (monklog:406): :-1:
12:44:12.207 MSGP I 261 (monklog:406): (-1)
12:44:12.207 MSGP I 261 (monklog:406):  -> "
12:44:12.207 MSGP I 261 (monklog:391):
01*01~02*8001*3001*14*alison~03*no*123abc>20.00>1*80.00~04*123abc>Tennis Shoes~0
12:44:12.207 MSGP I 261 (monklog:406): 5*5~
12:44:12.217 MSGP I 261 (monklog:406): ".
12:44:12.217 MNK  I 261 (monklog:406): FINISHED WITH STANDARDIZATION OF ONE RECORD!
```

MNK Debug Flag

MSG debug flag

The MSG debug flag logs Event-handling activity. Note the word "ROLLBACK" in the example below. This word indicates a failed Event publication.

```
FAILED:InsufficientData:product_description:ON:1:1:::-1:(-1):">"        -> "".
12:59:34.574 API  T 394 (translate.cxx:315): MonkTransLoadFailed E:0x20000050 (incorrect format)
12:59:34.574 API  A 394 (workitems.cxx:1757): TransFailed E:0x20000050 (incorrect format)
12:59:34.574 IQV  D 394 (iqmark.cxx:72): MsgMark: queue: iqPOS_DATA
12:59:34.574 MSG  T 394 (iqmark.cxx:99): Event Track (Mark: beg): IQ [iqPOS_DATA] Msg [etNEW_POS_DATA] Mark
[0x00000000]
12:59:34.574 IPV  D 394 (iosocketclient.cxx:863): SendAppMessage: msg command: MARK, data: 568
12:59:34.574 IPV  T 394 (iosocketclient.cxx:942): SendAppMessage: Wrote 584 bytes
12:59:34.574 IPV  D 394 (iosocketclient.cxx:1121): WaitForAppMessage: msg len: 32 code: ACK  compressed: NO
12:59:34.574 MSG  T 394 (iqmark.cxx:194): Event Track (Mark: end): IQ [iqPOS_DATA] Msg [etNEW_POS_DATA] Mark
[0x00000000]
12:59:34.574 MSG  T 394 (workitems.cxx:560): *ROLLBACK* inbound message [etNEW_POS_DATA] seq major: 0 minor: 16
priority: 0 enqueue: 05/07/2000 11:58:32.574
12:59:34.574 APIV D 324 (defeventproc.cxx:91): DefaultEventProc: STCE_INMESSAGE_ROLLBACK (An inbound message has
been rolled back)
```

Failed Event publication

MSGP debug flag

The MSGP debug flag records the parsing and mapping of data into an Event Type Definition (ETD). This debug flag is a useful troubleshooting tool for e*Ways using Monk Collaborations. See the example below.

```
12:44:10.445 MSGP I 261 (monklog:406): MAPPED:
12:44:10.445 MSGP I 261 (monklog:406): undefined[0]:ON
12:44:10.445 MSGP I 261 (monklog:406): :1:1:
12:44:10.445 MSGP I 261 (monklog:406): :
12:44:10.445 MSGP I 261 (monklog:406): :0:
12:44:10.505 MSGP I 261 (monklog:406): (0)
12:44:10.505 MSGP I 261 (monklog:406): :
12:44:10.505 MSGP I 261 (monklog:406): "~
12:44:10.505 MSGP I 261 (monklog:406): "
12:44:10.515 MSGP I 261 (monklog:406):          -> "
12:44:10.515 MSGP I 261 (monklog:406): ".
12:44:10.515 MSGP I 261 (monklog:406): MAPPED:
12:44:10.515 MSGP I 261 (monklog:406): etdPOS2[0]:ON
12:44:10.515 MSGP I 261 (monklog:406): :1:1:
12:44:10.515 MSGP I 261 (monklog:406): :
12:44:10.515 MSGP I 261 (monklog:406): :-1:
12:44:10.515 MSGP I 261 (monklog:406): (-1)
12:44:10.525 MSGP I 261 (monklog:406):          -> "
12:44:10.525 MSGP I 261 (monklog:391): 01*00~02*8001*3000*08*susan~03*no*123abc>20.00>1*80.00~04*123abc>Tennis
Shoes~05
12:44:10.525 MSGP I 261 (monklog:406): *5~
12:44:10.525 MSGP I 261 (monklog:406): ".
```

REG debug flag

The REG flag records interaction with the e*Gate registry. This debug flag is useful in allowing you to observe the loading of schema-supporting files. See the example below.

```
12:44:08.392 REG  D 261 (utilities.cxx:825): file etdPOS.ssc found at monk_scripts\common
12:44:08.392 REG  D 261 (clientapis.cxx:2292): Retrieving file:
d:\egate\client\monk_scripts\common\etdPOS.ssc
```

4.4.3 Ten Most Common Monk Errors

This section presents examples of the ten most frequently encountered problems on the e*Gate system, along with suggestions on how to use the log files to find and fix the problems.

RESOLVE_VARIABLE error

If you do not load the dll properly, the **RESOLVE_VARIABLE** error on the **make-connection-handle** appears in the ODBC, Oracle, or Sybase e*Way.

```
17:52:53.429 MNK  E 1 (monklog:391):  >>L66:C48:"/home/egate/client/
monk_library/dart/db-stdver-eway-funcs.monk
17:52:53.430 MNK  E 1 (monklog:406): "
17:52:53.431 MNK  E 1 (monklog:406):
17:52:53.431 MNK  E 1 (monklog:406): (make-connection-handle)
17:52:53.432 MNK  E 1 (monklog:391): >>>>MONKEXCEPT:0036:
RESOLVE_VARIABLE: variable <make-connection-handle> has not
17:52:53.433 MNK  E 1 (monklog:406):  been defined.
```

This error (highlighted in red above) is caused when the dll was not loaded properly. Look further up in the log file for something like:

```
17:52:41.922 COM  D 1 (cdll.cxx:286): Loaded DLL /home/egate/client/
bin/stc_dbmonkext.dll: 0xFEAF17DC
17:52:41.923 CDBV D 1 (dgdb_monk.c:6476): init_stc_dbmonkext: calling
DBLoad for: [ORACLE8i]
17:52:41.926 CDB  E 1 (dgdbload.c:218): STCDB error: ld.so.1: /home/
egate/client/bin/stcewgenericmonk.exe: fatal: libclntsh.so.8.0: open
failed: No such file or directory
17:52:41.927 CDB  E 1 (dgdb_monk.c:6483): init_stc_dbmonkext: unable
to load: [ORACLE8i]
17:52:41.927 MNK  I 1 (monklog:406): Loaded stc_dbmonkext.dll
```

Possible solution

The message "No such file or directory" (above in red) indicates that the **.dll** was not loaded.

If you are running UNIX:

- Make sure that the Oracle library is set in the library path.
- Make sure that the Oracle library is not set to the wrong version of UNIX.
- Make sure that all the variables were set correctly. The number of variables you need to check depends upon which platform of UNIX you are using.

If you are running Windows:

- Make sure that the Client software is installed.

Unable to connect to e*Insight backend

```
14:51:45.963 EWY  I 347 (jvminit.cxx:1012): ewjx: Java VM CLASSPATH
set to:
C:\eGate\client\classes;C:\eGate\client\classes\xml.jar;C:\eGate\client\
classes\workflow.jar;C:\eGate\client\classes\stcjcs.jar
14:51:50.910 EWY  I 347 (jvminit.cxx:1046): ewjx: Successfully
instantiated a Java 2 VM
14:51:56.819 EWY  I 347 (svcmain.cxx:143): ServiceMain Started
14:52:01.405 EWY  W 347 (jvm_extension.cxx:233): Cannot connect to
eBPM Backend DataBase: oracle.jdbc.driver.OracleDriver
```

Check your **CLASSPATH** (highlighted in red above). Make sure that your **CLASSPATH**, which either is external or in the configuration file, includes the following:

C:\eGate\client\classes\egate.jar

C:\eGate\client\classes\classes12.zip as well as **C:\eGate\client\classes**

C:\eGate\client\classes\xml.jar

C:\eGate\client\classes\workflow.jar

C:\eGate\client\classes\stcjcs.jar

Batch e*Way

The Batch e*Way accepts the first Event sent to it, and rejects all subsequent Events. It also appears to report that it is disconnected from the external system.

```
[first event]
MNK  I 1680 (monklog:406): batch-ext-connect
MNK  I 1680 (monklog:406): ftp-ext-connect
MNK  I 1680 (monklog:406): ftp-ext-connect: allow connection is
FALSE
MNK  I 1680 (monklog:406): ftp-ext-connect: Connection deferred
MNK  I 1680 (monklog:406): batch-proc-out:
MNK  I 1680 (monklog:406): batch-regular-proc-out: Processing:
<record1>
MNK  I 1680 (monklog:406): batch-regular-proc-out: Must open a new
output file to write to.
MNK  I 1680 (monklog:406): batch-regular-proc-out: ftell = 0

[second event]
MNK  I 1680 (monklog:406): batch-ext-connect
MNK  I 1680 (monklog:406): ftp-ext-connect
MNK  I 1680 (monklog:406): ftp-ext-connect: allow connection is
FALSE
MNK  I 1680 (monklog:406): ftp-ext-connect: Connection deferred
MNK  I 1680 (monklog:406): batch-proc-out:
MNK  I 1680 (monklog:406): batch-regular-proc-out: Processing:
<record 2>
MNK  I 1680 (monklog:406): batch-regular-proc-out: Must open a new
output file to write to.
MNK  I 1680 (monklog:406): batch-regular-proc-out: ftell = 21
MNK  I 1680 (monklog:391): batch-regular-proc-out: Got new event
while record type is single record  Will c
MNK  I 1680 (monklog:406): all insert-exchange-data-event to send
out the last event
EWY  T 1680 (actions.cxx:182): batch-proc-out: returned [RESEND
...] length: 6
EWY  I 1680 (actions.cxx:223): batch-proc-out returned a RESEND.
Waiting for 10 seconds before resending.
```

Explanation

The sequence noted above occurs when the e*Way is configured to process “Single Record” Record types, and the given “Remote File Name” does not contain sequence-number qualifiers. That is, the e*Way is asked to transfer each outgoing Event (into separate destination files), but is not given a suitable formula for determining how to name these files distinctly. The e*Way's “process outgoing” script is written so that if this condition is encountered, all Events subsequent to the first are rejected, preventing them from being spooled along with the first Event. Only after the first Event has been successfully transferred to the external system, will the e*Way accept the next Event.

Rejected Events are rolled back into their originating queues; no data loss occurs. The resend indication “RESEND” is returned by the *process outgoing* script to signal the e*Way to try again (later).

Possible solution

This implementation of the e*Way is probably intentional. In the case that it is not, either provide a "Remote File Name" value with sequence-number qualifiers if the intent is to send individual Events as individual files, or use a "Delimited" Record Type if the intent is to transfer Events together in one file.

Siebel Event-driven e*Way

The Siebel Event-driven e*Way emits a **RESOLVE_VARIABLE** error, complaining that the function **co-create-instance** is not defined.

```
EWY  T 778 (schedules.cxx:193): processed schedule:
EWGM_SCHEDULE_CONNDOWN
EWY  T 778 (schedules.cxx:193): processed schedule:
EWGM_SCHEDULE_CONNUP
MNK  E 749 (monklog:391):
>>L10:C29:"C:\stc\eGate\client\monk_library\ewsiebeleventdriven/
siebel-eventdri
MNK  E 749 (monklog:406): ven-connect.monk"
MNK  E 749 (monklog:406):
MNK  E 749 (monklog:406): (co-create-instance
"SiebelDataServer.ApplicationObject")
MNK  E 749 (monklog:391): >>>>MONKEXCEPT:0036: RESOLVE_VARIABLE:
variable <co-create-instance> has not bee
MNK  E 749 (monklog:406): n defined.
MNK  E 749 (monklog:391):
>>L9:C9:"C:\stc\eGate\client\monk_library\ewsiebeleventdriven/
siebel-eventdrive
MNK  E 749 (monklog:406): n-connect.monk"
MNK  E 749 (monklog:406):
MNK  E 749 (monklog:406): (set! siebobj-handle (co-create-instance
"SiebelDataServer.ApplicationObject"))
MNK  E 749 (monklog:406):  >>Evaluation Of Dynamically Created
Expression.
MNK  E 749 (monklog:406):
MNK  E 749 (monklog:406): (siebel-eventdriven-connect)
EWY  A 749 (actions.cxx:827): MonkFailed
```

Explanation

The function **co-create-instance** is defined in **stc_monkcom.dll**. The logic implemented in the **siebel-eventdriven-init.monk** e*Way initialization script loads **stc_monkcom.dll** only if the e*Way is being used bidirectionally or in "Outbound Toward Siebel" mode, while **stc_monkdcom.dll** is loaded only if the e*Way is being used bidirectionally or in "Inbound From Siebel" mode.

Possible Solution

Modify the initialization script so that the **stc_monkcom.dll** is loaded unconditionally.

No Batch e*Way files are transferred

The Batch e*Way appears to be running with no errors and there are log indications that Events are being received and spooled, but no files are transferred.

```
EWY  T 1680 (actions.cxx:857): batch-ext-verify: returned [UP ...]
length: 2
EWY  T 1680 (svcmain.cxx:385): processed action:
EWGM_ACTION_CHECKCONN
EWY  T 1068 (schedules.cxx:247): Ignore EWGM_SCHEDULE_CONNDOWN -
because eWay state: Up External state: Up has check connection down
event: FALSEMNK  I 1680 (monklog:406): batch-ext-verify
MNK  I 1680 (monklog:406): ftp-ext-verify
EWY  T 1068 (schedules.cxx:259): processed schedule:
EWGM_SCHEDULE_CONNUP

EWY  T 1680 (actions.cxx:857): batch-ext-verify: returned [UP ...]
length: 2
EWY  T 1680 (svcmain.cxx:385): processed action:
EWGM_ACTION_CHECKCONN
EWY  T 1068 (schedules.cxx:259): processed schedule:
EWGM_SCHEDULE_CONNUP
EWY  T 1068 (schedules.cxx:247): Ignore EWGM_SCHEDULE_CONNDOWN -
because eWay state: Up External state: Up has check connection down
event: FALSE
MNK  I 1680 (monklog:406): batch-ext-verify
MNK  I 1680 (monklog:406): ftp-ext-verify

EWY  T 1680 (actions.cxx:857): batch-ext-verify: returned [UP ...]
length: 2
EWY  T 1680 (svcmain.cxx:385): processed action:
EWGM_ACTION_CHECKCONN
EWY  T 1068 (schedules.cxx:247): Ignore EWGM_SCHEDULE_CONNDOWN -
because eWay state: Up External state: Up has check connection down
event: FALSEMNK  I 1680 (monklog:406): batch-ext-verify
MNK  I 1680 (monklog:406): ftp-ext-verify
EWY  T 1068 (schedules.cxx:259): processed schedule:
EWGM_SCHEDULE_CONNUP
```

Explanation

The **Process Outgoing Message** function is responsible for receiving outbound Events from e*Gate and spooling them into temporary files. It is the **Exchange Data With External** function that performs the actual file transfer. This function is only triggered if a **Start Exchange Data Schedule** or **Exchange Data Interval** is specified for the e*Way. Without a schedule or interval, the e*Way will continue to spool/stage outbound Events, but none will be transferred to the external system.

Possible Solution

Edit the Communications Setup section of the Batch e*Way configuration to specify either a **Start Exchange Data Schedule** or an **Exchange Data Interval**, or both.

Note: *See the **Batch e*Way Intelligent Adapter User's Guide** for more information.*

Failure to map data to an ETD

The API flag (top red highlight below) records the failure of a data transformation within a Collaboration.

```
12:59:34.954 MNK  E 394 (monklog:406):
FAILED:InsufficientData:product_description:ON:1:1:::-1:(-1):">"-> "".
12:59:34.954 API  T 394 (translate.cxx:315): MonkTransLoadFailed
E:0x20000050 (incorrect format)
12:59:34.954 API  A 394 (workitems.cxx:1757): TransFailed E:0x20000050
(incorrect format)
12:59:34.954 IQV  D 394 (iqmark.cxx:72): MsgMark: queue: iqPOS_DATA
12:59:34.954 MSG  T 394 (iqmark.cxx:99): Event Track (Mark: beg): IQ
[iqPOS_DATA] Msg [etNEW_POS_DATA] Mark [0x00000000]
12:59:34.954 IPV  D 394 (iosocketclient.cxx:863): SendAppMessage: msg
command: MARK, data: 568
12:59:34.964 IPV  T 394 (iosocketclient.cxx:942): SendAppMessage: Wrote
584 bytes
12:59:34.964 IPV  D 394 (iosocketclient.cxx:1121): WaitForAppMessage:
msg len: 32 code: ACK  compressed: NO
12:59:34.964 MSG  T 394 (iqmark.cxx:194): Event Track (Mark: end): IQ
[iqPOS_DATA] Msg [etNEW_POS_DATA] Mark [0x00000000]
12:59:34.964 MSG  T 394 (workitems.cxx:560): *ROLLBACK* inbound message
[etNEW_POS_DATA] seq major: 0 minor: 16 priority: 0 enqueue: 05/07/2000
11:58:32.574
12:59:34.964 APIV D 324 (defeventproc.cxx:91): DefaultEventProc:
STCE_INMESSAGE_ROLLBACK (An inbound message has been rolled back)
```

Explanation

The culprit for this failure is incorrectly formatted data; that is, the received data did not conform to the appropriate ETD (see the second red highlight above). Note other related information indicated by the MNK and MSG flag entries. The result is a failed Event publication (see the lower red highlight above).

Possible Solution

Check your configuration for the correct format.

Invalid Monk code

The MNK flag (see the top red highlight) records invalid variables and function names in the Monk code.

```
(bad "string-ci=? ~input%etdPOS.PO1_item_data.promo? \"yes\")")
13:03:15.671 MNK  E 375 (monklog:406): >>>>MONKEXCEPT:0036:
RESOLVE_VARIABLE: variable <bad> has not been defined.
13:03:15.671 MNK  E 375 (monklog:406):  >>Evaluation Of Dynamically Created
Expression.
13:03:15.671 MNK  E 375 (monklog:406):
13:03:15.682 MNK  E 375 (monklog:391): (if (bad "string-ci=?
~input%etdPOS.PO1_item_data.promo? \"yes\")") (begin (iq-p
13:03:15.682 MNK  E 375 (monklog:406): ut "etPROMO_DATA" (get
~output%etdPOS) (list "etNEW_POS_DATA") 0 0 5)) (begin))
13:03:15.682 MNK  E 375 (monklog:406):  >>Evaluation Of Dynamically Created
Expression.
13:03:15.682 MNK  E 375 (monklog:406):
13:03:15.682 MNK  E 375 (monklog:391): (begin (copy-strip ~input%etdPOS
~output%etdPOS "") (if (bad "string-ci=? ~input
13:03:15.682 MNK  E 375 (monklog:391): %etdPOS.PO1_item_data.promo?
\"yes\")") (begin (iq-put "etPROMO_DATA" (get ~outp
13:03:15.682 MNK  E 375 (monklog:406): ut%etdPOS) (list "etNEW_POS_DATA") 0
0 5)) (begin)))
13:03:15.692 MSG  T 375 (iqmark.cxx:99): Event Track (Mark: beg): IQ
[iqPOS_DATA2] Msg [etNEW_POS_DATA] Mark [0x00000000]
13:03:15.692 IPV  D 375 (iosocketclient.cxx:863): SendAppMessage: msg
command: MARK, data: 592
13:03:15.692 IPV  T 375 (iosocketclient.cxx:942): SendAppMessage: Wrote 608
bytes
13:03:15.692 IPV  D 375 (iosocketclient.cxx:1121): WaitForAppMessage: msg
len: 32 code: ACK  compressed: NO
13:03:15.692 MSG  T 375 (iqmark.cxx:194): Event Track (Mark: end): IQ
[iqPOS_DATA2] Msg [etNEW_POS_DATA] Mark [0x00000000]
13:03:15.702 MSG  T 375 (workitems.cxx:560): *ROLLBACK* inbound message
[etNEW_POS_DATA] seq major: 0 minor: 13 priority: 0 enqueue: 05/07/2000
12:03:02.733
```

Explanation

Invalid variables (for example: the “variable <bad> has not been defined” string, which is highlighted in red at the top of the example) and function names in the Monk code are illegal Monk statements and stop the Event’s publication (see the bottom red highlight).

Possible Solution

Syntax errors in Monk code authored in the e*Gate Collaboration Rules Editor or the e*Gate ID Editor may be detected when the file is saved, or when the Editor is explicitly commanded to validate the code. The Editor takes syntactically incorrect code and places it within a (bad ...) enclosure.

For example, the following code lacks a missing open-parenthesis before the symbol "string-ci=?" [pseudo-GUI view]

```
IF string-ci=? ~input%etdPOS.PO1_item_data.promo? "yes")
    DISPLAY "yes"
ELSE
    DISPLAY "no"
```

The Editor will "wrap" this erroneous segment in a (bad ...) expression, when the file is saved [raw Monk file view]:

```
(if (bad "string-ci=? ~input%etdPOS.PO1_item_data.promo? \"yes\")")
    (begin
        (display "yes")
    )
    (begin
        (display "no")
    )
)
```

There is no actual "bad" function. It is used to prevent the erroneous expression from being evaluated and possibly causing unintentional and undesirable side-effects. Instead, Monk is induced to call the "bad" function; since "bad" is non-existent, the Monk response is to emit a **RESOLVE_VARIABLE** error. This reaction is intentionally elicited.

No configuration file specified for an e*Way

The "invalid parameter passed" entry (highlighted in red) and subsequent file trailer lines indicate the shutdown of the current e*Way.

```
17:18:02.323 API  D 250 (workitems.cxx:972): Work slice [<Inbound Group> - 1]
configuration loaded.  About to start processing
17:18:02.323 IPV  D 250 (iosocketclient.cxx:863): SendAppMessage: msg command:
CLSE, data: 0
17:18:02.333 IPV  T 250 (iosocketclient.cxx:942): SendAppMessage: Wrote 16 bytes
17:18:02.333 IPV  D 250 (iosocketclient.cxx:1121): WaitForAppMessage: msg len: 32
code: ACK  compressed: NO
17:18:02.333 IPV  T 250 (iosocketclient.cxx:141): Destructor
17:18:02.333 IPV  T 250 (iosocketclient.cxx:160): Close(): 212
17:18:02.333 COMV T 250 (iosocket.cxx:100): Destructor
17:18:02.333 COM  D 250 (cdll.cxx:240): Unloding DLL: 0x028B0000
17:18:02.343 API  T 250 (transsvc.cxx:270): InvalidParam E:0x20000002 (invalid
parameter passed)
17:18:02.343 COM  D 250 (cdll.cxx:240): Unloding DLL: 0x028C0000
17:18:02.603 CBV  D 258 (ncbtcp.c:161): Closing socket descriptor 228.
17:18:02.603 IPV  D 389 (iosocketclient.cxx:863): SendAppMessage: msg command:
rRLS, data: 0
17:18:02.653 IPV  T 389 (iosocketclient.cxx:942): SendAppMessage: Wrote 16 bytes
17:18:02.693 IPV  D 389 (iosocketclient.cxx:1121): WaitForAppMessage: msg len: 24
code: gACK compressed: NO
17:18:02.693 IPV  T 389 (iosocketclient.cxx:141): Destructor
17:18:02.693 IPV  T 389 (iosocketclient.cxx:160): Close(): 160
17:18:02.693 COMV T 389 (iosocket.cxx:100): Destructor
17:18:02.693 COMV D 389 (file.cxx:648): File_::Close(): Handle: 0x00000064
17:18:02.693 COMV D 389 (file.cxx:1260): File_::Unlock(): handle: 0x00000070
17:18:02.693 COMV D 389 (file.cxx:648): File_::Close(): Handle: 0x00000070
--------------------------------------------------------------------------------
[07-May-2000 17:18:02.703] *** END LOG FOR: ewPOS_IN2 on akhosravy pid: 246 ***
--------------------------------------------------------------------------------
```

Explanation

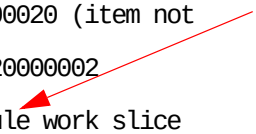
e*Gate encountered an invalid parameter in the e*Way, causing it to abort the command and shut down the e*Way.

Possible Solution

Edit the e*Way, correcting the invalid parameter.

No Collaboration specified for an e*Way or BOB

Note the strings “LoadWSFailed” (WS stands for work slice) and “ConfigLoadFailed” highlighted in red in the log example below.



```
17:21:57.280 REG  A 327 (clientapis.cxx:1779): ServerNak E:0x20000020 (item not
found)
17:21:57.280 API  T 327 (configload.cxx:1285): NoWorkslices E:0x20000002
(invalid parameter passed)
17:21:57.280 API  F 327 (configload.cxx:330): Unable to load module work slice
configuration. Please make sure the configuration, collaboration, or event type
for this component has been committed to the Registry.
17:21:57.290 API  A 327 (configload.cxx:349): LoadWSFailed E:0x000006B5 ((from
OS) The interface is unknown.)
17:21:57.351 API  A 327 (acquire.cxx:382): ConfigLoadFailed E:0x000006B5 ((from
OS) The interface is unknown.)
17:21:57.361 IPV  D 327 (iosocketclient.cxx:863): SendAppMessage: msg command:
rRLS, data: 0
17:21:57.371 IPV  T 327 (iosocketclient.cxx:942): SendAppMessage: Wrote 16 bytes
17:21:57.471 IPV  D 327 (iosocketclient.cxx:1121): WaitForAppMessage: msg len:
24 code: gACK compressed: NO
17:21:57.471 IPV  T 327 (iosocketclient.cxx:141): Destructor
17:21:57.471 IPV  T 327 (iosocketclient.cxx:160): Close(): 160
17:21:57.471 COMV T 327 (iosocket.cxx:100): Destructor
17:21:57.471 COMV D 327 (file.cxx:648): File_::Close(): Handle: 0x00000064
17:21:57.471 COMV D 327 (file.cxx:1260): File_::Unlock(): handle: 0x00000070
17:21:57.481 COMV D 327 (file.cxx:648): File_::Close(): Handle: 0x00000070
17:21:57.481 EWY  A 327 (svcmain.cxx:372): AcquireFailed E:0x20000020 (item not
found)
17:21:57.481 EWY  A 327 (svcmain.cxx:366): ServiceMain(): FAILED: 0x20000020
(item not found)
--------------------------------------------------------------------------------
[07-May-2000 17:21:57.481] *** END LOG FOR: ewPOS_IN2 on akhosravy pid: 332 ***
```

Explanation

These strings (highlighted in red above) point out that the interface is unknown, which could mean that there is no Collaboration or Event Type available. The subsequent file trailer lines document the shutdown of the e*Way.

Possible Solution

The error message, which is pointed out with the red arrow (text begins with “Unable to load module work slice ... “), tells you exactly what should be done: Make sure the configuration, Collaboration, or Event type for this component has been committed to the Registry.

Specified IQ is not connected to an IQ Manager

Upon attempting to publish the Event noted by the MSGV flag (see red highlight below), the IQ Manager records that the connection to the specified IQ has failed (see the three red highlights at the bottom of the example).

```
19:25:27.105 API  D 391 (workitems.cxx:972): Work slice [<Inbound Group> - 1]
configuration loaded.  About to start processing
19:25:27.115 MSGV T 391 (queuelayer.cxx:102): Putting message [etNEW_POS_DATA].
Body follows: Data Follows (bytes 122):
      30 31 2A 30 32 7E 30 32 2A 37 30 30 31 2A 32 30  |  01*02~02*7001*20
      30 33 2A 34 32 7E 30 33 2A 79 65 73 2A 31 32 33  |  03*42~03*yes*123
      61 62 63 3E 32 30 2E 30 30 3E 31 2A 37 38 39 78  |  abc>20.00>1*789x
      79 7A 3E 31 30 2E 30 30 3E 33 2A 63 61 73 68 2A  |  yz>10.00>3*cash*
      35 30 2E 30 30 7E 30 34 2A 31 32 33 61 62 63 3E  |  50.00~04*123abc>
      54 65 6E 6E 69 73 20 53 68 6F 65 73 3E 2A 37 38  |  Tennis Shoes>*78
      39 78 79 7A 3E 54 65 6E 6E 69 73 20 42 61 6C 6C  |  9xyz>Tennis Ball
      73 3E 2A 7E 30 35 2A 35 2A 7E                    |  s>*~05*5*~

*****
some entries deleted for brevity
*****
19:25:27.205 IPV  D 339 (iosocketclient.cxx:1121): WaitForAppMessage: msg len: 24
code: gACK compressed: NO
19:25:27.205 IPV  T 339 (iosocketclient.cxx:141): Destructor
19:25:27.205 IPV  T 339 (iosocketclient.cxx:160): Close(): 312
19:25:27.205 COMV T 339 (iosocket.cxx:100): Destructor
19:25:27.766 EWYV T 284 (polldirectory.cxx:519): NotFound E:0x20000020 (item not
found)
19:25:28.637 IP   A 391 (iosocketclient.cxx:796): connectFailed E:0x0000274D
(unknown or system failure)
19:25:28.637 IQ   A 391 (iqinitialize.cxx:320): UnableToConnect E:0x20050000
(connection failed)
19:25:28.637 IQ   T 391 (send.cxx:199): NoServer E:0x20010000 (connection lost)
19:25:28.637 IQ   A 391 (iq.cxx:830): IQMsgPutFailed E:0x20010000 (connection
lost)
```

Explanation

This can occur when the e*Way loses the connection, or is unable to connect with the IQ Manager.

Possible Solution

Event format may play a role sometimes, but this can be network related, too. If it is a connection problem, we do not have a uniform solution to provide.

4.4.4 Five Common Java Errors

This section presents examples of the five frequently encountered Java problems on the e*Gate system, along with suggestions on how to use the log files to find and fix the problems.

- **"Missing ETD .jar file" on page 63**
- **"Missing custom ETD .ssc file" on page 64**
- **"Sending large messages without increasing the heap size" on page 66**
- **"Making changes to an ETD and not recompiling the Collaboration" on page 67**
- **"e*Way Connection does not have a configuration file" on page 68**

Missing ETD .jar file

The "Java collaboration initialize() method" (the first red highlight below) threw an exception. The following error message, "java.lang.NoClassDefFoundError" (the second red highlight below), indicates that no class definition was found, which caused the initializaton of a class to fail.

```
13:47:23.146 COLV T 8428 (initialize.cxx:234): JCS: Check if an
exception occurred calling Java Collaboration initialize() method ...
13:47:23.156 COL  I 8428 (initialize.cxx:258): JCS: Java collaboration
initialize() method threw a java.lang.Exception!
13:47:23.156 COL  I 8428 (java_extensions.cxx:934): ***************
Exception occurred **************
13:47:23.166 COL  I 8428 (java_extensions.cxx:934):  toString:
java.lang.NoClassDefFoundError: com/stc/test/EmpRoot
13:47:23.166 COL  I 8428 (java_extensions.cxx:934):
13:47:23.176 COL  I 8428 (java_extensions.cxx:934):  getMessage: com/
stc/test/EmpRoot
13:47:23.186 COL  I 8428 (java_extensions.cxx:934):
13:47:23.196 COL  I 8428 (java_extensions.cxx:934):  StackTrace:
java.lang.NoClassDefFoundError: com/stc/test/EmpRoot
    at java.lang.Class.forName0(Native Method)
    at java.lang.Class.forName(Unknown Source)
    at
com.stc.common.collabService.JCCollabControllerImpl.initialize(JCColla
bControllerImpl.java:335)

13:47:23.206 COL  I 8428 (java_extensions.cxx:934):
**********************************************
13:47:23.216 COL  F 8428 (initialize.cxx:1107): Initialization of the
CollabController class failed
13:47:23.216 COL  A 8428 (initialize.cxx:1456): InitUserClassFailed
E:0x000003E5 ((from OS) Overlapped I/O operation is in progress.
)
13:47:23.246 COL  I 8428 (initialize.cxx:1439): JCS: Detached current
thread from Java 2 VM; 0 attached threads remaining
13:47:23.256 COL  A 8428 (terminate.cxx:200): GetDataFailed
E:0x20000002 (invalid parameter passed)
```

Explanation

When you compile and save an ETD, a **.jar** file (Java archive file) is created (its file name is the same as the **.xsc** file name). The **.jar** file contains **.java** (Java source code) and **.class** (executable Java bytestream) files whose names correspond to the root node names in the ETD. If the **.jar** file was not committed to the proper directory when the ETD was compiled, an error message is created.

Possible solution

Recompile the ETD, making sure that the **.jar** file is committed to the right location under **\eGate\Server**:

1. In the e*Gate Integrator Enterprise Manager, open the Java ETD Editor.
2. Choose **File** on the menu bar and then click **Open** to open the ETD.
3. When the ETD is open, choose **File** on the menu bar and then click **Compile And Save**.
4. Check the **Error** dialog box; if no errors are listed, the compilation succeeded without a problem.

Note: *For additional information see the* ***e*Gate Integrator User's Guide****.*

Missing custom ETD .ssc file

The "java.lang.RuntimeException" message (the first red highlight below) alerts you to look for an error message. In this example, the Java translation failed when the Collaboration attempted to translate a method. The "C:\EGATE\Client\etd\EventTypeDefinition1.ssc" not readable string informs you where the error occurred.

```
13:49:34.360 COL  I 14388 (java_extensions.cxx:934): ***************
Exception occurred **************
13:49:34.370 COL  I 14388 (java_extensions.cxx:934):  toString:
com.stc.common.collabService.CollabConnException:
JCCollabControllerImpl (translate) Errorin executing translate method

3:49:34.390 COL  I 14388 (java_extensions.cxx:934):
13:49:34.390 COL  I 14388 (java_extensions.cxx:934):  getMessage:
JCCollabControllerImpl (translate) Error   in executing translate method

3:49:34.400 COL  I 14388 (java_extensions.cxx:934):
13:49:34.420 COL  I 14388 (java_extensions.cxx:934):  StackTrace:
java.lang.RuntimeException: Loading
"C:\EGATE\Client\etd\EventTypeDefinition1.ssc" failed:
MONKEXCEPT:0091: MONK_I_find_file: file
"C:\EGATE\Client\etd\EventTypeDefinition1.ssc" not readable.
    at com.stc.jcsre.ssc.Mocca.load(Mocca.java:102)
    at com.stc.jcsre.ssc.IGlassImpl.factory(IGlassImpl.java:170)
    at .
.
.
.
13:49:34.510 COL  I 14388 (java_extensions.cxx:934):
**********************************************
13:49:34.520 COL  T 14388 (translate.cxx:1513):
***ExceptionOccurred****
13:49:34.530 COL  T 14388 (translate.cxx:1722): JavaTranslateFailed
E:0x20000010 (item exists)
```

Explanation

The SeeBeyond ETD Editor converts existing Monk ETDs (**.ssc** files) to Java-enabled ETDs (**.xsc** files) using the SSC Wizard. If the conversion of the Monk **.ssc** file to this custom ETD **.ssc** file and its subsequent commitment to the correct directory when the ETD (**.xsc** file) is compiled fails, an error log with messages similar to those in this example are generated.

Possible solution

Make sure that the custom ETD **.ssc** file has been committed to the correct location under **\eGate\Server**.

1 In the e*Gate Integrator Enterprise Manager, open the Java ETD Editor.

Note: *For additional information see the* ***e*Gate Integrator User's Guide****.*

2 Choose **File** on the menu bar and then click **New** to display the ETD Builder Wizards.

3 Double-click the **SSC Wizard**.

4 Follow the on-screen instructions to locate the **.ssc** file and then convert it into an **.xsc** file.

5 When the file opens in the ETD Editor, modify it as needed.

6 Choose **File** on the menu bar and then click **Compile And Save**. Compiling the ETD generates **.xsc**, **.ssc**, and **.jar** files containing the changes you made using the ETD Editor.

7 Check the **Error** dialog box; if no errors are listed, the compilation succeeded without a problem.

Sending large messages without increasing the heap size

The "java.lang.Exception" string indicates an error occurred, and the "JavaTranslateFailed" string identifies the error as a translation error (see the first and last red highlights below). In this example, the Collaboration was created correctly. The reason why the message failed to be delivered is because the system ran out of memory. When this happens, a "java.lang.OutOfMemoryError" error is generated.

Note: *There could be many reasons for the "OutOfMemoryError" message, such as a large message (for example: 10 MB).*

```
14:07:25.923 COLV T 8932 (initialize.cxx:234): JCS: Check if an
exception occurred calling Java Collaboration translate ...
14:07:25.933 COL  I 8932 (initialize.cxx:258): JCS: Java collaboration
translate threw a java.lang.Exception!
14:07:25.943 COL  I 8932 (java_extensions.cxx:934): ***************
Exception occurred **************
14:07:25.953 COL  I 8932 (java_extensions.cxx:934):  toString:
java.lang.OutOfMemoryError
14:07:25.963 COL  I 8932 (java_extensions.cxx:934):
14:07:25.963 COL  I 8932 (java_extensions.cxx:934):  getMessage: null
14:07:25.973 COL  I 8932 (java_extensions.cxx:934):
14:07:25.983 COL  I 8932 (java_extensions.cxx:934):  StackTrace:
java.lang.OutOfMemoryError
     <<no stack trace available>>

14:07:25.993 COL  I 8932 (java_extensions.cxx:934):
*****************************************************
14:07:26.003 COL  T 8932 (translate.cxx:1513): ***ExceptionOccurred***
14:07:26.003 COL  T 8932 (translate.cxx:1722): JavaTranslateFailed
E:0x20000010 (item exists)
```

Explanation

Multi-Mode e*Ways allow you to set the initial and maximum heap size of reserved memory that e*Gate uses for the temporary storage of data structures. When set, the heap allows e*Gate to request free memory, access it, and then free it up when finished using it. If the maximum amount of heap memory has been set too low, you can run out of memory while processing.

Possible solutions

Increase the maximum heap size in the Multi-Mode e*Way configuration file:

1. Open the e*Gate Integrator Enterprise Manager.
2. From the Navigator pane **Components** view, select the Multi-Mode e*Way and click the **Properties** icon on the toolbar.
3. On the **General** tab of the **e*Way - Multi-Mode Properties** dialog box, click **Edit** under **Configuration file**.
4. Scroll to **Maximum Heap Size** and increase the size.
5. Choose **File** from the menu bar and then click **Save**.
6. Close the **Edit Settings** dialog box and then close the **e*Way - Multi-Mode Properties** dialog box.

Or

Increase the heap size using the **-mx** option in the Collaboration Rule initialization string:

1. Open the e*Gate Integrator Enterprise Manager.
2. From the Navigator pane **Components** view, select the **Collaboration Rules** folder.
3. In the Editor pane, select the Collaboration Rule that you want to edit, and then click the **Properties** icon on the toolbar.
4. Under the **General** tab of the **Collaboration Rules Properties** dialog box, add **-mx** ***<amount of required space>*** to the **Initialization string** box, taking care not to change any of the settings currently listed in the box.
5. When finished, click **OK**.

Making changes to an ETD and not recompiling the Collaboration

The "java.lang.Exception" string indicates an error occurred, and the "JavaTranslateFailed" string identifies the error as a translation error (see the first and last red highlights below). In this example, the "java.lang.NoSuchMethodError" means that the Collaboration looked for a method that did not exist.

```
14:19:48.981 COLV T 5440 (initialize.cxx:234): JCS: Check if an
exception occurred calling Java Collaboration translate ...
14:19:48.991 COL  I 5440 (initialize.cxx:258): JCS: Java collaboration
translate threw a java.lang.Exception!
14:19:49.001 COL  I 5440 (java_extensions.cxx:934): ***************
Exception occurred **************
14:19:49.142 COL  I 5440 (java_extensions.cxx:934):  toString:
java.lang.NoSuchMethodError
14:19:49.152 COL  I 5440 (java_extensions.cxx:934):
14:19:49.162 COL  I 5440 (java_extensions.cxx:934):  getMessage: null
14:19:49.162 COL  I 5440 (java_extensions.cxx:934):
14:19:49.172 COL  I 5440 (java_extensions.cxx:934):  StackTrace:
java.lang.NoSuchMethodError
    at newcollab.executeBusinessRules(newcollab.java:55)
    at com.stc.jcsre.JCollaboration.translate(JCollaboration.java:133
    at
com.stc.common.collabService.JCCollabControllerImpl.translate(JCCollab
ControllerImpl.java:591)

14:19:49.182 COL  I 5440 (java_extensions.cxx:934):
**********************************************
14:19:49.192 COL  T 5440 (translate.cxx:1513): ***ExceptionOccurred***
14:19:49.202 COL  T 5440 (translate.cxx:1722): JavaTranslateFailed
E:0x20000010 (item exists)
```

Explanation

If changes are made to an ETD (such as deleting a node; the reason for this error message), you must recompile the Collaboration to correctly identify the method. If the Collaboration is not recompiled, the Java translation will fail because the method information originally generated has changed and is no longer valid.

Possible solution

Make sure that you recompile the Collaboration after making changes to an associated ETD.

1. In the e*Gate Integrator Enterprise Manager, open the Collaboration Rule in the Java Collaboration Rules Editor.
2. Choose **File** on the menu bar and then click **Compile**.
3. Check the Compile pane to see if the compilation succeeded without a problem.

Note: *For additional information see the* ***e*Gate Integrator User's Guide****.*

e*Way Connection does not have a configuration file

For this error, the log file is explicit in describing the problem, "ConnectionPoint Configuration File is blank for ewayCon" (first red highlight below), meaning there is no configuration file. When an e*Way Connection is missing its configuration file, you will see this error.

```
14:25:54.411 COL  T 6660 (configuration.cxx:477): JCS:
AddToJCollabInstanceMaps was successful for out
14:25:54.421 COLV T 6660 (configuration.cxx:481): AFTER Calling
addToJCollabInstanceMaps.
14:25:54.431 COL  T 6660 (configuration.cxx:320): JCS: Setup the
CollabConnectionPoint Objects.
14:25:54.441 COL  W 6660 (configuration.cxx:267): ConnectionPoint
Configuration File is blank for ewayCon
14:25:54.451 COL  W 6660 (configuration.cxx:836): Error in
SetupConnectionPointObjects name for
14:25:54.461 COL  T 6660 (translate.cxx:1724): JavaTransLoadFailed
E:0x20000010 (item exists)
```

Explanation

Every e*Way Connection must have a configuration file.

Possible solution

Create a configuration file for the e*Way Connection:

1. Open the e*Gate Integrator Enterprise Manager.
2. From the Navigator pane **Components** view, select the **e*Way Connections** folder.
3. In the Editor pane, select the e*Way Connection that you want to edit, and then click the **Properties** icon on the toolbar.
4. On the **General** tab of the **e*Way Connection Properties** dialog box, click **New** under **e*Way Connection Configuration File**. The **Edit Settings** dialog box appears.

Note: *For JMS e*Way Connection configuration parameters, see the* ***SeeBeyond JMS Intelligent Queue User's Guide*** *for information on what to enter. For all other e*Way Connection parameters, see the e*Way user's guide that is specific to the external system to which you are connecting for information on what to enter.*

5 When finished setting the configuration parameters, save them by choosing **File** on the menu bar and then clicking **Save As**.

6 Close the **Edit Settings** dialog box and then close the **e*Way Connection Properties** dialog box.

4.5 Viewing Debug Tracing

You can view debug tracing at the same time you open and log on to the Enterprise Manager window. This feature allows you to view debugging trace code in a DOS Console window that opens along with the Enterprise Manager. To use the feature, you must set specific flags in the property sheet for the Enterprise Manager.

To view debug tracing

1 With the mouse, right-click on the **Enterprise Manager** icon on your desktop and select the **Properties** menu option.

The **e*Gate Enterprise Manager** property sheet appears.

2 Click the **Shortcut** tab. By default, the **Target** text box displays the following text (if you used the default installation path on drive C):

```
C:\eGate\client\bin\stcguistart.exe  -ra "-nv -lc en_US"  -ja
 "-mx64m"  -rc com.stc.egate.MainApp  -jre "C:\Program
Files\JavaSoft\JRE\1.1\bin\jrew.exe"
```

Note: *If you used a different installation path, that path displays in the first line of the text shown above.*

3 Change the text after the path statement and **stcguistart.exe** to read as follows:

```
-ra "-d -nv -lc en_US"  -ja "-mx64m"  -rc com.stc.egate.MainApp
  -jre "C:\Program Files\JavaSoft\JRE\1.1\bin\jre.exe"
```

Be sure to add the **-d** flag after the first double quotation mark and, at the end of the line, change **jrew.exe** to **jre.exe**. Take care to make *both* changes.

4 Run the Enterprise Manager.

The Enterprise Manager window opens and, in addition, a DOS Console window opens, allowing you to view debug operations. To disable this feature the next time you run the Enterprise Manager, return the **Target** text box to its default text.

Caution: *If you close the DOS Console window while you are using this feature, you automatically exit the Enterprise Manager. You receive no warning.*

Chapter 5

e*Gate Alert and Notification System

This chapter introduces how the e*Gate system handles monitoring and notification Events. It then explains e*Gate notifications, their codes, and Alert/error messages as displayed in the e*Gate Monitor. Finally, it provides troubleshooting tables based on the Alert messages.

Note: *Use Alert notification messages and the information they contain to troubleshoot your e*Gate system.*

5.1 Overview

The following information is provided in this chapter:

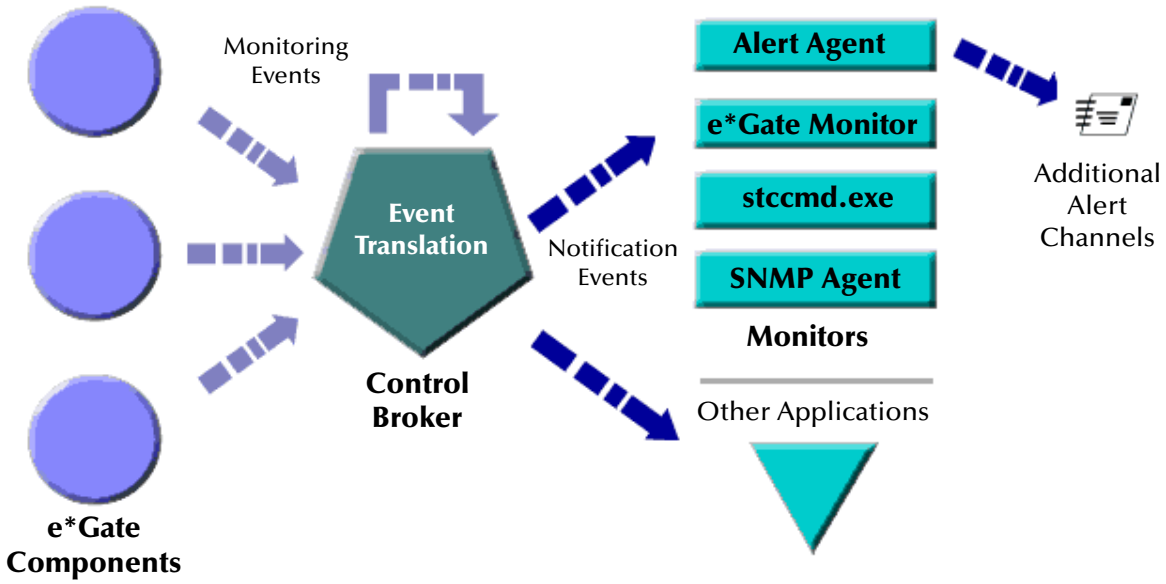
- A description of the notification process.
- Information on developing a notification strategy.
- Examples of monitoring- and notification-Event definitions.
- Detailed tables that define Alert notification codes and how to use them.
- Troubleshooting tables that list (by component or feature) the Alert messages, the problems they indicate, and suggested actions to fix the problems.

This chapter explains

5.2 The Notification Process

Notification is a chain of Events that begins with a component reporting its status to the Control Broker and ends with that Alert or status information being sent to a user. Figure 9 illustrates the notification process.

Figure 9 Notification Process



Basic steps

The notification process involves the following basic steps:

1. Components issue monitoring Events

The modules within the e*Gate system, for example, e*Way Intelligent Adapters, BOBs, and IQ Managers, send monitoring Events to the Control Broker.

These Events range from simple status messages, such as "component responding" or "data with a content of interest" to errors such as "component down" to "disk space threshold exceeded." Monitoring Events contain not just the notification code and description, but other information including the time the monitoring Event occurred and the element that generated the Event.

In addition to the monitoring Events other components send, the Control Broker can submit its own monitoring Events and Collaboration Rules scripts can issue monitoring Events using the Monk function **event-send**.

Note: *The **event-send** function can issue standard monitoring Events or custom Events that are specific to a user-developed component; see* **"Using the event-send function" on page 87** *for more information.)*

2. Monitoring Events become notifications

The Control Broker translates the monitoring Events into Alert and status notifications, which combine the monitoring Event with routing information that is used to direct the information to its intended destination.

3. Notifications are sent to monitors

After creating notification Events (Alert and status notifications), the Control Broker sends them to interactive monitors (the graphical e*Gate Monitor and the text-based **stccmd** utility; see **Appendix B** for information on how to read these notifications), SNMP agents that forward notifications to an SNMP Event manager, and Alert agents that can deliver notifications via any of the following channels:

Notification Channels

- e-mail
- Printer

Note: *Some of these notification channels are not supported on certain platforms. See the **e*Gate Integrator Alert Agent User's Guide** for details.*

Alert and status notifications can also be sent to other applications through application program interface (API) calls. The Control Broker can also execute a command script instead of sending a notification to a monitor.

If the Control Broker stops functioning, each component will queue its outbound Alert and status notifications until the Control Broker resumes normal functioning or the component itself is no longer responding (in which case the unsent notification Events will be lost).

Escalation

e*Gate also provides a means to escalate unresolved monitoring Events, so that system managers can apply additional support resources as necessary.

Any monitoring Event can be escalated if it is *resolvable*. Generally speaking, resolvable Events describe conditions that must be addressed. For example, a "component down" monitoring Event indicates that a component has stopped functioning; the condition can be "resolved" by bringing the component back up.

e*Gate can resolve notification Events under either of these conditions:

- A monitoring Event may be "resolvable" by its nature. For example, the condition that causes the Event "component down" can be resolved by restarting a component, which in turn generates a "component up" status notification. Because the condition "component up" does not require resolution, it is considered not resolvable. Monitoring Events that are "resolvable by nature" include the codes of their resolving notification Events.
- Any monitoring Event that is both designated an "Alert" and that is sent to an interactive monitor can be resolved, even if the Event would normally be considered unresolvable. For example, if a "component up" status notification is sent to the Alert tab of the e*Gate monitor, the Event can be resolved because the Alert tab provides a "resolved" check box (see **Figure 5 on page 28**)—even though, practically speaking, this condition requires no resolution.

All resolvable monitoring Events are automatically escalated at a user-definable interval. You can define the action to be taken at each level of escalation and configure the interval at which an Event moves from one level to another. For example, the first notification of a given Event can be sent to monitors, the second to a system operator's e-mail address, and the third to a supervisor's printer.

5.3 Developing and Implementing Notification Strategies

The notification strategies you develop will reflect the structure of your support organization. Here are just a few ideas for routing notifications:

- **By Module**. Route e*Way notifications to one person, Control Broker notifications to another.
- **By Severity**. Route fatal errors to one person, performance issues to another.
- **By Subsystem or Cause**. Route notifications based on the content of the element name, reason name, Alert or status notification name, or other fields in the notification Event (for example, to direct SAP-related Events to a SAP specialist).

Note: *If your notification strategy relies on component names, remember that any time you change a component's name, you must adjust your Notification Routing script accordingly. The rename feature in the Enterprise Manager will not automatically update Collaboration Rules scripts that refer to component names.*

5.3.1 Collaboration Rules Script

To implement your notification strategy, you need to devise a Collaboration Rules script that does the following:

- Uses the contents of the monitoring Event to determine the nature or type of the Event.
- Uses that information to determine the content and recipients of the notification.

e*Gate supplies a Collaboration called **NotificationRouting** to perform these functions. By default, this Collaboration uses the script file:

```
\monk_scripts\common\Notification.tsc
```

The file's path is relative to the root of the schema, not the e*Gate directory structure on disk. You can view or edit the default **Notification.tsc** shipped with e*Gate using the Collaboration Editor.

5.3.2 Notification Strategy Guidelines

Follow these general steps to develop and implement a notification strategy:

1. Determine the conditions for which you want to receive notifications.
2. Identify the information you want to include in each notification. For example, an Alert notification might include the error message, description, and time the error occurred.
3. Develop the logic to test for those conditions. Refer to **Table 7 on page 77** for the elements of the monitoring Event that you may wish to examine.
4. Identify the channels through which you want to receive notifications and the recipients of those notifications for each channel. At this point, you should also design your escalation strategy, determining which users should receive notification first, which second, and so on. This recipient information will be copied to the appropriate nodes within the notification Event structure (see **"Notification-Event Definition" on page 80** for more information).
5. Modify the **NotificationRouting** script to implement the strategy you have designed. You can use the default **Notification.tsc** script supplied with e*Gate as a basis for your own script.

Note: *We recommend you make a copy of the default* ***Notification.tsc*** *file (using the Collaboration Editor's* ***Save As*** *menu option) before you make any site-specific changes, to preserve the original copy and provide a fall-back as you develop your site-specific script. If there are* ***any*** *errors in the Notification script, no notifications of any kind will be sent.*

The next two sections discuss the anatomy of the monitoring Event and notification Event definitions. See **"Creating Custom Notifications" on page 86** for more information.

5.4 Monitoring-Event Definition

Monitoring Events have two major components: **EventHeader** and **EventBody**.

Note: *The figures in the following sections are taken from the* ***Notification.tsc*** *Collaboration Rules script. Within each section, tables explain the content of each node within the relevant sections of the Event definition. For more information about Event definitions, see the e*Gate user's guides. For more information about the Collaboration Rules Editors or Event Type Definition Editors, see the* ***e*Gate Integrator User's Guide*** *or each editor's Help system.*

The EventHeader section begins on the next page; EventBody is discussed beginning on **"EventBody" on page 79**.

Caution: *Nodes of data type "string" have no functional limit imposed on the string length. However, components that rely on the contents of the string may have their own format or length restrictions. For example, fields requiring e-Mail addresses must contain valid e-Mail addresses for Agents receiving notifications to be able to act upon those addresses; however, no format- or length-checking is performed within monitoring Event or notification definitions.*

5.4.1 About Retries and Attempts

Two channels (pager and phone) provide the means to retry notification delivery if initial attempts fail. You can set the following parameters for each channel:

- The number of **attempts** the system will make to deliver the notification, and the time between attempts.
- For each attempt, you can specify the number of **retries**, and the time between those retries.

For example, if you specify two attempts and four retries, the system would make the following delivery attempts:

Attempt #1
Retry 1
Retry 2
Retry 3
Retry 4

Attempt #2
Retry 1
Retry 2
Retry 3
Retry 4

We recommend you begin with a simpler configuration that specifies some number of attempts with *one* retry, and modify these parameters based on your site's requirements.

5.4.2 EventHeader

The EventHeader (shown in **Figure 10 on page 76**) contains basic information about the monitoring Event. The figure shows the Event Type Definition (ETD) as it would appear in the e*Gate Enterprise Manager's ETD Editor. The parts of the Event are called nodes.

Figure 10 EventHeader nodes

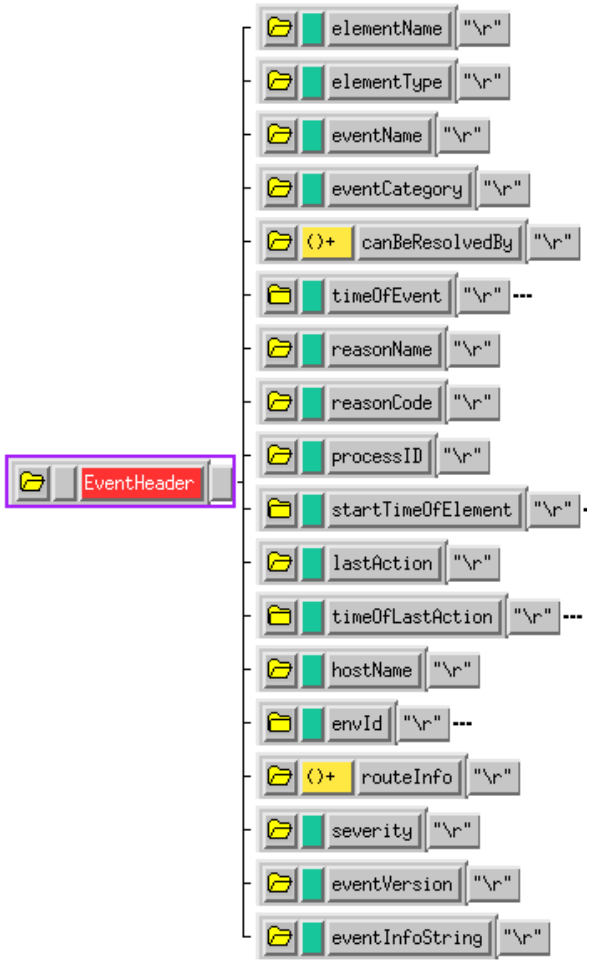


Table 7 describes the EventHeader nodes.

Table 7 EventHeader Node Elements

Element Name	Purpose	Data Type	Subnodes/Purpose
elementName	Name of the e*Gate system component that generated the monitoring Event, or the component on whose behalf the Control Broker generated the error.	String	
elementType	Type of the e*Gate system component that generated the monitoring Event (for example, a BOB, a Control Broker, and so on).	Character	
eventName	Name of the monitoring Event (for example, "Alert Agent cannot connect to Control Broker").	String	
eventCategory	Category of the monitoring Event (for example, is it a resource or performance problem). See "byte 3" codes in **Table 17 on page 89** for a list of valid Event categories.	String	
eventCode	Monitoring-Event code. See **Table 17 on page 89** for more information on code syntax.	String	
resolves	Monitoring-Event code(s) that this monitoring Event resolves.	Semicolon-delimited string	
canBe ResolvedBy	Monitoring-Event code that can resolve this monitoring Event.	Semicolon-delimited string	
timeOfEvent	Time the monitoring Event occurred.	Hex string	julianDateStamp; see **"Notes on julianDateStamp and timeStamp nodes" on page 79**.
		Hex string	timeStamp; see **"Notes on julianDateStamp and timeStamp nodes" on page 79**.
reasonName	Reason that the monitoring Event occurred.	String	
reasonCode	Code corresponding to the monitoring- Event reason.	String	
processID	ID of the process that generated the monitoring Event (supplied by the operating system).	Hex string	

Table 7 EventHeader Node Elements (Continued)

Element Name	Purpose	Data Type	Subnodes/Purpose
startTime OfElement	Time that the component that generated this code began operating.	Hex string	julianDateStamp; see **"Notes on julianDateStamp and timeStamp nodes" on page 79**.
		Hex string	timeStamp; see **"Notes on julianDateStamp and timeStamp nodes" on page 79**.
lastAction	Last action this component performed before generating the error.	String	
timeOf LastAction	Time that the last action occurred.	Hex string	julianDateStamp; see **"Notes on julianDateStamp and timeStamp nodes" on page 79**.
		Hex string	timeStamp; see **"Notes on julianDateStamp and timeStamp nodes" on page 79**.
hostName	Name of the Participating Host on which the monitoring Event-generating element was running.	String	
envID			SharedDataDir/ Stores working data (see "File Locations (**.egate.store**)" in the *e*Gate Integrator User's Guide*).
			controlPort/ TCP/IP Port shared by Control Broker and component issuing the monitoring Event.
routeInfo	A semi-colon-delimited list of other e*Gate elements that may be affected by the monitoring Event, as defined within the schema being monitored.	String	

Table 7 EventHeader Node Elements (Continued)

Element Name	Purpose	Data Type	Subnodes/Purpose
severity	Severity code for this monitoring Event: D (debugging), I (information), E (error), W (warning), F (fatal), or blank.	String	
eventVersion	Reserved for SeeBeyond use: do not alter this field. Default value: 1.0	String	
eventInfo String	Reserved for user agents or other user-developed applications using SeeBeyond's API to create monitoring Events that use this field.	String	

Notes on julianDateStamp and timeStamp nodes

The julianDateStamp node is a Julian date in hex format. The timeStamp node is a hex representation of a 32-bit integer that uses the following format:

```
Bits:  01234567890123456789012345678901
Use:    |  hr|| min|| sec||millisec||unused
      (0-24)
```

Note: *The time is Greenwich mean time; not local time.*

5.4.3 EventBody

The EventBody node is a repeating set of eventDetail nodes, each of which contains a SectionName and a Label/Content pair.

You only need to use this node if you want to extend messaging capabilities beyond those provided by the EventHeader using your own messaging applications (see Figure 11).

Figure 11 EventBody nodes

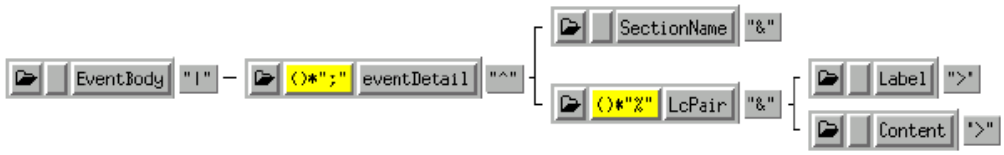


Table 8 on page 80 lists all the components of the EventBody node.

Table 8 EventBody Node Elements

Element Name	Purpose	Subnodes	Purpose
SectionName	The name of the section in the message. An optional field with which you can sort or filter Label/Content pairs.		
LcPair		Label	Field label
		Content	Field content

5.5 Notification-Event Definition

The notification Event describes the channels by which information from a monitoring Event is sent to its eventual destination.

The notification Event is defined by the NotificationMessage ETD, which contains two main nodes, the NotificationHeader and the NotificationBody (see Figure 12).

Figure 12 NotificationMessage nodes

The first major node, NotificationHeader, is discussed below, and NotificationBody is discussed in the section **"NotificationBody" on page 81**.

5.5.1 NotificationHeader

The NotificationHeader node contains information describing the notification itself (see Figure 13).

Figure 13 NotificationHeader nodes

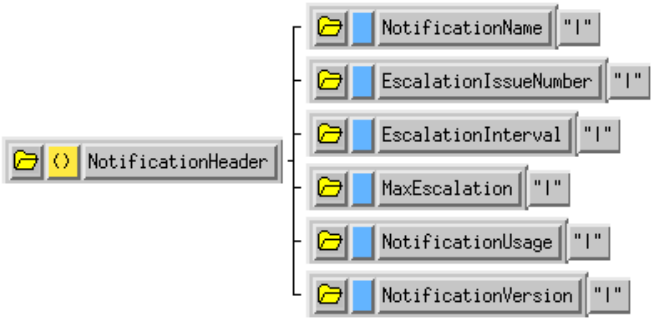


Table 9 describes the NotificationHeader nodes.

Table 9 NotificationHeader node elements

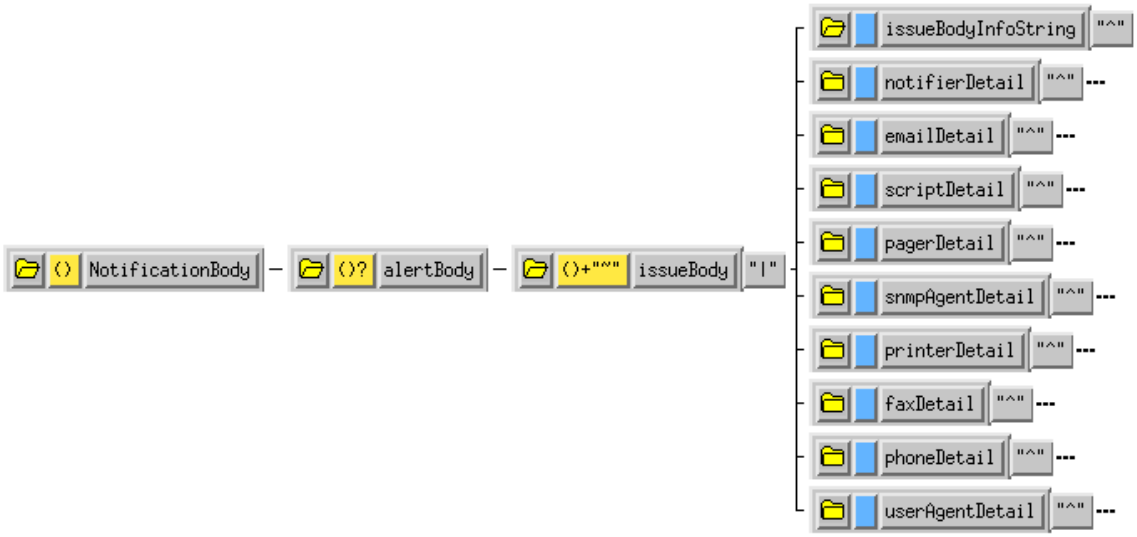
Element Name	Purpose	Required Value (if any)
NotificationName	Provides a means to add your own custom string to this notification.	N/A
EscalationIssueNumber	The initial escalation number; reserved for future use.	No specific value required, but a non-negative integer must be specified. Use the default (0) unless otherwise instructed.
EscalationInterval	Number of seconds to wait between escalations.	No specific value required, but a positive non-zero value must be specified.
MaxEscalation	Reserved for SeeBeyond use.	*Do not use.*
NotificationUsage	Describes whether the monitoring Event is a status message or alert (required by SeeBeyond's monitoring applications).	One of four values: **A, ALERT, S,** or **STATUS**. Alert messages are displayed on the e*Gate Monitor's Alert tab; status messages are displayed on the Status tab.
NotificationVersion	Provides a means to add your own revision number to this notification.	N/A

5.5.2 NotificationBody

The NotificationBody node contains ten subnodes. The first, **issueBodyInfoString**, can be used as a free-form text area that contains information about the nth *issuance* (escalation) of the notification being sent.

Only Alert messages (with a NotificationUsage set to "A" or "ALERT"—see **"NotificationHeader" on page 80**) can have more than one copy of NotificationBody defined per notification; status messages (with a NotificationUsage set to "S" or "STATUS") must have only one (see Figure 14).

Figure 14 NotificationBody nodes



The remaining NotificationBody elements describe channels for message delivery (see **"Notification Channels" on page 72** and **"Customizing Default Notification Routing" on page 154**). Some channels require Agents to deliver the notification through the channel to its final recipient; see the *e*Gate Integrator Alert Agent User's Guide* for more information.

5.5.3 notifierDetail

The notifierDetail node contains information required to direct Alert and status notifications to SeeBeyond's interactive monitoring applications (see Figure 15). For more information, see **"Using the e*Gate Monitor" on page 25**.

Figure 15 notifierDetail nodes

Table 10 describes notifierDetail nodes.

Table 10 notifierDetail node elements

Element Name	Purpose	Required Value (if any)
notifierAddr	Lists the users who should receive a given notification.	A comma-delimited list of user names (as defined within the Enterprise Manager), or "*" to send Events to all logged-in users.

emailDetail

The emailDetail node contains information required to e-Mail Alert and status notifications via a mail server (see Figure 16).

Figure 16 emailDetail nodes

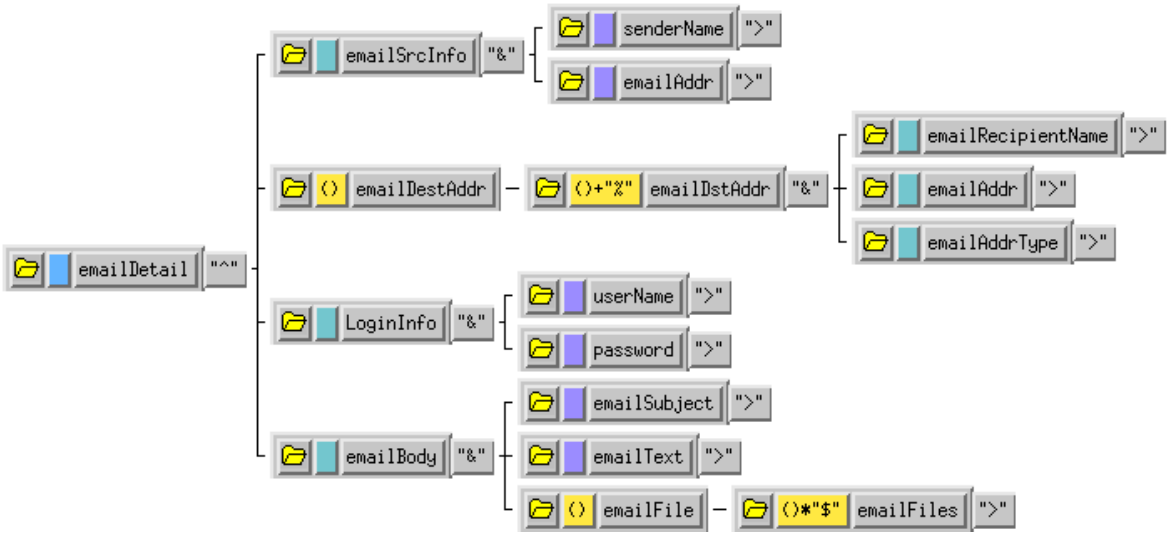


Table 11 describes the emailDetail nodes.

Table 11 emailDetail node elements

Node	Element Name	Purpose	Required Value (if any)
emailSrcInfo	senderName	Sender's name (such as "J. Doe").	
	emailAddr	Sender's e-Mail address.	
emailDestAddr	emailRecipient Name	Recipient's name.	
	emailAddr	Recipient's e-Mail address.	
	emailAddrType	Mail field into which recipient name will be inserted.	To, CC, Bcc
LoginInfo	userName	User name on the mail server for the account by which this message will be sent.	
	password	Mail server account password.	
emailBody	emailSubject	Subject of the e-Mail message.	
	emailText	Text of the e-Mail message.	
	emailFiles	Not supported; set to empty string. Reserved for future use.	

scriptDetail

Unlike the other notification channels, which send messages of various kinds, the scriptDetail channel executes a command, which can be a command script or an executable file. This notification channel provides you with a great range of flexibility, limited only by the power of the script you instruct the Control Broker to execute.

For example, you can use a Timer Event (whose schedule is set in the Control Broker properties within the Enterprise Manager) to launch a script at a given time by parsing the Event code for the Timer Event code, or launch a disk cleanup script if you receive a "disk usage" notification.

See **Table 17 on page 89** for more information on Alert notification codes, and **"The Notification System and Self-Correction" on page 138** for more ideas about using the Script channel to automate e*Gate maintenance (see Figure 17).

Figure 17 scriptDetail nodes

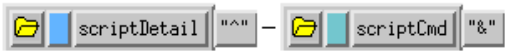


Table 12 describes scriptDetail nodes.

Table 12 scriptDetail node elements

Element Name	Purpose
scriptCmd	File name of the script to be executed.

Note: *The script must reside on the same Participating Host as the Control Broker that executes it.*

snmpAgentDetail

The snmpAgentDetail node contains the information required to direct notifications to the e*Gate SNMP Agent (see Figure 18).

Figure 18 snmpAgentDetail nodes

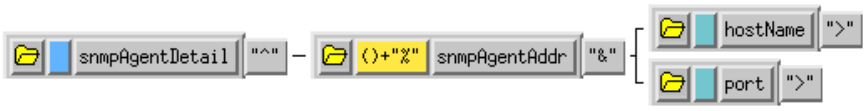


Table 13 describes the snmpAgentDetail nodes.

Table 13 snmpAgentDetail node elements

Element Name	Purpose
hostName	"*" or the name of the host on which the SNMP Agent is running (see the note at the end of this table).
port	TCP/IP port by which the SNMP agent communicates.

Note: *The "*" host name instructs the Control Broker to send notifications to any connected SNMP Agent. If "*" is specified, do not specify a port number: leave the field blank.*

If you wish to specify a particular SNMP agent host name, you must specify both the host name and the port number.

printerDetail

The printerDetail node contains the information required to direct Events to a printer (see Figure 19).

Figure 19 printerDetail nodes

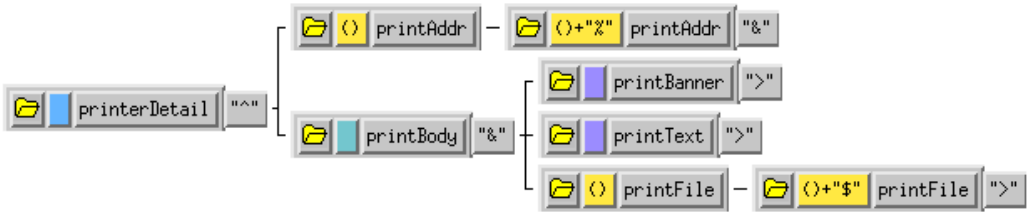


Table 14 describes the printerDetail nodes.

Table 14 printerDetail node elements

Node	Element Name	Purpose	Required Value (if any)
printAddr	printAddr	The name of the printer.	Printer name or port designation
printBody	printBanner	Text to be printed in the banner.	
	printText	Body of the notification to be printed.	
	printFile	Files to be attached to the message. The file must be located on the same system running the e*Gate Alert Agent, or be accessible from that system (for example, a valid Windows-NT UNC name in the format \\host\path\name).	

userAgentDetail

The userAgentDetail node enables you to send notifications to your own custom-written notification agent (see Figure 20).

Figure 20 userAgentDetail nodes

Table 15 describes the userAgentDetail nodes.

Table 15 userAgentDetail node elements

Element Name	Purpose	Required Value (if any)
UserAgentAddr	Host name and port address, or list of user names.	Non-interactive user agents (similar to the e*Gate Alert Agent) require a host name and port address. Interactive agents (similar to SeeBeyond's e*Gate Monitor or command-line application **stccmd**) require a list of user IDs.

5.6 Creating Custom Notifications

By default, the e*Gate notification system uses the Collaboration Rules script **Notification.tsc** to route Alert and status notification messages. You can edit that file, or create a different file with your own custom notification routing.

Note: *We recommend you make a copy of the default* ***Notification.tsc*** *file (using the Collaboration Editor's* ***Save As*** *function) before you make any site-specific changes, to preserve the original copy and provide a backup as you develop your site-specific script. If there are any errors in the Notification script, no notifications of any kind will be sent.*

To edit a Control Broker's notification routing

1. Open the Control Broker whose routing you want to edit.
2. Select the **Notification Routing** tab.
3. Click **NotificationRouting**.
4. When the **Notification Routing Properties** dialog box opens, click **Edit**.
5. Make any changes to the Collaboration Rules, then save the Rules file and exit the Editor.

The new rules will go into effect automatically as long as you edit your Notification Routing script using the above procedure. We strongly recommend that you do not bypass this procedure (for example, by editing the Notification script manually with a text editor). Doing so will not cause new rules to go into effect automatically and can potentially cause other problems.

The default **Notification.tsc** file shipped with e*Gate contains examples illustrating how to use every notification channel (see "Notification Channels" in the *e*Gate Integrator User's Guide*). The default routing sends notifications only to logged-in monitors; however, you can send notifications using any channel you like simply by changing an "if" statement and entering your own site-specific channel information (such as e-Mail addresses or pager numbers). See **Appendix B** for more information.

5.7 Using the event-send function

The Monk function **event-send** enables you to issue a monitoring Event from any Monk script. Events can use the standard SeeBeyond Event codes, or a "user Event" code you can use to communicate status conditions of user-created applications.

The syntax of the **event-send** function is:

```
(event-send "alert-category" "alert-sub-category"
"info-code" "custom-code" "reason-name"
"event-info-string" reason-code
(list))
```

Note that ***reason-code*** is unquoted, since it is an integer rather than a string.

Table 16 lists the possible values for each argument.

Table 16 Command Arguments for event-send

Argument	Type	Possible Values	Description
alert-category	String	ALERTCAT_STATE_ELEM	Element state
		ALERTCAT_MESSAGE_CONTENT	Message content
		ALERTCAT_STATE_EXTERNAL	External state
		ALERTCAT_OPERATIONAL	Operational
		ALERTCAT_PERFORMANCE	Performance
		ALERTCAT_RESOURCE	Resource
		ALERTCAT_USERDEFINED	User defined

Table 16 Command Arguments for event-send (Continued)

Argument	Type	Possible Values	Description
alert-subcategory	String	ALERTSUBCAT_CUSTOM	Custom category
		ALERTSUBCAT_DOWN	Down
		ALERTSUBCAT_UP	Up
		ALERTSUBCAT_UNRESP	Unresponsive
		ALERTSUBCAT_RESP	Responded
		ALERTSUBCAT_CANTCONN	Unable to connect
		ALERTSUBCAT_CONN	Connected
		ALERTSUBCAT_LOSTCONN	Lost Connection
		ALERTSUBCAT_UNUSABLE	Unusable/cannot ID
		ALERTSUBCAT_INTEREST	
		ALERTSUBCAT_EXPIRED	Expired
		ALERTSUBCAT_INTHRESH	Input threshold
		ALERTSUBCAT_OUTTHRESH	Output threshold
		ALERTSUBCAT_USERAUTH	User authentication
		ALERTSUBCAT_DELIVERY	Alert delivery
		ALERTSUBCAT_UNQUEUEABLE	Unqueueable
		ALERTSUBCAT_DISKTHRESH	Disk threshold
		ALERTSUBCAT_IQLIMIT	IQ Limit
		ALERTSUBCAT_STATUS	Status
		ALERTSUBCAT_TIMER	Timer
info-code	String	ALERTINFO_NONE	None
		ALERTINFO_FATAL	Fatal
		ALERTINFO_CONTROLLED	Controlled
		ALERTINFO_USER	User
		ALERTINFO_LOW	Low
		ALERTINFO_HIGH	High
		ALERTINFO_IOFAILED	IO Failure
		ALERTINFO_BELOW	Below
		ALERTINFO_ABOVE	Above
custom-code	String	Any one-byte (printable) character	Any meaning required for user application
reason-name	String	Descriptive string	Reason that the Event (described by reason-code) occurred

Table 16 Command Arguments for event-send (Continued)

Argument	Type	Possible Values	Description
event-info-string	String	Reserved for user agents or other applications using SeeBeyond's API to create monitoring Events that use this field	
reason-code	integer	Status or error code	Status/error code sent by the operating system or by the application generating the Event
event-detail	list of lists	Reserved for future use. In this field, always enter only the **(list)** expression, which will generate an empty list	

5.8 Understanding Notification Codes

Alert and status notifications both contain codes that give you additional information on each Event. This section provides a a series of lists and tables that explain e*Gate notification codes as a reference for classes of e*Gate monitoring Events.

The e*Gate system's notification codes are an 8-byte alphanumeric string that uses the format shown in Table 17. The notification codes are organized as follows:

- Each subcategory of the notification Event (bytes 1 through 4) has its own topic page.
- Each topic page discusses specific codes (specific to bytes 5 through 8) within a subcategory.

5.8.1 Notification Code Syntax

The entry position for e*Gate notification codes, along with its description and purpose are listed in Table 17.

Table 17 Notification Code Syntax

Entry	Description	Purpose
1	"e*Gate Monitoring Event" prefix	1
2	Author	0=SeeBeyond 1=User

Table 17 Notification Code Syntax (Continued)

Entry	Description	Purpose
3	Category	1=e*Gate component state 2=Event content 3=External state 4=Operational 5=Performance 6=Resource 7=User-defined 8=Internal state of a running e*Gate component 9=Script
4	Sub-category	0=Custom code, or the monitoring Event does not belong within an established category 1=Down 2=Up 3=Unresponsive 4=Responded 5=Unable to connect 6=Connected 7=Lost Connection 8=Unusable, Can't ID 9=Content of interest A=Expired B=Input threshold C=Output threshold D=User Authentication E=Alert Delivery F=Unqueueable G=Tally K=Disk Threshold L=Limit S=Status access T=Timer V=Can't start module W=Intelligent Queue (IQ) Operations

Table 17 Notification Code Syntax (Continued)

Entry	Description	Purpose
5	Element sending Event	1=Control Broker 2 =Registry 3=IQ Manager 4=Operating system A=User agent B=SeeBeyond Alert Agent C=e*Gate Monitor D=SeeBeyond SNMP Agent E=stccmd.exe F=command script G=e*Gate Enterprise Manager a=IQ b=Business Object Broker c=Communications client d=e*Way z=External
6	Element on whose behalf the Event is sent	Element code (same as above, except 0=self)
7	Failure code	1=Fatal 2=Controlled 3=User 4=Low 5=High 6=I/O failure 7=Below 8=Above 9=Configuration A=Events in B=Events out
8	Custom code	0 or user code (any printable character)

Note: *The elements of entry 5 in Table 17 are broken out and explained in further detail in the tables that appear in section* **"Alert Notification Messages" on page 96***.*

5.8.2 Notification Code Examples

The examples provided in this section illustrate how to read notification codes using the values listed in **Table 17 on page 89**.

105Cd070

- 1: Standard prefix for all e*Gate notification codes.
- 0: The code was generated by a SeeBeyond component.
- 5: Performance.
- C: Output threshold.
- d: e*Way.
- 0: Monitoring Event issued on behalf of self (in other words, the monitoring Event describes the element that issued it, not another element).
- 7: Above threshold.
- 0: Default value (no further information).

Translation. An e*Way reports that its output threshold has been exceeded.

10171C00

- 1: Standard prefix for all e*Gate notification codes.
- 0: The code was generated by a SeeBeyond component.
- 1: e*Gate component status.
- 7: Lost connection.
- 1: Control Broker.
- C: on behalf of the e*Gate Monitor.
- 0: No further information.
- 0: Default value (no further information).

Translation. The Control Broker reports that the e*Gate Monitor has lost its connection to its Control Broker. The monitoring Event is reported on behalf of the e*Gate Monitor. Because the e*Gate Monitor has lost the connection, it is unable to report the problem itself.

Note: *The system sends a special message "Lost Connection to Registry" alerting you when this connection is lost. See the* ***e*Gate Integrator System Administration and Operations Guide*** *for more information on the e*Gate Registry.*

104S1010

- 1: Standard prefix for all e*Gate notification codes.
- 0: The code was generated by a SeeBeyond component.
- 4: Operational.
- S: Status file access.
- 1: Control Broker.
- 0: Monitoring Event issued on behalf of self.
- 1: Fatal.
- 0: Default value (no further information).

Translation. The Control Broker reports that it cannot open the status file of a component, and thus, cannot report that component's status.

5.8.3 Notification Codes Used by Standard Components

Of the many possible notification codes the coding system supports, standard e*Gate components only use those shown in Table 18. These codes are more fully discussed in the **"Alert Notification Messages" on page 96** and in the e*Gate Monitor's Help system (use the Table of Contents or the Index to look up specific codes).

Table 18 Notification Codes Used by Standard e*Gate Components

Code	Description
10111310	IQ Manager down fatal
10111b10	BOB down fatal
10111d10	e*Way down fatal
10113020	IQ Manager down controlled
1011b020	BOB down controlled
1011d020	e*Way down controlled
1011d030	e*Way down user
10123000	IQ Manager up
1012b000	BOB up
1012d000	e*Way up
10131300	Control Broker detects IQ Manager unresponsive
10131b00	Control Broker detects BOB unresponsive
10131d00	Control Broker detects e*Way unresponsive
10131A00	Control Broker detects User Agent unresponsive
10131B00	Control Broker detects Alert Agent unresponsive
10131C00	Control Broker detects e*Gate Monitor unresponsive
10131D00	Control Broker detects SNMP Agent unresponsive

Table 18 Notification Codes Used by Standard e*Gate Components (Continued)

Code	Description
10131E00	Control Broker detects **stccmd** unresponsive
10131G00	Control Broker detects Enterprise Manager unresponsive
1013A100	User Agent detects Control Broker unresponsive
1013B100	Alert Agent detects Control Broker unresponsive
1013C100	e*Gate Monitor detects Control Broker unresponsive
1013D100	SNMP Agent detects Control Broker unresponsive
1013G100	Enterprise Manager detects Control Broker unresponsive
10141300	Control Broker detects IQ Manager responded
10141b00	Control Broker detects BOB responded
10141d00	Control Broker detects e*Way responded
10141A00	Control Broker detects User Agent responded
10141B00	Control Broker detects Alert Agent responded
10141C00	Control Broker detects e*Gate Monitor responded
10141D00	Control Broker detects SNMP Agent responded
10141E00	Control Broker detects **stccmd.exe** responded
10141G00	Control Broker detects Enterprise Manager responded
1014A100	User Agent detects Control Broker responded
1014B100	Alert Agent detects Control Broker responded
1014D100	SNMP Agent detects Control Broker responded
1014G100	Enterprise Manager detects Control Broker responded
10151D00	Control Broker cannot connect to SNMP Agent
1015A100	User Agent cannot connect to Control Broker
1015B100	Alert Agent cannot connect to Control Broker
1015C100	e*Gate Monitor cannot connect to Control Broker
1015E100	**stccmd** cannot connect to Control Broker
1015G100	Enterprise Manager cannot connect to Control Broker
1016A100	User Agent connected to Control Broker
1016B100	Alert Agent connected to Control Broker
1016C100	e*Gate Monitor connected to Control Broker
1016E100	**stccmd** connected to Control Broker
1016G100	Enterprise Manager connected to Control Broker
10161D00	Control Broker connected to SNMP Agent
10171A00	Control Broker lost connection to User Agent
10171B00	Control Broker lost connection to Alert Agent
10171C00	Control Broker lost connection to e*Gate Monitor

Table 18 Notification Codes Used by Standard e*Gate Components (Continued)

Code	Description
10171D00	Control Broker lost connection to SNMP Agent
10171E00	Control Broker lost connection to **stccmd**
10171G00	Control Broker lost connection to Enterprise Manager
1017A100	User Agent lost connection to Control Broker
1017B100	Alert Agent lost connection to Control Broker
1017C100	e*Gate Monitor lost connection to Control Broker
1017D100	SNMP Agent lost connection to Control Broker
1017E100	**stccmd** lost connection to Control Broker
1017G100	Enterprise Manager lost connection to Control Broker
10280000	Unusable message; cannot ID
10290000	Event content of interest
102A0000	Queue message expired
102F0000	Unqueueable message
1035dz00	e*Way can't connect to external
1036dz00	e*Way connected to external
1037dz00	e*Way lost connection to external
104D1010	User authentication failure
104E0010	Alert delivery failure
104ED010	e*Way delivery failure
104S1010	Control Broker can't get status
104V1010	Control Broker can't start module
104W0a10	IQ failure
105Bd060	e*Way input below threshold
105Bd070	e*Way input above threshold
105Cd060	e*Way output below threshold
105Cd070	e*Way output above threshold
105Bb040	BOB input below threshold
105Bb050	BOB input above threshold
105Cb060	BOB output below threshold
105Cb070	BOB output above threshold
106La000	IQ limit exceeded
106K1480	Control Broker detects disk usage above threshold
106T0000	Timer Event[a]
10700000	User defined
1089b090	Internal content problem with current configuration

a. A Timer Event triggers a signal to the CB to perform certain tasks at predetermined times. It is set up in the **Timer Events Properties** dialog box, which is accessed through the **Timers** tab on the CB's properties dialog box. The **Schedule Name** box is the "LogicalName" of the timer; the **Notes** box is the "eventInfoString," which is a description of the timer; and the **Schedule information** box is the *<name of that timer>* and sets the timing intervals (populates the **Timer Event** box on the **Timers** tab). The information entered in these boxes instructs the CB where and when to look for the Timer Event.

5.9 Explanation of Alert and Status Notification Messages

Alert and status notification messages both have two bytes that always contain the following information:

- The fifth byte identifies the component or feature that issued the monitoring Event.
- The sixth byte identifies the component or feature on behalf of which the monitoring Event was issued.

5.9.1 Alert Notification Messages

The elements that send monitoring Events are listed in entry 5 in **Table 17 on page 89**. These elements in turn contain Alert notification messages (called **Alert Names** in the GUI) that are a combination of the Element Name and the Event Name.

Note: *Registry, Operating system, command script, e*Gate Enterprise Manager, and Communications client (2, 4, F, G, and c respectively from entry 5 in* **Table 17 on page 89***) do generate Alert notifications.*

These elements are broken out and explained in further detail in the tables in this section. See **Table 17 on page 89** for a list of the component codes that are instrumental in understanding these messages.

Control Broker

The **Alert Name** messages, along with the problem and recommended action, for the Control Broker Alert notifications are listed in Table 19.

Table 19 Control Broker Alert Notifications

Alert Name	Code	Problem Possible Cause	Recommended Action
Alert Agent Detects CB Unresponsive	1013B100	User Agent detects a Control Broker that is unresponsive.	- Check the status of the CB that is unresponsive. - Check the network connection between the Alert Agent and the CB.

Table 19 Control Broker Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
Alert Delivery Failure	104E0010	An e*Gate component is unable to deliver a monitoring Event or notification. - The target monitor is not operating correctly. - The network connection between the monitor and the component has been broken. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Confirm that the monitor is operating properly. - Confirm that the network connection between the target monitor and the e*Gate component is functioning properly.
CB Can't Connect to SNMP Agent	10151D00	Control Broker is unable to connect to an SNMP Agent . - The configuration of the SNMP Agent is incorrect. - The network connection between the SNMP Agent and the Control Broker is broken. - The Control Broker is not running or not functioning properly. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the SNMP Agent's configuration. - Check the network connection between the SNMP Agent and the Control Broker. - Make sure that the Control Broker is running properly; restart it as necessary.

Table 19 Control Broker Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
CB Can't Get Status	104S1010	The Control Broker failed to access a component's status information. This is the only notification code (104S1010) of this type that is generated by a standard e*Gate component. - The status-information file for the target component is corrupted, inaccessible, or does not exist. - The target component has never been started successfully. See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	Check the e*Gate directory **\client\status** for a file named ***component_name*.stcstat** (where ***component*_name** is the name of the target component). If the file does not exist, start the target component; the file will be created automatically.
CB Can't Start Module	104V1010	The Control Broker failed to start the component. This is the only notification code (104V1010) of this type that is generated by a standard e*Gate. The most likely reason that the Control Broker could not start a component is a configuration error. See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check that the component is correctly configured, and that any related components (for example, IQs to which an e*Way will publish) have been correctly configured. - Correct any additional configuration errors - Attempt to restart the affected component.

Table 19 Control Broker Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
CB Detects Disk Usage Above Threshold	106K1480	Disk-space usage has exceeded a user-definable threshold. This is the only monitoring Event code (106K1480) generated by a standard e*Gate component. The amount of disk space on the monitored volume has exceeded the user-specified amount. SeeTable 17 on page 89 for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Increase the available disk space on the monitored volume. - If the increased disk usage is acceptable, raise the threshold at which the monitoring Event is sent.
CB Detects STCCmd Unresponsive	10131E00	Control Broker detects that STCCmd is unresponsive.	1 Check the status of the STCCmd. 2 Check the network connection.
CB Lost Connection To Alert Agent	10171B00	The Control Broker has lost its connection to an Alert Agent. An Alert Agent has ceased functioning properly, or the network connection between the Alert Agent and the Control Broker has been broken. **Additional Information** SeeTable 17 on page 89 for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	This condition may be difficult to resolve remotely. We recommend you directly log into the appropriate Participating Host when investigating and remediating this condition. - Check the status of the Alert Agent that has lost its connection to the Control Broker. - Check the network connection between the Control Broker and the Alert Agent.

Table 19 Control Broker Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
CB Lost Connection To e*Gate Monitor	10171C00	The Control Broker has lost its connection to an e*Gate Monitor. An e*Gate Monitor has ceased functioning properly, or the network connection between the e*Gate Monitor and the Control Broker has been broken. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	This condition may be difficult to resolve remotely. We recommend you directly log into the appropriate Participating Host when investigating and remediating this condition. - Check the status of the e*Gate Monitor that has lost its connection to the Control Broker. - Check the network connection between the Control Broker and the e*Gate Monitor.
CB Lost Connection To Registry	10171200	The Control Broker has lost its connection to a Registry. A Registry has ceased functioning properly, or the network connection between the Registry and the Control Broker has been broken. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	This condition may be difficult to resolve remotely. We recommend you directly log into the appropriate Participating Host when investigating and remediating this condition. - Check the status of the Registry that has lost its connection to the Control Broker. - Check the network connection between the Control Broker and the Registry.

Table 19 Control Broker Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
CB Lost Connection To SNMP Agent	10171D00	The Control Broker has lost its connection to an SNMP Agent. An SNMP Agent has ceased functioning properly, or the network connection between the SNMP Agent and the Control Broker has been broken. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	This condition may be difficult to resolve remotely. We recommend you directly log into the appropriate Participating Host when investigating and remediating this condition. - Check the status of the SNMP Agent that has lost its connection to the Control Broker. - Check the network connection between the Control Broker and the SNMP Agent.
CB Lost Connection To STCCmd	10171E00	The Control Broker has lost its connection to an STCCmd. An STCCmd has ceased functioning properly, or the network connection between the STCCmd and the Control Broker has been broken. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	This condition may be difficult to resolve remotely. We recommend you directly log into the appropriate Participating Host when investigating and remediating this condition. - Check the status of the STCCmd that has lost its connection to the Control Broker. - Check the network connection between the Control Broker and the STCCmd.

Table 19 Control Broker Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
CB Lost Connection To User Agent	10171A00	The Control Broker has lost its connection to a User Agent. A User Agent has ceased functioning properly, or the network connection between the User Agent and the Control Broker has been broken. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	This condition may be difficult to resolve remotely. We recommend you directly log into the appropriate Participating Host when investigating and remediating this condition. - Check the status of the User Agent that has lost its connection to the Control Broker. - Check the network connection between the Control Broker and the User Agent.
e*Gate Monitor Detects CB Unresponsive	1013C100	The Control Broker is not responding to queries from the e*Gate Monitor. In the Enterprise Manager **Control Broker Properties** dialog box, the **Check for Module Unresponsive** every value is greater than the interval the e*Gate Monitor uses to query the Control Broker.	1. Start the Enterprise Manager. Then, open the schema that contains the Control Broker being monitored by the e*Gate Monitor. 2. In the Navigator, select the host that contains the Control Broker. 3. In the Editor, select the Control Broker and click  on the toolbar. 4. In the **Check for Module Unresponsive every** box, type a smaller value.
CB Detects IQManager Unresponsive	10131300	Control Broker detects that the IQ Manager is unresponsive.	- Check the status of the IQ Manager that is unresponsive. - Check the network connection between the IQ Manager and the CB.

Table 19 Control Broker Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
IQManager Down Controlled	10113020	An e*Gate component/process has halted execution. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	No recommended action.

Table 19 Control Broker Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
IQManager Down Fatal	10111310	An e*Gate component/process has halted execution. Abnormal termination/shutdown monitoring Events can be caused when the executing process is externally terminated, as a response to the Task Manager's End Process command (under Windows NT), a **kill -9** command (under UNIX), or some other process-termination signal. Other causes include problems in the Participating Host's system, or severe configuration errors, or faults in user-written components. Normal termination/shutdown monitoring Events are most frequently generated when a component is shutdown via an interactive monitor, but any condition that causes a component to shutdown cleanly will generate this monitoring Event. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	1. Restart the component. If the component is configured to restart automatically and the component halted due to an abnormal termination, the component will restart itself. (The “auto restart” feature will not restart a component if it was brought down by a user command.) 2. In the Enterprise Manager, check the component's configuration and correct any configuration errors. 3. If the component continues to halt abnormally, set debugging flags, restart the component, and examine the component's log file. Although a **kill -9** command (under UNIX) is not an abnormal termination Event (it is an action a user can perform), the IQ Manager automatically restarts after this command has been issued if **Restart after abnormal termination** has been selected under the **Start Up** tab on the IQ Manager Properties dialog box.

Table 19 Control Broker Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
IQManager Down User	10113030	An e*Gate component/process has halted execution. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	There is no recommended action.
Operational Alert Delivery	104E*nnnn*	An e*Gate component is unable to deliver a monitoring Event or notification. - The target monitor is not operating correctly. - The network connection between the monitor and the component has been broken. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Confirm that the monitor is operating properly. - Confirm that the network connection between the target monitor and the e*Gate component is functioning properly.
Operational Can't Start Component	104V*nnnn*	The Control Broker has been unable to start a component. The most likely reason that the Control Broker could not start a component is a configuration error. See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check that the component is correctly configured, and that any related components (for example, IQs to which an e*Way will publish) have been correctly configured. - Correct any additional configuration errors - Attempt to restart the affected component.

Table 19 Control Broker Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
Operational User authentication	104D*nnnn*	An e*Gate process/component has failed in its attempt to authenticate with the Control Broker. A component has used a valid username/password combination to begin execution, but the schema has specified a different user in its **Run As** box on its property sheet's **General** tab. **Additional Information** Notification code 104Dxx1x means the authentication attempt failed. See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	Change the **Run As** parameter, or launch the component using the correct user name.
Resource Disk threshold	106K*nnnn*	Disk-space usage has exceeded a user-definable threshold. The amount of disk space on the monitored volume has exceeded the user-specified amount. **Additional Information** Notification code 106K1480 means disk usage above threshold. This is the only monitoring Event code generated by a standard e*Gate component. See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Increase the available disk space on the monitored volume. - If the increased disk usage is acceptable, raise the threshold at which the monitoring Event is sent.

Table 19 Control Broker Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
SNMP Agent Detects CB Unresponsive	1013D100	SNMP Agent detects a Control Broker that is unresponsive.	- Check the status of the CB that is unresponsive. - Check the network connection between the SNMP Agent and the CB.
User Agent Detects CB Unresponsive	1013A100	User Agent detects a Control Broker that is unresponsive.	- Check the status of the CB that is unresponsive. - Check the network connection between the User Agent and the CB.
User Authentication Failure	104D1010	An e*Gate process/component has failed in its attempt to authenticate with the Control Broker. A component has used a valid user name/password combination to begin execution, but the schema has specified a different user in its **Run As** box on its property sheet's **General** tab. **Additional Information** Notification code 104Dxx1x means the authentication attempt f**ailed.** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	Change the "Run As" parameter, or launch the component using the correct user name.

IQ Manager

The **Alert Name** messages, along with the problem and recommended action, for the IQ Manager Alert notifications are listed in Table 20.

Table 20 IQ Manager Alert Notifications

Alert Name	Code	Problem Possible Cause	Recommended Action
Out of disk space		The amount of disk space has been exceeded.	Increase the amount of available disk space.

Table 20 IQ Manager Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
Message Content Of Interest	10290000	An Event whose content may be of interest has been processed. This is the only notification code (10290000) of this type that is generated by a standard e*Gate. The conditions under which this monitoring Event is generated are completely under user control.	Take whatever action that your installation's procedures require.

User Agent

The **Alert Name** messages, along with the problem and recommended action, for the User Agent Alert notifications are listed in Table 21.

Table 21 User Agent Alert Notifications

Alert Name	Code	Problem Possible Cause	Recommended Action
CB Detects User Agent Unresponsive	10131A000	Control Broker detects a User Agent that is unresponsive.	- Check the status of the User Agent that is unresponsive. - Check the network connection between the User Agent and the CB.
User Agent Can't Connect To CB	1015A100	User Agent is unable to connect to a Control Broker. - The configuration of the User Agent is incorrect. - The network connection between the User Agent and the Control Broker is broken. - The Control Broker is not running or not functioning properly. **Additional Information** See **Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the User Agent's configuration. - Check the network connection between the User Agent and the Control Broker. - Make sure that the Control Broker is running properly; restart it as necessary.

Table 21 User Agent Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
User Agent Lost Connection To CB	1017A100	The User Agent has lost its connection to a Control Broker. A Control Broker has ceased functioning properly, or the network connection between the Control Broker and the User Agent has been broken. **Additional Information** See Table 17 on page 89 for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	This condition may be difficult to resolve remotely. We recommend you directly log into the appropriate Participating Host when investigating and remediating this condition. - Check the status of the Control Broker that has lost its connection to the User Agent. - Check the network connection between the User Agent and the Control Broker.

SeeBeyond Alert Agent

The **Alert Name** messages, along with the problem and recommended action, for the SeeBeyond Alert Agent Alert notifications are listed in Table 22.

Table 22 SeeBeyond Alert Agent Alert Notifications

Alert Name	Code	Problem	Solution
Alert Agent Can't Connect To CB	1015B100	Alert Agent is unable to connect to a Control Broker. - The configuration of the Alert Agent is incorrect. - The network connection between the Alert Agent and the Control Broker is broken. - The Control Broker is not running or not functioning properly. **Additional Information** See Table 17 on page 89 for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the Alert Agent's configuration. - Check the network connection between the Alert Agent and the Control Broker. - Make sure that the Control Broker is running properly; restart it as necessary.

Table 22 SeeBeyond Alert Agent Alert Notifications (Continued)

Alert Name	Code	Problem	Solution
Alert Agent Lost Connection To CB	1017B100	The Alert Agent has lost its connection to a Control Broker. A Control Broker has ceased functioning properly, or the network connection between the Control Broker and the Alert Agent has been broken. **Additional Information** See Table 17 on page 89 for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	This condition may be difficult to resolve remotely. We recommend you directly log into the appropriate Participating Host when investigating and remediating this condition. - Check the status of the Control Broker that has lost its connection to the Alert Agent. - Check the network connection between the Alert Agent and the Control Broker.
CB Detects Alert Agent Unresponsive	10131B00	Control Broker detects an Alert Agent that is unresponsive.	- Check the status of the Alert Agent that is unresponsive. - Check the network connection between the Alert Agent and the CB.

e*Gate Monitor

The **Alert Name** messages, along with the problem and recommended action, for the e*Gate Monitor Alert notifications are listed in Table 23.

Table 23 e*Gate Monitor Alert Notifications

Alert Name	Code	Problem Possible Cause	Recommended Action
CB Detects e*Gate Monitor Unresponsive	10131C00	Control Broker detects that the e*Gate Monitor is unresponsive.	- Check the status of the e*Gate Monitor that is unresponsive. - Check the network connection between the e*Gate Monitor and the CB.

Table 23 e*Gate Monitor Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
e*Gate Monitor Cannot Connect to CB	1015C100	e*Gate Monitor is unable to connect to a Control Broker. - The configuration of the e*Gate Monitor is incorrect. - The network connection between the e*Gate Monitor and the Control Broker is broken. - The Control Broker is not running or not functioning properly. **Additional Information** See **Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Ensure that the Control Broker is functioning properly. - Check the e*Gate system's network connections. - Check the e*Gate Monitor configuration. - Check the network connection between the e*Gate Monitor and the Control Broker. - Make sure that the Control Broker is running properly; restart it as necessary.
e*Gate Monitor Detects CB Unresponsive	1013C100	The Control Broker is not responding to queries from the e*Gate Monitor. In the Enterprise Manager **Control Broker Properties** dialog box, the **Check for Module Unresponsive every** value is greater than the interval the e*Gate Monitor uses to query the Control Broker.	1. Start the Enterprise Manager. Then, open the schema that contains the Control Broker being monitored by the e*Gate Monitor. 2. In the Navigator, select the host that contains the Control Broker. 3. In the Editor, select the Control Broker and click  on the toolbar. 4. In the **Check for Module Unresponsive every** box, type a smaller value.

Table 23 e*Gate Monitor Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
e*Gate Monitor Lost Connection to CB	1017C100	The e*Gate Monitor has lost its connection to a Control Broker. A Control Broker has ceased functioning properly, the network connection between the Control Broker and the e*Gate Monitor has been broken, or the Control Broker has been shut down. **Additional Information** See Table 17 on page 89 for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	This condition may be difficult to resolve remotely. We recommend you directly log into the appropriate Participating Host when investigating and remedying this condition. - Check the status of the Control Broker that has lost its connection to the e*Gate Monitor. - Check the network connection between the e*Gate Monitor and the Control Broker. - Restart the Control Broker.

SeeBeyond SNMP Agent

The **Alert Name** messages, along with the problem and recommended action, for the SeeBeyond SNMP Agent Alert notifications are listed in Table 24.

Table 24 SeeBeyond SNMP Agent Alert Notifications

Alert Name	Code	Problem Possible Cause	Recommended Action
CB Detects SNMP Agent Unresponsive	10131D00	Control Broker detects an SNMP Agent that is unresponsive.	- Check the status of the SNMP Agent that is unresponsive. - Check the network connection between the SNMP Agent and the CB.
SNMP Agent Lost Connection To CB	1017D100	The SNMP Agent has lost its connection to a Control Broker. A Control Broker has ceased functioning properly, or the network connection between the Control Broker and the SNMP Agent has been broken. **Additional Information** See **Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	This condition may be difficult to resolve remotely. We recommend you directly log into the appropriate Participating Host when investigating and remediating this condition. - Check the status of the Control Broker that has lost its connection to the SNMP Agent. - Check the network connection between the SNMP Agent and the Control Broker.

stccmd.exe

The **Alert Name** messages, along with the problem and recommended action, for the **stccmd.exe** Alert notifications are listed in Table 25.

Table 25 stccmd.exe Alert Notifications

Alert Name	Code	Problem	Solution
CB Detects STCCmd Unresponsive	10131E00	Control Broker detects that STCCmd is unresponsive.	1 Check the status of the STCCmd. 2 Check the network connection.
STCCmd Can't Connect To CB	1015E100	STCCmd is unable to connect to a Control Broker. - The configuration of the STCCmd is incorrect. - The network connection between the STCCmd and the Control Broker is broken. - The Control Broker is not running or not functioning properly. **Additional Information** See Table 17 on page 89 for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the STCCmd configuration. - Check the network connection between the STCCmd and the Control Broker. - Make sure that the Control Broker is running properly; restart it as necessary.
STCCmd Lost Connection To CB	1017E100	The STCCmd has lost its connection to a Control Broker. A Control Broker has ceased functioning properly, or the network connection between the Control Broker and the STCCmd has been broken. **Additional Information** See Table 17 on page 89 for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	This condition may be difficult to resolve remotely. We recommend you directly log into the appropriate Participating Host when investigating and remediating this condition. - Check the status of the Control Broker that has lost its connection to the STCCmd. - Check the network connection between the STCCmd and the Control Broker.

IQ

The **Alert Name** messages, along with the problem and recommended action, for the IQ Alert notifications are listed in Table 26.

Table 26 IQ Alert Notifications

Alert Name	Code	Problem Possible Cause	Recommended Action
IQ Failure	104000a10	An e*Gate IQ Manager has failed in its attempt to authenticate with the Control Broker. A component has used a valid username/password combination to begin execution, but the schema has specified a different user in its **Run As** box on its property sheet's **General** tab. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	Change the **Run As** parameter, or launch the component using the correct username.

Business Object Broker

The **Alert Name** messages, along with the problem and recommended action, for the Business Object Broker Alert notifications are listed in Table 27.

Table 27 Business Object Broker Alert Notifications

Alert Name	Code	Problem Possible Cause	Recommended Action
BOB Down Controlled	1011b020	An e*Gate component/process has halted execution. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	Check the Alert or log file to find out what the problem is, then act accordingly.

Table 27 Business Object Broker Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
BOB Down Fatal	10111b10	An e*Gate component/process has halted execution. Abnormal termination/shutdown monitoring Events can be caused when the executing process is externally terminated, as a response to the Task Manager's End Process command (under Windows NT), a **kill -9** command (under UNIX), or some other process-termination signal. Other causes include problems in the Participating Host's system, or severe configuration errors, or faults in user-written components. Normal termination/shut down monitoring Events are most frequently generated when a component is shut down via an interactive monitor, but any condition that causes a component to shutdown cleanly will generate this monitoring Event. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	1. Restart the component. If the component is configured to restart automatically and the component halted due to an abnormal termination, the component will restart itself. (The “auto restart” feature will not restart a component if it was brought down by a user command.) 2. In the Enterprise Manager, check the component's configuration and correct any configuration errors. 3. If the component continues to halt abnormally, set debugging flags, restart the component, and examine the component's log file. Although a **kill -9** command (under UNIX) is not an abnormal termination Event (it is an action a user can perform), the BOB automatically restarts after this command has been issued if **Restart after abnormal termination** has been selected under the **Start Up** tab on the BOB Properties dialog box.

Table 27 Business Object Broker Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
BOB Input Above Threshold	105Bb050	The user-defined Event Input threshold for an e*Gate component has been exceeded. The monitored component has input too many Events, relative to the threshold which you specified within the Enterprise Manager. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the components to which this component's Collaboration(s) are subscribing to make sure that they are publishing the appropriate number of Events. - If system performance is acceptable, change the threshold settings to accommodate the current rate of Event processing.
BOB Input Below Threshold	105Bb040	The user-defined Event Input is lower than the specified minimum threshold for an e*Gate component. The monitored component has input too few Events, relative to the specified threshold within the Enterprise Manager. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the components to which this component's Collaboration(s) are subscribing to make sure that they are publishing the appropriate number of Events. - If system performance is acceptable, change the threshold settings to accommodate the current rate of Event processing.

Table 27 Business Object Broker Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
BOB Output Above Threshold	105Cb070	The user-defined Event Output threshold for an e*Gate component/process has been exceeded. The monitored component has output too many Events, relative to the threshold which you specified within the Enterprise Manager. **Additional Information** SeeTable 17 on page 89 for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the components to which this component's Collaboration(s) are subscribing to make sure that the appropriate number of Events are being published. - Check the Collaboration Rules scripts to ensure that Events are being processed appropriately. - If system performance is acceptable, change the threshold settings to accommodate the current rate of Event processing.
BOB Output Below Threshold	105Cb060	The user-defined Event Output is lower than the specified minimum threshold for an e*Gate component/process. The monitored component has output too few Events, relative to the threshold which you specified within the Enterprise Manager. **Additional Information** SeeTable 17 on page 89 for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the components to which this component's Collaboration(s) are subscribing to make sure that the appropriate number of Events are being published. - Check the Collaboration Rules scripts to ensure that Events are being processed appropriately. - If system performance is acceptable, change the threshold settings to accommodate the current rate of Event processing.

Table 27 Business Object Broker Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
CB Detects BOB Unresponsive	10131b00	Control Broker detects a BOB that is unresponsive.	- Check the status of the BOB that is unresponsive. - Ensure that the BOB was configured correctly. - Check the network connection between the IQ Manager and the CB.
Internal State API	108U*nnnn*	An e*Gate component/process has failed in its attempt to perform an API call. **Additional Information** See **Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	Check the log file of the e*Gate component that generated the Alert to determine the cause. If not enough information has been saved in the log file, increase the logging level and check the log file a second time to see if you can identify the cause.
Operational Status Access	104S*nnnn*	An e*Gate process/component has attempted to access the status information of another process/component. - The status-information file for the target component is corrupted, inaccessible, or does not exist. - The target component has never been started successfully. See **Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the e*Gate directory **\client\status** for a file named ***component_name*.stcstat** (where ***component*_name** is the name of the target component). If the file does not exist, start the target component; the file will be created automatically. - Check your log file for reasons why the Collaboration may have failed (for example, **MONKEXCEPT**), and fix the errors.

Table 27 Business Object Broker Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
UnQueueable Message	102F0000	An Event cannot be inserted into a queue. Notification code 102F0000, which means Event is Unqueueable, is the only code of this type that is generated by a standard e*Gate. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- If you are using the SeeBeyond Standard IQ Service, make sure that the IQ Manager supervising the affected IQ is operating correctly. - If you are using a database IQ Service, make sure that the client components of the database are properly installed and configured, and that you have sufficient access to perform the required operations. - Make sure there is sufficient disk space on the system hosting the IQ.

e*Way

The **Alert Name** messages, along with the problem and recommended action, for the e*Way Alert notifications are listed in Table 28.

Table 28 e*Way Alert Notifications

Alert Name	Code	Problem Possible Cause	Recommended Action
CB Detects E*Way Unresponsive	10131d00	Control Broker detects an e*Way that is unresponsive.	- Check the status of the e*Way that is unresponsive. - Ensure that the e*Way was configured correctly. - Check the network connection between the e*Way and the CB.

Table 28 e*Way Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
E*Way Can't Connect to External	1035dz00	An e*Way cannot connect to an external component. Notification code 1035dz00 is the standard code for this monitoring Event. - The connection between the component and its target has failed. - The target to which the component is attempting to connect is not functioning properly or is not able to permit the connection to be made. (For example, an e*Way is unable to make a connection to a database.) **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the connection between the component and the target. - Confirm that the target is accessible and is functioning properly.
E*Way Delivery Failure	104Ed010	An e*Gate component is unable to deliver a monitoring Event or notification. - The target monitor is not operating correctly. - The network connection between the monitor and the component has been broken. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Confirm that the monitor is operating properly. - Confirm that the network connection between the target monitor and the e*Gate component is functioning properly.

Table 28 e*Way Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
E*Way Down Controlled	1011d020	An e*Gate component/process has halted execution. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	Check the Alert or log file to find out what the problem is, then act accordingly.

Table 28 e*Way Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
E*Way Down Fatal	10111d10	An e*Gate component/process has halted execution. Abnormal termination/shutdown monitoring Events can be caused when the executing process is externally terminated, as a response to the Task Manager's End Process command (under Windows NT), a **kill -9** command (under UNIX), or some other process-termination signal. Other causes include problems in the Participating Host's system, or severe configuration errors, or faults in user-written components. Normal termination/shut down monitoring Events are most frequently generated when a component is shut down via an interactive monitor, but any condition that causes a component to shutdown cleanly will generate this monitoring Event. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	1 Restart the component. If the component is configured to restart automatically and the component halted due to an abnormal termination, the component will restart itself. (The “auto restart” feature will not restart a component if it was brought down by a user command.) 2 In the Enterprise Manager, check the component's configuration and correct any configuration errors. 3 If the component continues to halt abnormally, set debugging flags, restart the component, and examine the component's log file. Although a **kill -9** command (under UNIX) is not an abnormal termination Event (it is an action a user can perform), the e*Way automatically restarts after this command has been issued if **Restart after abnormal termination** has been selected under the **Start Up** tab on the e*Way Properties dialog box.

Table 28 e*Way Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
E*Way Input Above Threshold	105Ba070	The user-defined Event Input threshold for an e*Gate component has been exceeded. The monitored component has input too many Events, relative to the threshold which you specified within the Enterprise Manager. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the components to which this component's Collaboration(s) are subscribing to make sure that they are publishing the appropriate number of Events. - If system performance is acceptable, change the threshold settings to accommodate the current rate of Event processing.
E*Way Input Below Threshold	105Ba060	The user-defined Event Input is lower than the specified minimum threshold for an e*Gate component. The monitored component has input too few Events, relative to the specified threshold within the Enterprise Manager. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the components to which this component's Collaboration(s) are subscribing to make sure that they are publishing the appropriate number of Events. - If system performance is acceptable, change the threshold settings to accommodate the current rate of Event processing.

Table 28 e*Way Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
E*Way Lost Connection External	1037dz00	The Control Broker has lost its connection to an e*Gate component/process. Notification code 1037dz00 is the standard code for this monitoring Event. - The connection between the component and its target has failed. - The target to which the component is attempting to connect has stopped functioning properly or no longer permits the connection to be made. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the connection between the component and the target. - Confirm that the target is accessible and is functioning properly.
E*Way Not Connected to External	1038dz00	The Control Broker has lost its connection to an e*Gate component/process. - The connection between the component and its target has failed. - The target to which the component is attempting to connect has stopped functioning properly or no longer permits the connection to be made. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the connection between the component and the target. - Confirm that the target is accessible and is functioning properly.

Table 28 e*Way Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
E*Way Output Above Threshold	105Ca070	The user-defined Event Output threshold for an e*Gate component/process has been exceeded. The monitored component has output too many Events, relative to the threshold which you specified within the Enterprise Manager. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the components to which this component's Collaboration(s) are subscribing to make sure that the appropriate number of Events are being published. - Check the Collaboration Rules scripts to ensure that Events are being processed appropriately. - If system performance is acceptable, change the threshold settings to accommodate the current rate of Event processing.
E*Way Output Below Threshold	105Ca060	The user-defined Event Output is lower than the specified minimum threshold for an e*Gate component/process. The monitored component has output too few Events, relative to the threshold which you specified within the Enterprise Manager. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the components to which this component's Collaboration(s) are subscribing to make sure that the appropriate number of Events are being published. - Check the Collaboration Rules scripts to ensure that Events are being processed appropriately. - If system performance is acceptable, change the threshold settings to accommodate the current rate of Event processing.

Table 28 e*Way Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
Internal State API	108U*nnnn*	An e*Gate component/process has failed in its attempt to perform an API call. **Additional Information** See **Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	Check the log file of the e*Gate component that generated the Alert to determine the cause. If not enough information has been saved in the log file, increase the logging level and check the log file a second time to see if you can identify the cause.
Operational Status Access	104S*nnnn*	An e*Gate process/component has attempted to access the status information of another process/ component. - The status-information file for the target component is corrupted, inaccessible, or does not exist. - The target component has never been started successfully. See **Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the e*Gate directory **\client\status** for a file named ***component_name*.stcstat** (where ***component*_name** is the name of the target component). If the file does not exist, start the target component; the file will be created automatically. - Check your log file for reasons why the Collaboration may have failed (for example, **MONKEXCEPT**), and fix the errors.

Table 28 e*Way Alert Notifications (Continued)

Alert Name	Code	Problem Possible Cause	Recommended Action
UnQueueable Message	102F0000	An Event cannot be inserted into a queue. Notification code 102F0000, which means Event is Unqueueable, is the only code of this type that is generated by a standard e*Gate. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- If you are using the SeeBeyond Standard IQ Service, make sure that the IQ Manager supervising the affected IQ is operating correctly. - If you are using a database IQ Service, make sure that the client components of the database are properly installed and configured, and that you have sufficient access to perform the required operations. - Make sure there is sufficient disk space on the system hosting the IQ.

External

The **Alert Name** messages, along with the problem and recommended action, for the External Alert notifications are listed in Table 28.

Table 29 External Alert Notifications

Alert Name	Code	Problem Possible Cause	Recommended Action
Event Content Unusable Can't Id	1028*nnnn*	An Event is so garbled that e*Gate cannot identify or process it. Normally, only inbound e*Ways generate this monitoring Event. - The data source may be sending garbled information. - The connection between the data source and the e*Gate system may be introducing noise. **Additional Information** Notification code 10280000, meaning Unusable Event, is the only code of this type that is generated by a standard e*Gate. See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Confirm that the data being sent to the e*Gate system is being transmitted cleanly and correctly. - Check the connection between the external system and the e*Gate system.

Table 29 External Alert Notifications

Alert Name	Code	Problem Possible Cause	Recommended Action
External State Lost Connection	1037*nnnn*	The Control Broker has lost its connection to an e*Gate component/process. - The connection between the component and its target has failed. - The target to which the component is attempting to connect has stopped functioning properly or no longer permits the connection to be made. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the connection between the component and the target. - Confirm that the target is accessible and is functioning properly.
External State Unable to Connect	1035*nnnn*	An e*Way cannot connect to an external component. - The connection between the component and its target has failed. - The target to which the component is attempting to connect is not functioning properly or is not able to permit the connection to be made. (For example, an e*Way is unable to make a connection to a database.) **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Check the connection between the component and the target. - Confirm that the target is accessible and is functioning properly.

Table 29 External Alert Notifications

Alert Name	Code	Problem Possible Cause	Recommended Action
Internal State Unusable Can't Id	1088*nnnn*	An Event is so garbled that e*Gate cannot identify or process it. Normally, only inbound e*Ways generate this monitoring Event. - The data source may be sending garbled information. - The connection between the data source and the e*Gate system may be introducing noise. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Confirm that the data being sent to the e*Gate system is being transmitted cleanly and correctly. - Check the connection between the external system and the e*Gate system.
Script Unusable Can't Id	1098*nnnn*	An Event is so garbled that e*Gate cannot identify or process it. Normally, only inbound e*Ways generate this monitoring Event. - The data source may be sending garbled information. - The connection between the data source and the e*Gate system may be introducing noise. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Confirm that the data being sent to the e*Gate system is being transmitted cleanly and correctly. - Check the connection between the external system and the e*Gate system.

Table 29 External Alert Notifications

Alert Name	Code	Problem Possible Cause	Recommended Action
Unusable Message	10280000	An Event is so garbled that e*Gate cannot identify or process it. Notification code 10280000, meaning Unusable Event, is the only code of this type that is generated by a standard e*Gate. Normally, only inbound e*Ways generate this monitoring Event. - The data source may be sending garbled information. - The connection between the data source and the e*Gate system may be introducing noise. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.	- Confirm that the data being sent to the e*Gate system is being transmitted cleanly and correctly. - Check the connection between the external system and the e*Gate system.
User Defined Unusable Can't Id	1078*nnnn*	Standard definition of “Event Unusable” monitoring Events: An Event is so garbled that e*Gate cannot identify or process it. Normally, only inbound e*Ways generate this monitoring Event. **Additional Information** See**Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components. Note that in this instance notification code 1078xx0x means standard code for “Event Unusable” monitoring.	This monitoring Event has been generated by a user-written component. The circumstances under which this monitoring Event is generated, and any corrective action that may be required, are defined at your installation. Please consult your installation's procedures for more information.

5.9.2 Status Notification Messages

The e*Gate Monitor also displays status notification messages. These messages are informational only, listing status of e*Gate components that are operating normally and do not require action by the user.

Status codes and status messages

The codes and messages are the same as for Alert Events. See **"Notification Code Syntax" on page 89** for details.

Note: *See* **"Status Notification Descriptions" on page 146** *for a list of the status Event messages.*

5.10 Custom Alert Notification Codes

SeeBeyond has created a number of Alert notifications that are not generated by standard e*Gate components. Instead, they have been created for the sole purpose of customization. You will only see these custom Alert notification messages if they have been customized and a user-written component generates them.

Table 30 lists the Alert notifications that are available to customize.

Table 30 Alert Notification Codes Available To Customize

Notification Code	Alert Name/Message
1018*nnnn*	Element State Unusable Can't Id
101A*nnnn*	Element State Expired
101B*nnnn*	Element State Input threshold
101C*nnnn*	Element State Output threshold
101D*nnnn*	Element State User authentication
101E*nnnn*	Element State Alert Delivery
101F*nnnn*	Element State Unqueueable
101G*nnnn*	Element State Tally
101K*nnnn*	Element State Disk threshold
101S*nnnn*	Element State Status Access
101T*nnnn*	Element State Timer
101U*nnnn*	Element State API
101V*nnnn*	Element State Can't Start Component
101W*nnnn*	Element State IQ Operations
1021*nnnn*	Event Content Down
1022*nnnn*	Event Content Up

Table 30 Alert Notification Codes Available To Customize (Continued)

Notification Code	Alert Name/Message
1023*nnnn*	Event Content Unresponsive
1024*nnnn*	Event Content Responded
1025*nnnn*	Event Content Unable to Connect
1026*nnnn*	Event Content Connected
1027*nnnn*	Event Content Lost Connection
102B*nnnn*	Event Content Input threshold
102C*nnnn*	Event Content Output threshold
102D*nnnn*	Event Content User authentication
102E*nnnn*	Event Content Alert Delivery
102G*nnnn*	Event Content Tally
102K*nnnn*	Event Content Disk threshold
102L*nnnn*	Event Content Limit
102S*nnnn*	Event Content Status Access
102T*nnnn*	Event Content Timer
102U*nnnn*	Event Content API
102V*nnnn*	Event Content Can't Start Component
102W*nnnn*	Event Content IQ Operations
1031*nnnn*	External State Down
1032*nnnn*	External State Up
1033*nnnn*	External State Unresponsive
1034*nnnn*	External State Responded
1038*nnnn*	External State Unusable Can't Id
1039*nnnn*	External State Content of interest
103A*nnnn*	External State Expired
103B*nnnn*	External State Input threshold
103C*nnnn*	External State Output threshold
103D*nnnn*	External State User authentication
103E*nnnn*	External State Alert Delivery
103F*nnnn*	External State Unqueueable
103G*nnnn*	External State Tally
103K*nnnn*	External State Disk threshold
103L*nnnn*	External State Limit
103S*nnnn*	External State Status Access
103T*nnnn*	External State Timer

Table 30 Alert Notification Codes Available To Customize (Continued)

Notification Code	Alert Name/Message
103U*nnnn*	External State API
103V*nnnn*	External State Can't Start Component
103W*nnnn*	External State IQ Operations
1041*nnnn*	Operational Down
1042*nnnn*	Operational Up
1043*nnnn*	Operational Unresponsive
1044*nnnn*	Operational Responded
1045*nnnn*	Operational Unable to Connect
1046*nnnn*	Operational Connected
1047*nnnn*	Operational Lost Connection
1048*nnnn*	Operational Unusable Can't Id
1049*nnnn*	Operational Content of interest
104A*nnnn*	Operational Expired
104B*nnnn*	Operational Input threshold
104C*nnnn*	Operational Output threshold
104F*nnnn*	Operational Unqueueable
104G*nnnn*	Operational Tally
104K*nnnn*	Operational Disk threshold
104L*nnnn*	Operational Limit
104T*nnnn*	Operational Timer
104U*nnnn*	Operational API
1051*nnnn*	Performance Down
1052*nnnn*	Performance Up
1053*nnnn*	Performance Unresponsive
1054*nnnn*	Performance Responded
1055*nnnn*	Performance Unable to Connect
1056*nnnn*	Performance Connected
1057*nnnn*	Performance Lost Connection
1058*nnnn*	Performance Unusable Can't Id
1059*nnnn*	Performance Content of interest
105A*nnnn*	Performance Expired
105D*nnnn*	Performance User authentication
105E*nnnn*	Performance Alert Delivery
105F*nnnn*	Performance Unqueueable

Table 30 Alert Notification Codes Available To Customize (Continued)

Notification Code	Alert Name/Message
105G*nnnn*	Performance Tally
105K*nnnn*	Performance Disk threshold
105L*nnnn*	Performance Limit
105S*nnnn*	Performance Status Access
105T*nnnn*	Performance Timer
105U*nnnn*	Performance API
105V*nnnn*	Performance Can't Start Component
105W*nnnn*	Performance IQ Operations
1061*nnnn*	Resource Down
1062*nnnn*	Resource Up
1063*nnnn*	Resource Unresponsive
1064*nnnn*	Resource Responded
1065*nnnn*	Resource Unable to Connect
1066*nnnn*	Resource Connected
1067*nnnn*	Resource Lost Connection
1068*nnnn*	Resource Unusable Can't Id
1069*nnnn*	Resource Content of interest
106A*nnnn*	Resource Expired
106B*nnnn*	Resource Input threshold
106C*nnnn*	Resource Output threshold
106D*nnnn*	Resource User authentication
106G*nnnn*	Resource Tally
106S*nnnn*	Resource Status Access
106U*nnnn*	Resource API
106V*nnnn*	Resource Can't Start Component
106W*nnnn*	Resource IQ Operations
108V*nnnn*	Internal State Can't Start Component
108W*nnnn*	Internal State IQ Operations
109V*nnnn*	Script Can't Start Component
109W*nnnn*	Script IQ Operations

5.11 The Notification System and Self-Correction

You can use the Notification system to launch command scripts that enable e*Gate to modify system parameters or even its own operations as follows:

- The "Script" Notification channel launches a command script or executable file. By setting up conditions in the Notification Routing script (see **"Creating Custom Notifications" on page 86**), you can launch a command script to address specific Monitoring Events within the e*Gate system.

  For example, e*Gate can monitor disk usage and send a Monitoring Event when a monitored volume's disk space exceeds a given amount (see "Setting Disk-usage Thresholds" in the *e*Gate Integrator User's Guide*). Instead of simply broadcasting the warning to the e*Gate Monitor, the Notification Routing script might launch a custom disk-cleanup script that frees disk space in whatever manner you determine.

- You can also address operational issues using other Monitoring Events and **stccmd**. For example, you can configure BOBs and e*Ways to send Monitoring Events if the number of Events processed within a given period exceeds a lower limit (see "Setting Event-processing Thresholds" in the *e*Gate Integrator User's Guide*). You can configure the Notification Routing script to use **stccmd** to start additional e*Ways or BOBs (previously defined within the schema) for workload balancing.

- You can also use the Script channel to provide Notification functions besides those within e*Gate. For example, you could send messages directly to users' terminals using applications like UNIX **write**, or launch a mail client and send e-Mail using your own mail application

- Any command or command script that can be launched at a shell prompt can be launched by the Script channel.

Chapter 6

Frequently Asked Questions

This chapter lists some common problems that may be encountered when using e*Gate, and answers frequently asked questions (FAQs). It also offers tips on how to make e*Gate run more effectively.

6.1 Introduction: Using These FAQs and Tips

The purpose of this chapter is to make you aware of some of the problems that may arise when using e*Gate. They are a combination of hints, tips, and ways to obtain optimum performance from your system.

Use this information to help you operate your e*Gate system in the most efficient manner, and, at the same time, aid you in spotting problems to avoid.

This chapter explains

6.2 General FAQs

1 **What should I do if I encounter a Motif memory leak?**

This problem is usually found in the following locations:

- Event Type Definition (ETD) Editor
- Collaboration Editor

The following steps help prevent Motif memory leakage:

A Save often when working with large Collaborations.

B Close any editor not currently in use.

C Use the Task Manager to monitor memory usage.

D Save your files, close them, and relaunch the editor to free additional memory when necessary.

E Reboot the machine.

Note: *This problem is only found in e*Gate release 4.1.2 and before.*

2 **When I import from one host to another, my system cannot start the Control Broker. Why is this?**

A Most likely you have defined your Participating Host as "localhost," and e*Gate cannot identify the Participating Host when it is defined as "localhost." To fix this problem, give the Participating Host its own unique name.

3 **If I access the ETD Editor from the e*Gate Enterprise Manager toolbar, it does not run a check to verify that any third party requirements are installed before opening, specifically, any Java classes that are required by the application (e*Way Intelligent Adapter, and so on), which must be placed in the ClassPath environment variable. Can I fix this?**

Yes, this can be fixed in Windows. From the Windows **Start** button:

A Choose the **Settings > Control Panel > System** menu option.

B When the **System Properties** dialog box opens, select the **Environment** tab and set the **Value** for the ClassPath as required.

6.3 e*Gate Monitor FAQs

1 **What should I do if my monitor is not operating?**

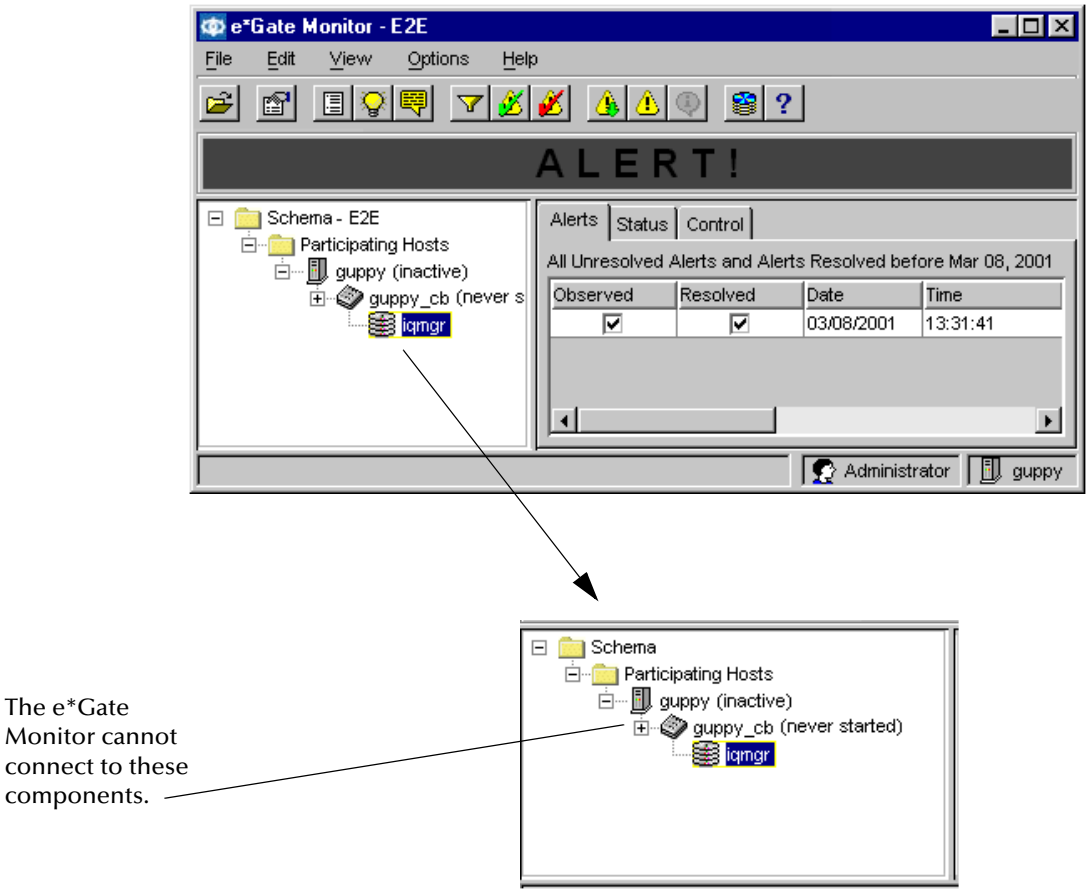
- Check that the Control Broker is running correctly.
- Make sure the network name of the host where the Control Broker is running is correct, and is pointing to the correct host.

2 **What does it mean if my host is inactive and/or if my host's Control Broker is not started?**

If your host or hosts supporting a schema are inactive, and/or if your host's Control Broker is not started, then the e*Gate Monitor is not connected to that Control Broker. The end result is that the e*Gate Monitor appears not to operate with the affected schema.

The e*Gate Monitor display alerts you to these possible problems in the Component pane. **Figure 21 on page 141** shows an example of the e*Gate Monitor window when these problems are present.

Figure 21 e*Gate Monitor with inactive/unstarted components



For information on how to configure and modify the Control Broker, see "Control Broker: stccb" in the *e*Gate Integrator System Administration and Operations Guide*. For information on activating the Participating Hosts, see "Installer Service: stcinstd" in the *e*Gate Integrator System Administration and Operations Guide*.

3 **I cannot get the Alert Agent to send out e-mail. How can I debug this problem?**

Before running the Alert Agent in an active schema, test the e-mail channel independently via the e*Gate Alert Agent configuration tool, which opens the **e*Gate Alert Agent Control and Configuration** window (see the *e*Gate Integrator Alert Agent User's Guide* for more information). Running a test from this window does not require the e*Gate system to be running. When you execute this test, it creates a log file called **stcaacfg.log** in the **eGate\client\logs** directory. This file should be inspected for any errors. See the following examples:

In this case, the error message ("Logon failed") indicates that the Alert Agent failed to log on to the SMTP server.

```
--------------------------------------------------------------------------------
[16-Jan-2001 9:25:52.741] *** START LOG FOR: stcaacfg on localhost pid: 289 ***
--------------------------------------------------------------------------------
09:25:52.741 EWY  I 264 (VoiceTextDefaults.cxx:43): loading default
text-to-speech information from the windows registry.
09:25:52.751 EWY  I 264 (CallProcessing.cxx:127): Initializing call
processing engine.
```

```
09:25:52.751 EWY  I 264 (TAPILine.cxx:113): Initializing TAPI core.
09:25:52.771 EWY  I 264 (AlertAgentConfiguration.cxx:40): loading alert
agent configuration data from windows registry.
09:25:52.771 EWY  I 264 (CallbackConfiguration.cxx:39): loading alert agent
configuration data from windows registry.
09:25:52.771 EWY  I 264 (ActiveDeliveryChannel.cxx:63): loading default
alert table.
09:25:55.435 EWY  I 264 (MailDefaults.cxx:57): loading default mail
information from the windows registry.
09:26:03.557 EWY  I 264 (MailDefaults.cxx:57): loading default mail
information from the windows registry.
09:26:58.646 EWY  E 264 (Mail.cxx:253): Logon failed. return code : 12
E:0x20002000 (internal system)
09:29:40.469 EWY  I 264 (MailDefaults.cxx:57): loading default mail
information from the windows registry.
```

In the following example, the Alert Agent was able to log on to the SMTP server, but was not able to send an e-mail out.

```
15:30:09.758 EWY  I 341 (MailDefaults.cxx:57): loading default mail information
from the windows registry.
15:31:09.173 EWY  I 341 (MailDefaults.cxx:96): saving default mail
information to the windows registry.
15:31:12.638 EWY  I 341 (MailDefaults.cxx:57): loading default mail information
from the windows registry.
15:31:19.468 EWY  I 341 (Mail.cxx:263): Logged on to SMTP server.
15:31:19.468 EWY  I 341 (Mail.cxx:301): Recipient : Alert Agent Admin added.
15:31:19.468 EWY  I 341 (Mail.cxx:315): SMTP body and sender composed.
15:31:20.720 EWY  E 341 (Mail.cxx:285): SMTP send mail failed. return code :
8 E:0x20002000 (internal system)
15:31:20.920 EWY  I 341 (Mail.cxx:277): Logged off to SMTP
17:02:35.232 EWY  I 341 (CallScheduler.cxx:369): Purging all queued calls and
dropping active call.
17:02:35.232 EWY  I 341 (CallProcessing.cxx:143): shutting down call processing
engine.
17:02:35.232 EWY  I 341 (TAPILine.cxx:445): Shutting down TAPI core.
17:02:35.232 EWY  I 341 (VoiceTextInterface.cxx:515): Text-to-Speech conversion
stopped.
17:02:35.232 EWY  I 341 (VoiceTextInterface.cxx:523): Text-to-Speech engine
shutdown
------------------------------------------------------------------------------
[05-Jan-2001 17:02:35.232] *** END LOG FOR: stcaacfg on localhost pid: 308 ***
------------------------------------------------------------------------------
```

In any case, you can test the connectivity and the ability to send e-mail from your machine by bypassing the Alert Agent. To do this, type the following commands via a telnet session:

```
>telnet mail01.mydomain.com 25

220 MAIL01.mydomain.com ESMTP Server (Microsoft Exchange Internet Mail
Service 5.5.2653.13) ready helo mydomain.com
250 OK
mail from: <employee1@mydomain.com>
250 OK - mail from <employee1@mydomain.com>
rcpt to: <employee2@mydomain.com>
250 OK - Recipient <employee2@mydomain.com> data
354 Send data.  End with CRLF.CRLF
This is a test.  THIS IS A TEST!!!!!
.
250 OK
quit
221 closing connection
```

where:

The lines that start with the three-digit number are the replies back from the e-mail server. The other lines are the lines that you type. The above set of commands mimic what the Alert Agent e-mail channel does when it sends e-mail.

Note: *If your telnet session is connected, but you cannot see anything when you type, select* ***Terminal > Preferences*** *on the Telnet window. Then check the* ***Local Echo*** *checkbox and click* ***OK****.*

6.4 Log File FAQs

1 **I have started my e*Way, but I can't find the log file. What should I do?**

There are several things that could cause this.

If the e*Way is staying up, check the following:

- That the "Use log file" check box has been checked. To do this:

A Select the **Components** tab in the Enterprise Manager, then use the Navigator to select the e*Way that has a missing log file.

B On the toolbar, click  .

C Select the **Advanced** tab, then click **Log**.

D The **Logging Properties** dialog box for the e*Way appears. Make sure that the **Use log file** check box is checked.

- Your **.egate.store**. Make sure that you are looking into the correct directory.

If the e*Way is not staying up, check the following:

- Make sure that you have given your e*Way the correct executable name.

2 **I received the log message "File not found in Repository" but know that this cannot be as the component started in the Participating Host. What is wrong?**

Any time you start a component in the Participating Host, the Registry attempts to download all the **.dll** files that the component needs. If another component that uses the same **.dll** files is already running on that Participating Host, it prevents the Registry from downloading the **.dll** files as it cannot overwrite **.dll** files that are already in use. Since the Registry did not successfully download all the **.dll** files, it creates the confusing entry in the log file that you saw. This is not really an error, and does not hurt anything.

6.5 Editing FAQs

1 **When I create a new .def file for an e*Way, any existing .sc files that were previously defined for that e*Way are not updated with the new or changed variables and options. What should I do?**

Use the e*Way Editor to update the existing **.sc** file:

A Specify the new or changed variables or options for every instance of that e*Way.

B After reviewing all of the existing settings, specify any new settings and save the **.sc** file to update the **.def** file.

2 **When a Collaboration contains a list of collapsed Rules that exceed the maximum number (estimated to be approximately 800 rules), the Collaboration Rules Editor displays inconsistent behavior.**

Although this appears to be a memory limit problem, it cannot be corrected by adding additional memory as there is a Motif limit that prevents the additional memory from solving the problem.

Note: *This problem is only found in e*Gate release 4.1.2 and before.*

To edit large Collaborations:

A Use an external editor.

Note: *It is advisable to develop modular solutions that store Rules in separate files/ functions.*

3 **What should I do when I cannot edit my Monk Collaboration script?**

Most likely your Monk Collaboration script has more than 6500 lines, which is the maximum number of lines it can contain. If your script has surpassed the maximum limit, open it in a text editor and edit it there.

6.6 Vendor FAQs

1 **My ODBC drivers on AIX bind fields of type 'timestamp' inconsistently. What can I do to fix this problem?**

Contact the vendors of these products as this is an issue that SeeBeyond has no control over.

6.7 Troubleshooting Tips

1. **Can I edit a .monk file using an external editor?**

   Yes. However, when you do, make sure that all rules are inside the Lambda body. If you do not do this, the ETD Editor and Collaboration Editor GUIs are not able to load the file.

2. **Can I change the base directory and Registry port on UNIX and Windows?**

   Yes.

   A. Modify the directories displayed on the left to the new home directory path.

   B. Modify all startup scripts in **rc3.d** to the new home directory and modify the Registry port if necessary.

   C. Modify the **bourne** and **korn** startup scripts in UNIX.

   D. Start the Registry.

   E. Copy any schemas to the new name.

3. **Is it possible to run multiple independent Registry Services, Control Brokers, or Intelligent Queue (IQ) Managers on the same machine?**

   Yes, the trick is you have to change the port number. To do this, perform the following:

   A. For the Registry Host, add **-rp %_REGHOST%** to the startup command of any module (for Windows).

   B. For the necessary steps to set up the Control Broker, see the *e*Gate Integrator System Administration and Operations Guide*.

Note: *Each Control Broker must have a unique port. The default port is 24053.*

   C. For IQ Managers, reset the IQ Manager ports.

4. **Can I transport a schema to another Participating Host?**

   Yes. For information on how to perform this task, see "Migrating Schemas and Components" in the *e*Gate Integrator System Administration and Operations Guide*.

Appendix A

Status Notification Descriptions

This appendix provides a list of e*Gate system status notification descriptions (messages). These descriptions are informational and do not require action.

A.1 Status Notification Messages List

Table 31 provides a list of e*Gate status notification messages.

Table 31 Status Notification Messages List

Description/Message	Source of Notification
Alert Agent Connected To CB	An Alert Agent is able to connect to a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events.
Alert Agent Detects CB Responded	Control Broker responded to a query from an Alert Agent. This monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** Notification code 1014xx0x means normal operations. SeeBeyond-standard e*Gate components use no other codes for "resolved" monitoring Events.
BOB Up	An e*Gate component/process is up and running. **Additional Information** Notification code 1012xx0x means normal operations. SeeBeyond-standard e*Gate components use no other codes for "up" Monitoring. See **Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.
CB Connected To Registry	A Control Broker is able to connect to a Registry. Notification code 10161200 means normal operations. SeeBeyond-standard e*Gate components use no other codes for "connected" monitoring Events. This monitoring Event resolves "Unable to Connect" monitoring Events.
CB Connected To SNMP Agent	A Control Broker is able to connect to an SNMP Agent. This monitoring Event resolves "Unable to Connect" monitoring Events.

Table 31 Status Notification Messages List (Continued)

Description/Message	Source of Notification
CB Detects Alert Agent Responded	Alert Agent responded to a query from a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** Notification code 1014xx0x means normal operations. SeeBeyond-standard e*Gate components use no other codes for "resolved" monitoring Events.
CB Detects BOB Responded	BOB responded to a query from a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** Notification code 1014xx0x means normal operations. SeeBeyond-standard e*Gate components use no other codes for "resolved" monitoring Events.
CB Detects dgCmd Responded	dgCmd responded to a query from a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** Notification code 1014xx0x means normal operations. SeeBeyond-standard e*Gate components use no other codes for "resolved" monitoring Events.
CB Detects E*Gate Monitor Responded	e*Gate Monitor responded to a query from a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** Notification code 1014xx0x means normal operations. SeeBeyond-standard e*Gate components use no other codes for "resolved" monitoring Events.
CB Detects E*Way Responded	e*Way responded to a query from a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** Notification code 1014xx0x means normal operations. SeeBeyond-standard e*Gate components use no other codes for "resolved" monitoring Events.
CB Detects Enterprise Manager Responded	Enterprise Manager responded to a query from a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** Notification code 1014xx0x means normal operations. SeeBeyond-standard e*Gate components use no other codes for "resolved" monitoring Events.

Table 31 Status Notification Messages List (Continued)

Description/Message	Source of Notification
CB Detects IQManager Responded	IQ Manager responded to a query from a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** Notification code 1014xx0x means normal operations. SeeBeyond-standard e*Gate components use no other codes for "resolved" monitoring Events.
CB Detects SNMP Agent Responded	SNMP Agent responded to a query from a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** Notification code 1014xx0x means normal operations. SeeBeyond-standard e*Gate components use no other codes for "resolved" monitoring Events.
CB Detects User Agent Responded	User Agent responded to a query from a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** Notification code 1014xx0x means normal operations. SeeBeyond-standard e*Gate components use no other codes for "resolved" monitoring Events.
e*Gate Monitor Connected to CB	The e*Gate Monitor is able to connect to the Control Broker. The system is operating normally.
E*Gate Monitor Connected To CB	An e*Gate Monitor is able to connect to a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events.
e*Gate Monitor Detects CB Responsive	The Control Broker responded to a query from the e*Gate Monitor. This monitoring Event resolves the monitoring Event "e*Gate Monitor Detects CB Unresponsive."
E*Way Connected to External	An e*Way established a connection to an external component. This monitoring Event resolves "Unable to Connect" monitoring Events. Notification code 1036dz00 means normal operations. SeeBeyond-supplied e*Gate components use no other codes for "connected" monitoring Events.
E*Way Up	An e*Gate component/process is up and running. **Additional Information** Notification code 1012xx0x means normal operations. SeeBeyond-standard e*Gate components use no other codes for "up" Monitoring. See **Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.
Element is up	An e*Gate component/process is up and running.

Table 31 Status Notification Messages List (Continued)

Description/Message	Source of Notification
Element State Connected	A monitor is able to connect to a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events.
Element State Up	An e*Gate component/process is up and running. **Additional Information** Notification code 1012*xx*0*x* means normal operations. SeeBeyond-standard e*Gate components use no other codes for "up" Monitoring. See **Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.
Enterprise Manager Connected To CB	An Enterprise Manager is able to connect to a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events.
Enterprise Manager Detects CB Responded	Control Broker responded to a query from an Enterprise Manager. his monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** Notification code 1014xx0x means normal operations. SeeBeyond-standard e*Gate components use no other codes for "resolved" monitoring Events.
External State Connected	An e*Way established a connection to an external component. This monitoring Event resolves "Unable to Connect" monitoring Events.
Internal State Connected	A monitor is able to connect to a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events.
Internal State Responded	An e*Gate component/process responded to a query from a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** See **Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.
Internal State Timer	The Control Broker has sent a "Timer" monitoring Event according to a user-defined schedule. This monitoring Event is generated by a user-settable parameter in the Control Broker properties. **Additional Information** See **Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.

Table 31 Status Notification Messages List (Continued)

Description/Message	Source of Notification
Internal State Up	An e*Gate component/process is up and running. **Additional Information** Notification code 1082xx0x means normal operations. SeeBeyond-supplied e*Gate components use no other codes for "up" monitoring Events.
IQManager Up	An e*Gate component/process is up and running. **Additional Information** Notification code 1012xx0x means normal operations. SeeBeyond-standard e*Gate components use no other codes for "up" Monitoring. See Table 17 on page 89 for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.
Resource Timer	The Control Broker has sent a "Timer" monitoring Event according to a user-defined schedule. This monitoring Event is generated by a parameter that the user can set in the Control Broker properties. **Additional Information** Notification code 106T0000 means Standard Timer Event code. This is the only monitoring Event of this type that is generated by a standard e*Gate component. See Table 17 on page 89 for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.
Script Connected	A monitor is able to connect to a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** See Table 17 on page 89 for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.
Script Responded	An e*Gate component/process responded to a query from a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** See Table 17 on page 89 for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.

Table 31 Status Notification Messages List (Continued)

Description/Message	Source of Notification
Script Timer	The Control Broker has sent a "Timer" monitoring Event according to a user-defined schedule. This monitoring Event is generated by a user-settable parameter in the Control Broker properties. **Additional Information** See Table 17 on page 89 for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.
Script Up	An e*Gate component/process is up and running. **Additional Information** Notification code 1092xx0x means normal operations. SeeBeyond-supplied e*Gate components use no other codes for "up" monitoring Events.
SNMP Agent Detects CB Responded	Control Broker responded to a query from an SNMP Agent. This monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** Notification code 1014xx0x means normal operations. SeeBeyond-standard e*Gate components use no other codes for "resolved" monitoring Events.
State Responded	An e*Gate component/process responded to a query from a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** Notification code 1014*xx*0*x* means normal operations. SeeBeyond-standard e*Gate components use no other codes for "resolved" monitoring Events.
STCCmd Connected To CB	An STCCmd is able to connect to a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events.
Timer Event	The Control Broker has sent a "Timer" monitoring Event according to a user-defined schedule. Notification code 106T0000 is Standard Timer Event code. This is the only monitoring Event of this type that is generated by a standard e*Gate component. This monitoring Event is generated by a parameter that the user can set in the Control Broker properties. **Additional Information** See Table 17 on page 89 for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components.
User Agent Connected To CB	A User Agent is able to connect to a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events.

Table 31 Status Notification Messages List (Continued)

Description/Message	Source of Notification
User Agent Detects CB Responded	Control Broker responded to a query from a User Agent. This monitoring Event resolves "Unable to Connect" monitoring Events. **Additional Information** Notification code 1014xx0x means normal operations. SeeBeyond-standard e*Gate components use no other codes for "resolved" monitoring Events.
User Defined	Generated by a custom code defined by the user.
User Defined Connected	Standard definition of "Connected" monitoring Events: A monitor is able to connect to a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events. This monitoring Event has been generated by a user-defined component. **Additional Information** See **Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components. Note that in this instance notification code 1076xx0x means standard code for "Connected" monitoring.
User Defined Responded	Standard definition of "Responded" monitoring Events: An e*Gate component/process responded to a query from a Control Broker. This monitoring Event resolves "Unable to Connect" monitoring Events. This monitoring Event has been generated by a user-defined component. **Additional Information** See **Table 17 on page 89** for the meaning of the seventh byte in the notification code, as it contains additional information about codes for this monitoring Event that are not generated by standard e*Gate components. Note that in this instance, notification code 1074xx0x means standard code for "responded" events.
User Defined Up	Standard definition of "Up" monitoring Events: An e*Gate component/process is up and running. This monitoring Event has been generated by a user-defined component. **Additional Information** Notification code 1072xx0x means Normal operations. SeeBeyond-supplied e*Gate components use no other codes for "up" monitoring Events.

A.2 Custom Status Notifications

The e*Gate system provides you with features that allow you to create your own custom status notifications. See section for details.

Appendix B

Customizing Default Notification Routing

The default **Notification.tsc** file shipped with e*Gate is designed to enable you to create custom Notification Routing scripts quickly and easily. This appendix explains how to use the script contained in this file.

B.1 About the Default Script

Default routing sends notifications only to logged-in monitors. However, you can send notifications through any of the available channels simply by changing a few conditional Collaboration Rules and entering your site-specific channel information (such as e-Mail addresses or pager numbers).

Note: *Before your e*Gate system can send notifications through e-Mail, pager, print, fax, or voice channels, you must run an Alert Agent such as the e*Gate Alert Agent. See "Non-interactive Monitoring" in the* ***e*Gate Integrator User's Guide*** *for more information. For more information on the Alert Agent, see the* ***e*Gate Integrator Alert Agent User's Guide****.*

A script contained in the **Notification.tsc** file allows you to create custom Notification Routings.

B.1.1 Structure

The default **Notification.tsc** script is divided into the following sections:

- The initial section defines global variables and sets basic parameters for the Notification Routing.
- The second section routes notifications to logged-in interactive monitors.
- The remaining sections handle Notification Routing for the other available channels. Each channel is handled by a separate section.

Each section begins with a block of Comment rules.

B.1.2 Important Principles for Editing Routing Scripts

The Enterprise Manager's Collaboration Rules Editor contains extensive information about how to use the Editor, as well as the usage and syntax of each Collaboration Rule. This guide assumes that anyone who edits the Notification Routing script is already familiar with the Collaboration Rules Editor and the basics of e*Gate Event processing.

This appendix does not discuss procedures for using the Collaboration Rules Editor. However, a few important principles bear repeating:

- Most of the work of the Notification Routing script is done using Copy rules. Strings to be copied *must* be surrounded by double quotes (for example, `"egate@yourcompany.com"`). If a string must include a double quote, escape it with a backslash (`\"`).
- Use the toolbar's **Validate** tool to check any changes that you make. While this tool will not catch logic errors, it will catch syntax errors.
- If you use the Monk **regex** function, all special characters in the regular expression must be escaped with back slashes. For example, the regular expression **"[246]"** must be entered as **"\[246\]"**.
- Many instructions tell you to use the copy and paste features. You can only copy/paste one rule at a time. However, if you copy a rule (such as the If rule) that contains other rules in a block, the entire block of rules will be copied.
- The routing mechanism requires that certain nodes within a notification Event have non-null values (for example, an initial escalation number and the escalation interval). These nodes are so indicated by comments in the script and in this chapter. You can change the values of these nodes to meet your site's requirements, but the nodes cannot be omitted.
- There is no undo function within the Collaboration Rules Editor. Save backup copies of your script before you make significant changes.
- Finally, and most importantly, *all* nodes within the Notification Routing script must have valid values or *no* notifications are sent.

B.1.3 Using If Rules to Enable Channels

Even though information required to send notifications through all available channels exists within the default script, most of that information is not used. Except for the section that sends notifications to logged-in interactive monitors, each channel's section is enclosed with an If rule that reads:

```
IF #f
```

The **#f** construction evaluates to "false." This specifies that the result of the If rule's test is false; thus, that If rule's contents are never executed, effectively commenting out its block of rules.

You could, if desired, simply change this construction to:

```
IF #t
```

The **#t** construction evaluates to "true." Using this construction would execute the block of rules within the If rule, enabling you to send notifications through a channel unconditionally.

Even though this may be useful for testing, this method would send all notifications unconditionally through a channel. While this sort of unconditional transmission is appropriate for logged-in monitors, it is less useful for the other channels. For example, there would be no reason to send an e-mail announcing that a particular component was up and functioning normally.

To create condition-specific notification rules

1. Use the copy and paste features to create a copy of the If-rule block that sends notifications through a specific channel. This ensures not only that you have placeholders for all the information that you need, but that you keep the original copy for reference.
2. Replace the If rule's **#f** statement with a test for the condition for which you want to send a notification.
3. Modify or replace the sample notification destinations within the If-rule block with your own site-specific information.

For example, to create a rule that sends an e-Mail message whenever an Intelligent Queue (IQ) Manager shuts down for any reason:

1. Use the copy and paste features to create a copy of the original If-rule block that sends e-Mail notifications.
2. Change the If rule to read as follows:

```
(regex "10113\.\*" ~input%EventMsg.Msg.EventHeader.eventCode)
```

   This rule tests whether the monitoring-Event code begins with 10113, a monitoring-Event code indicating that a component went down (1011), and that the component in question was the IQ Manager (3).
3. Change the quoted strings for sender and recipient information to appropriate values for your installation (be sure to keep the quotation marks intact).

See **"Sample If Rules" on page 170** for more examples of how to use If rules to send notifications selectively.

B.1.4 Deciding Which Rules To Copy

The directions in this appendix instruct you to use copy and paste to make new copies of certain rules that pertain to individual channels as follows:

- If you want to base the additional notifications on the *same* monitoring Event, copy and paste individual rules within the same If-rule block.
- If you want to base the additional notifications on *different* monitoring Events, copy and paste the entire If-rule block. Then, change the new If rule to match the new condition for which you want to send the notifications.

Note: *The instructions in this appendix assume that you will create new Notification recipients/escalations for the same Monitoring Event. Make the appropriate substitutions in the copy and paste instructions if you want to do otherwise.*

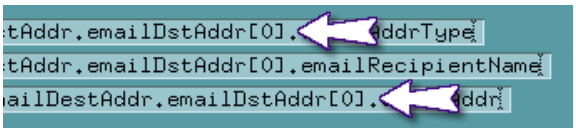
B.1.5 Specifying Additional Recipients

The default **Notification.tsc** file generally specifies a single recipient for each notification (for example, a single e-Mail address is the destination for e-Mail notifications). You can specify additional recipients if the channel supports them. All channels except the Script channel support multiple recipients.

You can easily specify additional recipients using the following procedure:

1. Use the copy and paste features to make a second copy of a rule that specifies a single recipient.
2. Change the instance number for the recipient address. Recipient instance numbers appear in square brackets after the text **Addr**, as in **emailDstAddr*[0]*** (see Figure 22).
3. Change the quoted recipient information to match your site's requirements (see Figure 22).

Figure 22 Recipient instance numbers



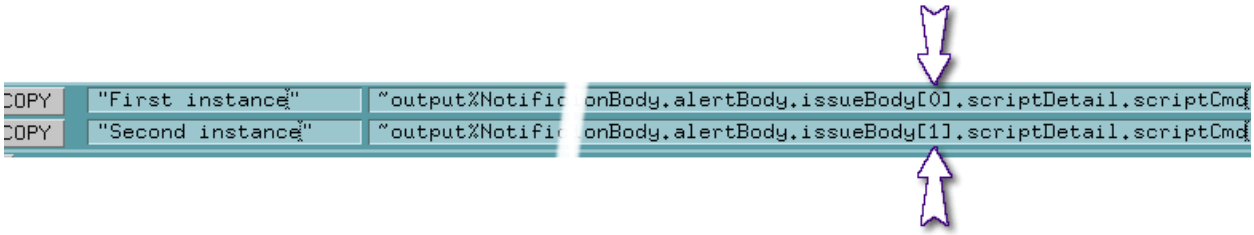
Channel-specific details can be found in the discussion of each channel's section of the routing script.

B.1.6 Specifying Escalations

You can configure the Notification Routing script to escalate a notification, sending information to different recipients for each escalation level. The escalation level is specified within a node's **issueBody** instance number (for example, **issueBody*[0]***). Escalations are numbered beginning with zero.

For example, in **Figure 23 on page 158**, the **issueBody** instance numbers determine that **First cmd** will be issued during the initial (zeroth) escalation, and that **Second cmd** will be issued during the next (first) escalation (see **Figure 23 on page 158**).

Figure 23 issueBody instance numbers



B.1.7 About the Figures

The figures in this chapter illustrate portions of the **Notification.tsc** script as it is seen in the Collaboration Rules Editor. Because the names of nodes in the Notification-Message Event Definition are so long, it would be very difficult to print them in their entirety at a size that would both fit the page and be large enough to read details comfortably.

All the output message nodes in the body of the notification Event begin with the following string:

```
~output%NotificationMessage.Msg.NotificationBody.AlertBody
```

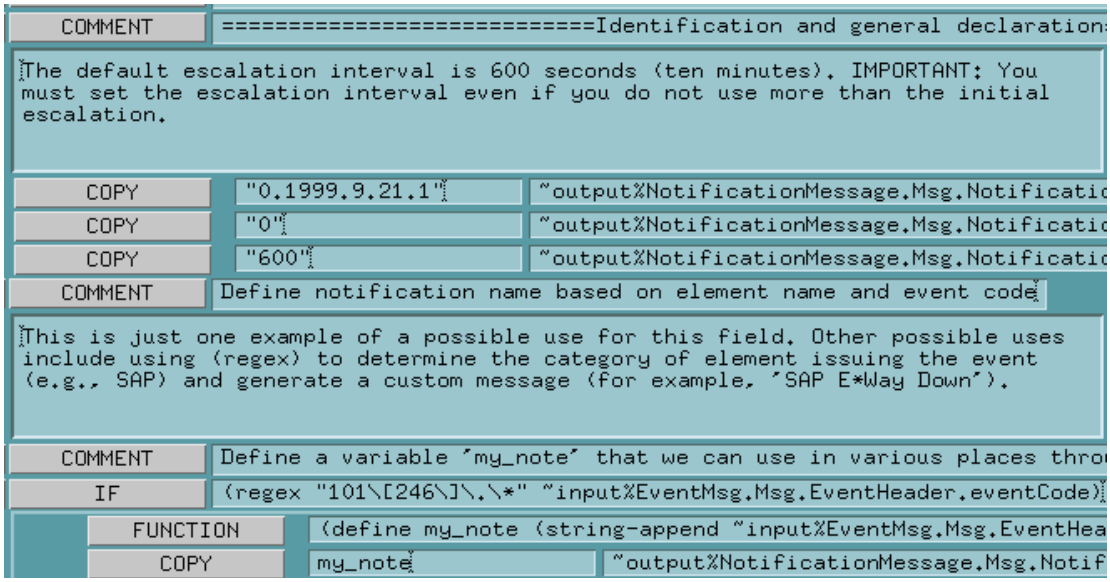
The figures in this appendix omit portions of this common string, so you can read the unique portions of the string at the end of the node name (as in **Figure 23 on page 158**). Do not retype portions of the **Notification.tsc** file simply because portions of the output-node name have been omitted from the figures.

B.2 Initial Declarations

The initial declarations in the default **Notification.tsc** script accomplish the following operations:

- Declares the internal version number of this Notification script.
- Sets the escalation interval to 600 sec (10 min).
- Sets the initial escalation number to zero.
- Defines a variable *my_note* that illustrates how to combine data from different nodes within the monitoring Event that can be reused throughout the notification Event (see **Figure 24 on page 159**).

Figure 24 Initial declarations



Entries you can change

- The version number. Replace it with your own version number.
- The escalation interval. Replace it with the number of seconds at which you want to send escalated notifications.

Important: *Even though you can set this interval to any value, you must set an escalation interval. Do not delete this rule.*

- The definition of the *my_note* variable. If you do not wish to use it, you can delete it entirely; however, if you do, be sure to delete references to it that appear later in this script.

Note: *SeeBeyond recommends that you do not make any changes to entries in this section until you become thoroughly familiar with Notification Routing.*

Entries you do not change

The initial escalation number. Leave this number at zero.

About the *my_note* variable

The default Notification script uses the Monk function **string-append** to create a string that reads *[Component name] reports [Monitoring-Event description]*. This string is stored in the variable *my_note*. Because certain monitoring Events store the description information in different locations, the Monk function **regex** parses the Event code and composes the variable appropriately.

The *my_note* string is used in the following locations:

- It is copied to the **NotificationName** node, which is displayed in the Alert and Status panes of the e*Gate Monitor.
- It is copied to the **emailText** node, which places the string within the body of e-Mail messages.

You can include the *my_note* variable in any notification channel where a short description of a monitoring Event is necessary. Notice, however, that since *my_note* is a variable, you do *not* enclose it in quotes when copying it to a notification node. Doing so would copy the literal string *my_note* to the node rather than the contents of the *my_note* variable (see Figure 25).

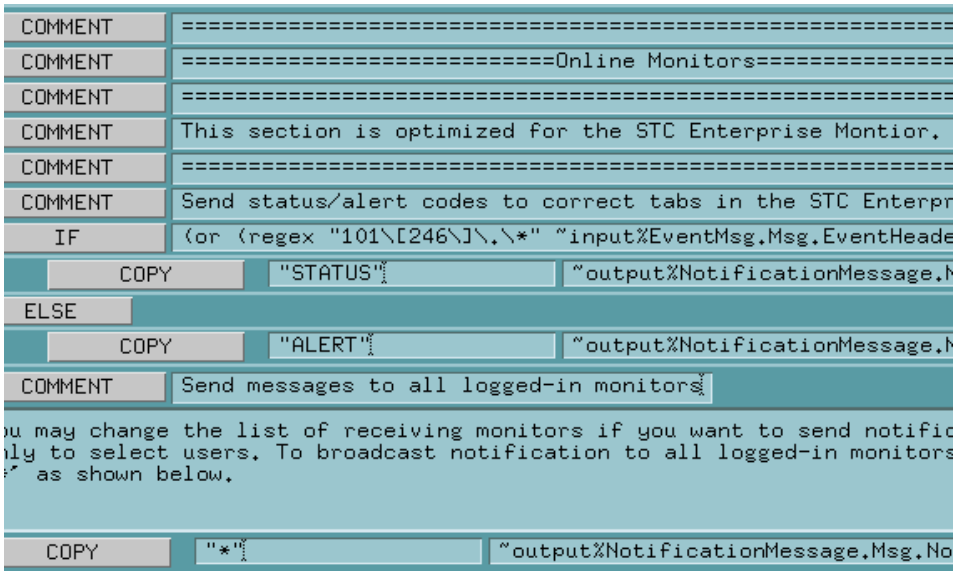
Figure 25 Using the my_note variable

B.3 Monitor Channel

The online monitors section of the script accomplishes the following operations:

- Sets the **NotificationUsage** value to **S** for status monitoring Events (those with codes that begin with 1012, 1014, or 1016, and other specific non-urgent monitoring Events) and to **A** for Alert monitoring Events (all other monitoring Events).
- Sends notifications on the zeroth (initial) escalation to all logged-in monitors (see Figure 26).

Figure 26 Monitor channel



Entries you can change

The list of logged-in users. The default, *, sends notifications to all monitors. You can replace this with a list of e*Gate user names, or devise an escalation strategy that sends certain escalations to certain users.

Note: *The "*" means any e*Gate user. You can also specify explicit user names.*

Entries you do not change

The If rule that assigns the status or Alert **NotificationUsage** value. You can only change this logic if you want different Events to be sent to the e*Gate Monitor's Status pane than are specified by the default script, and only if you are thoroughly familiar with Notification Routing.

The default script sends the following Events to the Status pane of the e*Gate Monitor:

- Monitoring Events whose Event code begins with 1012, 1014, or 1016.
- Configuration change Alerts.
- Tally Events (one Tally Event is sent per thousand data Events that a component processes).

To specify recipients within a new escalation

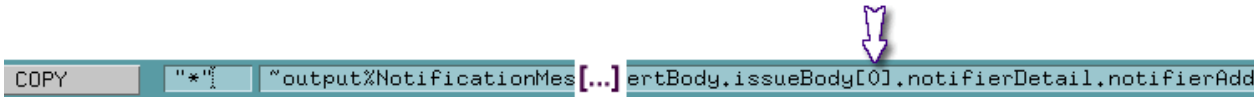
1 Use the copy and paste features to create a new copy of the rule that sends notifications to all logged-in monitors (see Figure 27).

Figure 27 Monitor-channel default recipient

2 For the zeroth (initial) escalation, change * to the e*Gate user name that should receive the initial escalation.

3 For each additional escalation, change * to the e*Gate user name that should receive the initial escalation. Then, change the **issueBody***[n]* escalation number to the escalation at which that user should receive notification (see Figure 28).

Figure 28 Monitor escalation instance numbers



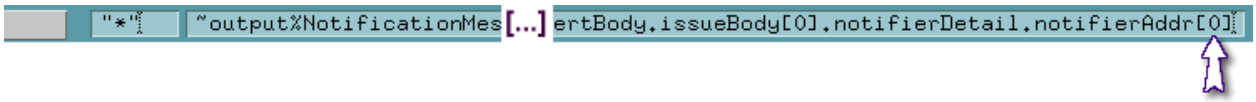
To specify additional recipients within an existing escalation

1 Use the copy and paste features to create a new copy of the existing rule that sends the desired notification.

2 In the first argument to the Copy rule, enter the e*Gate name of a user or a comma-delimited list of e*Gate users, enclosed in double quotes.

3 Increment the **notifierAddr*[n]*** instance number by one (see Figure 29). For example, if the rule you are editing has the issue number **notifierAddr*[0]***, change the number to **notifierAddr*[1]***.

Figure 29 Monitor recipients instance numbers



B.4 e-mail Channel

The e-mail section of the default **Notification.tsc** script specifies the following:

- Name and e-mail address to appear as the sender of the notification message.
- Name and e-mail address to appear in the e-mail message's **To** field.
- Subject, body, and optional text file for the e-mail message.
- User name and password for the SMTP server used to deliver the e-mail message (see Figure 30).

Figure 30 e-mail channel

Note: *In the e-mail channel example, the "File name" needs to be set to an empty string. The "File name" has been reserved for future use.*

The subject of the message is composed by copying two strings to the same node (**emailSubject**) within the notification Event. This effectively concatenates the two strings. You can use this technique whenever you want to create a complex string within the outgoing notification message.

Entries you can change

You can change all of the entries within this section; you can also delete any entries that are not appropriate (for example, you do not have to send a text file if you do not want to).

However, for *each* e-mail recipient, you must specify *all three* of the following items (only the final component of the entire node name is listed):

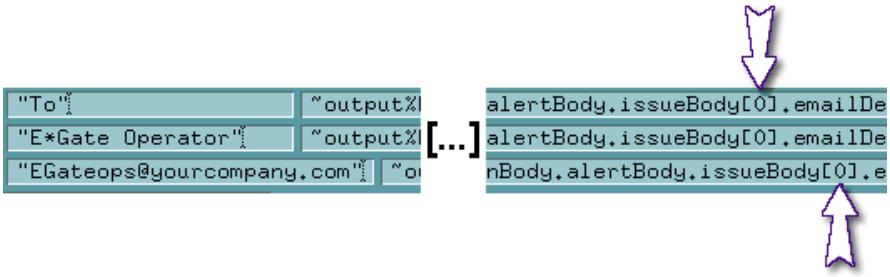
- **emailAddrType**. One of **To**, **Cc**, or **Bcc**.
- **emailRecipientName**. The "human name" of the recipient, such as "Joe Smith."
- **emailAddr**. The e-mail address of the recipient, such as **js@yourcompany.com**.

Each recipient must also have a unique **emailDstAddr***[n]* instance number.

To specify a recipient within a new escalation

1. Use the copy and paste features to create a new copy of the recipient information rules. You must copy each rule individually, as there is no way to copy more than one rule at a time.
2. Change recipient information as required. Be sure that you enclose all strings within double quotes, and that at least one recipient is designated to be within the e-mail message's **To** field.
3. In the three rules you just created, change the **issueBody***[n]* escalation number to the escalation at which that user should receive notification (see Figure 31).

Figure 31 e-mail Escalation instance number



To specify additional recipients within an existing escalation

1. Use the copy and paste features to create a new copy of the recipient information rules (see **Figure 32 on page 164**). You must copy each rule individually, as there is no way to copy more than one rule at once.

2 Change recipient information as required. Be sure that you enclose all strings within double quotes, and that at least one recipient is designated to be within the e-mail message's **To** field.

3 Increment the **emailDstAddr***[n]* instance number by one (see Figure 32). For example, if the rule you are editing has the issue number **emailDstAddr***[0]*, change the number to **emailDstAddr***[1]*.

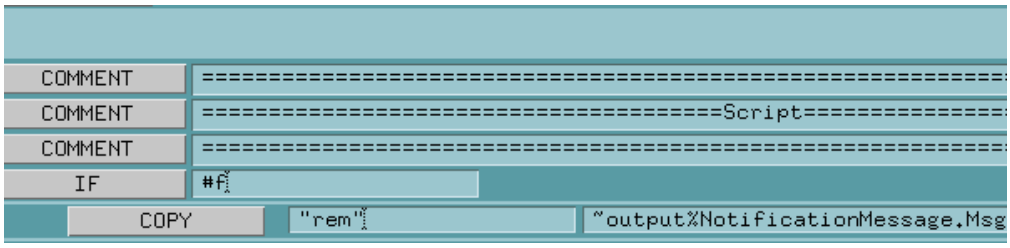
Figure 32 e-mail recipient instance numbers

Note: *You must have the e*Gate Alert Agent (or similar agent) properly configured and running before any e-Mail notifications can be delivered.*

B.5 Script Channel

The script channel specifies a command file to run. Unlike the other channels, which require some sort of monitor to deliver their notifications, the script channel's command files are executed directly by the Control Broker (see Figure 33).

Figure 33 Script channel



Entries you can change

There is only one entry: the name of the command or script file to be executed. You can change this value.

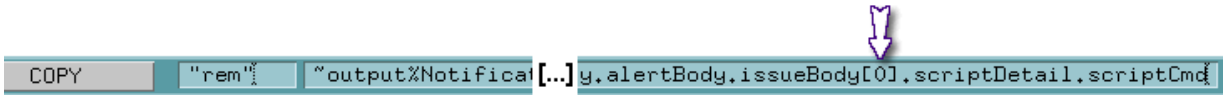
To specify a command file to be executed for a new escalation

1 Use the copy and paste features to create a new copy of the Copy rule.

2 Change the **rem** command to the command you want to issue, including any necessary arguments. Be sure to enclose the entire string within double quotes. If you need to include quotation marks as part of the command line, escape them with back slashes (such as \" or \').

3 In the rule you just created, change the **issueBody***[n]* escalation number to the escalation at which that user should receive notification (see Figure 34).

Figure 34 Script channel escalation number

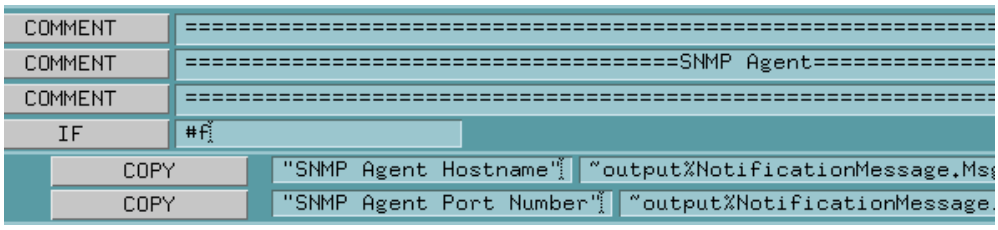


B.6 SNMP Agent Channel

The SNMP Agent channel carries the following information necessary to route a notification to an SNMP Agent:

- The name of the network host on which the SNMP Agent is running.
- The TCP/IP port over which the Control Broker communicates with the SNMP Agent, as specified in the SNMP Agent's property sheet in the Enterprise Manager and the SNMP Agent's configuration file (see Figure 35).

Figure 35 SNMP Agent channel



Entries you can change

You can change all the entries for this channel.

To change the host name or port number

Type the appropriate information between the quotation marks.

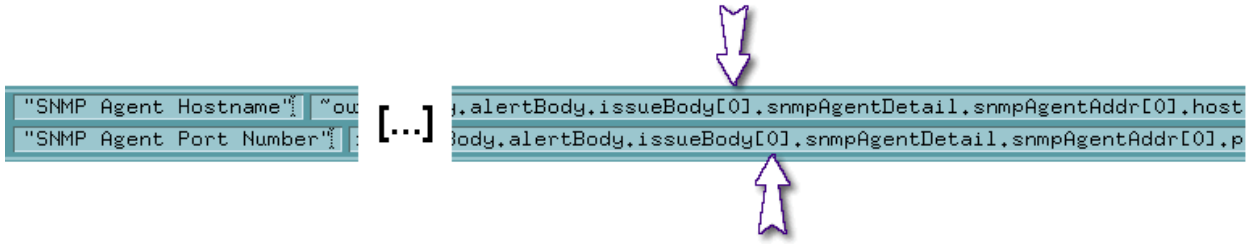
To specify a recipient within a new escalation

1 Use the copy and paste features to create a new copy of the two rules within the SNMP Agent channel's IF block.

2 Change the recipient information as required.

3 In each of the new rules you just created, change the **issueBody***[n]* escalation number to the escalation at which that Agent should receive notification (see Figure 36).

Figure 36 SNMP Agent escalation numbers



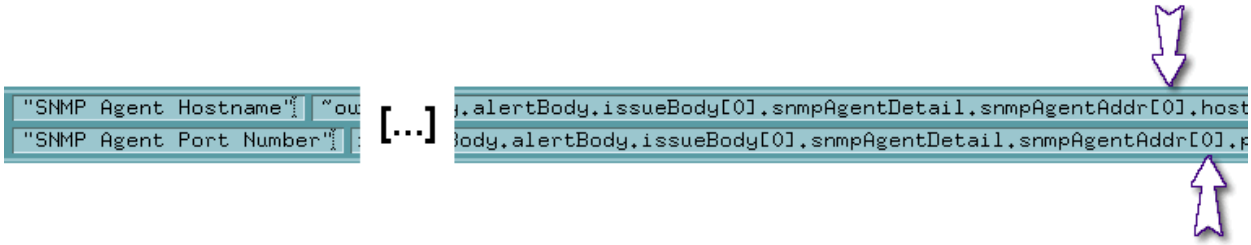
To specify additional recipients within an existing escalation

1 Use the copy and paste features to create a new copy of the two SNMP Agent rules. You must copy each rule individually, as there is no way to copy both rules at once.

2 Change recipient information as required. Be sure that you enclose all strings within double quotes.

3 Increment the **snmpAgentAddr***[n]* instance number by one (see Figure 37). For example, if the rule you are editing has the issue number **snmpAgentAddr***[0]*, change the number to **snmpAgentAddr***[1]*.

Figure 37 SNMP Agent recipient instance numbers



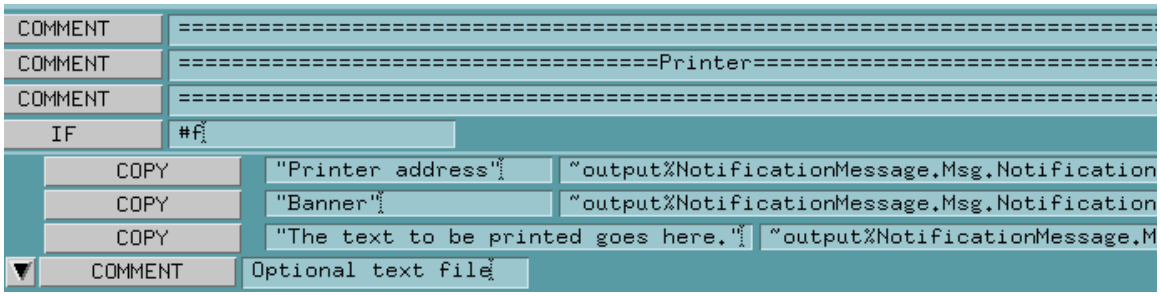
Note: *For more information on the e*Gate SNMP Agent, see the* ***e*Gate Integrator SNMP Agent User's Guide****.*

B.7 Printer Channel

The printer channel carries the following information to print notifications (for an example, see Figure 38):

- The system name of the printer.
- Text to appear on a banner.
- Text to be printed as the body of the printout.
- Name of an ASCII text file to be printed (optional).

Figure 38 Print channel



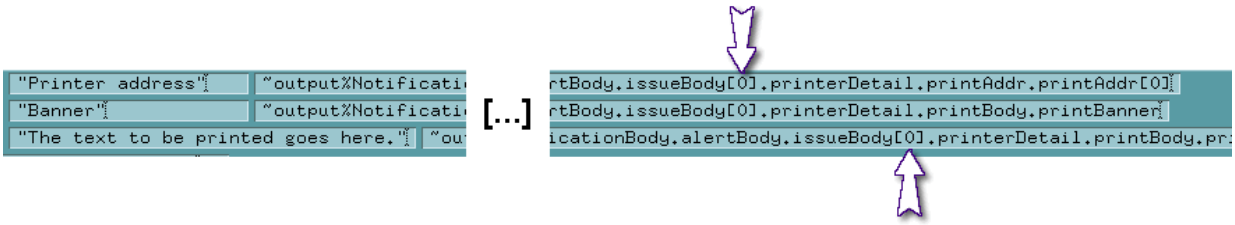
Entries you can change

You can change all the entries for this channel. If you do not want to print the optional text file, you can delete the rule that specifies the file name.

To specify a recipient within a new escalation

1. Use the copy and paste features to create a new copy of each rule within the print channel's IF block. You must copy each rule individually. There is no way to copy more than one rule at once.
2. Change the recipient information as required.
3. In each of the new rules you just created, change the **issueBody***[n]* escalation number to the escalation at which that notification should be printed (see Figure 39).

Figure 39 Printer channel escalation numbers



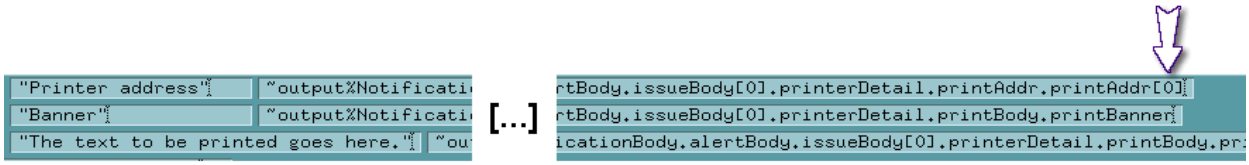
To specify additional recipients within an existing escalation

1. Use the copy and paste features to create a new copy of the "Printer address" rule.

2 Change the recipient information as required. Be sure that you enclose all strings within double quotes.

3 Increment the **printAddr***[n]* instance number by one (see Figure 40). For example, if the rule you are editing has the issue number **printAddr***[0]*, change the number to **printAddr***[1]*.

Figure 40 Printer channel recipient instance number



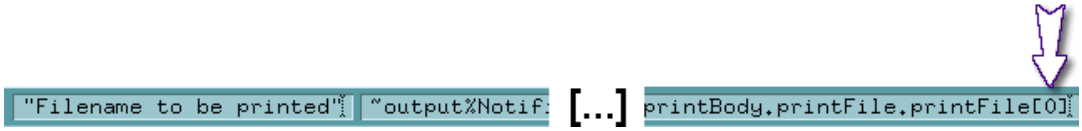
To specify additional files to be printed within an existing escalation

1 Use the copy and paste features to create a new copy of the "Filename to be printed" rule.

2 Change file-name information as required. Be sure to enclose the file name in double quotes.

3 Increment the **printFile***[n]* instance number by one (see Figure 41). For example, if the rule you are editing has the issue number **printFile***[0]*, change the number to **printFile***[1]*.

Figure 41 Printer channel file-to-print instance number



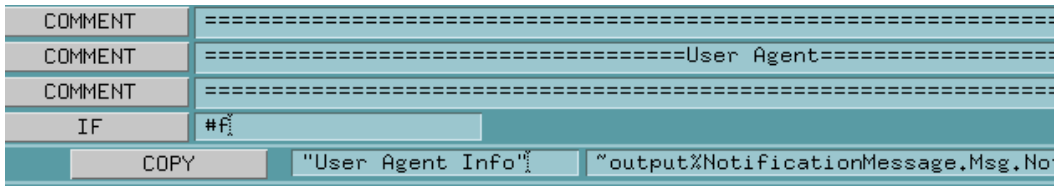
Note: *You must have the e*Gate Alert Agent (or similar agent) properly configured and running before any printer notifications can be delivered.*

B.8 User Agent Channel

The User Agent channel contains the information required to deliver notifications via a user-written monitoring agent.

This channel sends a single value, that is, the address information required by the user agent (see Figure 42).

Figure 42 User Agent channel



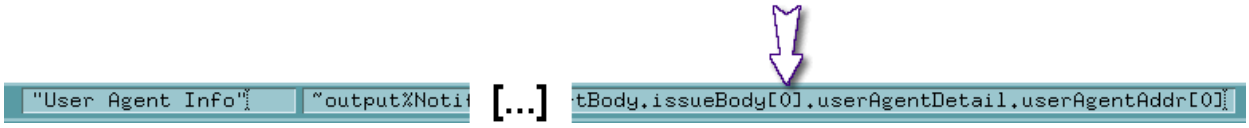
Entries you can change

You can change the entry for this channel.

To specify a recipient within a new escalation

1. Use the copy and paste features to create a new copy of the "User Agent Info" rule.
2. Change the information as required.
3. In the new rule you just created, change the **issueBody***[n]* escalation number to the escalation at which that Agent should receive notification (see Figure 43).

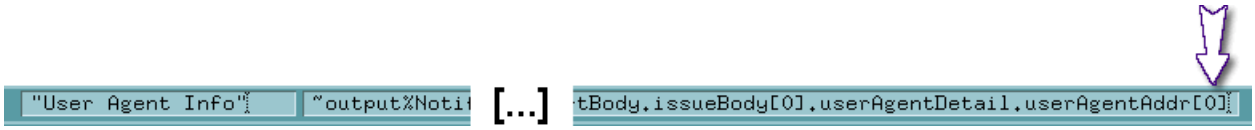
Figure 43 User Agent escalation numbers



To specify additional recipients within an existing escalation

1. Use the copy and paste features to create a new copy of the "User Agent Info" rule.
2. Change the information as required. Be sure that you enclose all strings within double quotes.
3. Increment the **userAgentAddr***[n]* instance number by one (see Figure 44). For example, if the rule you are editing has the issue number **userAgentAddr***[0]*, change the number to **userAgentAddr***[1]*.

Figure 44 User Agent recipient instance numbers



B.9 Sample If Rules

This section illustrates some sample If rules you can use when creating your custom notification script. You can modify these examples to meet your site's requirements or use them as a basis for building more complex rules of your own.

Table 32 (see later in this section) lists the following information:

- The **Purpose** column describes the test the If rule performs.
- The **Test** column contains the test to be applied by the If rule.
- The **Monitoring-Event Node** column contains the name of the node within the Monitoring-Event header to which the If rule's test should be applied. Only the final element of the complete node name is listed. Unless otherwise specified, the complete node name begins with this string:

  ```
  ~input%EventMsg.Msg.EventHeader
  ```

  For example, if the table lists "elementName," the complete node name is

  ```
  ~input%EventMsg.Msg.EventHeader.elementName
  ```

See **Table 7 on page 77** for a complete list of nodes within the Monitoring-Event Header. Table 32 shows a list of sample If rules.

Table 32 Sample If Rules

Purpose	Test	Monitoring-Event Node
Fatal errors only	string=? "F"	severity
Informational Events only	string=? "I"″	severity
Events from a specific host	string=? "hostname"	hostName
Events from a specific element	string=? "elementname"	elementName
Events from a class of element, such as BOBs or e*Way Intelligent Adapters	string=? "typecode" (see **Table 17 on page 89** for the element-type codes)	elementType
All Alerts	string=? "ALERT"	NotificationUsage
All status messages	string=? "STATUS"	NotificationUsage

For example, to test for fatal errors, use this expression:

```
(string=? "F" ~input%EventMsg.Msg.EventHeader.severity)
```

You can define more complex conditions using the (and) operator. The syntax of the (and) operator is:

```
(and (first-condition) (second-condition)...(nth condition))
```

For example, to test for fatal errors from IQ Managers, you would use this expression (printed on more than one line for clarity, but entered as a single line in the If rule):

```
(and (string=? "F" ~input%EventMsg.Msg.EventHeader.severity)
  (string=? "3" ~input%EventMsg.Msg.EventHeader.elementType))
```

Depending on the conditions for which you are testing, you may be able to get the same results using **regex** to parse the error code as you can from a complex conditional expression.

For example, the Monitoring-Event code for fatal IQ Manager errors matches the regular expression **101[357]3.***, so you can achieve the same results as the previous example using the following expression:

```
(regex "101\[357\]3\.\*"
       ~input%EventMsg.Msg.EventHeader.eventCode)
```

Appendix C

Java Error Messages

This appendix provides the Java error messages that e*Gate can generate. These error messages appear in the e*Gate Monitor.

C.1 Java Error Messages and Recommended Actions

Table 33 lists Java error messages, along with the problem or possible cause, and recommended actions.

Table 33 Java Error Messages

Java Error Message	Problem or Possible Cause	Recommended Action
Attempt to commit a non-transacted session	The user sent a commit command as a non-transacted session.	There is no commit command for a non-transacted session. Do not use this method to send a commit command.
Attempt to rollback a non-transacted session	The user sent a rollback command as a non-transacted session.	There is no commit command for a non-transacted session. Do not use this method to send a rollback command.
Attempt to recover a transacted session	The user tried to recover a transacted session.	Recovery must be performed in a non-transacted session.
Topic session no long connect to server	The session is already closed. A closed session cannot be used to connect to the server.	You must use an open session to connect to the server.
STCBytesMessage is read-only; can't write boolean	The user attempted to add a boolean string into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a boolean to this message.

Table 33 Java Error Messages (Continued)

Java Error Message	Problem or Possible Cause	Recommended Action
STCBytesMessage is read-only; can't write byte	The user attempted to add a byte into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a byte to this message.
STCBytesMessage is read-only; can't write short	The user attempted to add a short into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a short to this message.
STCBytesMessage is read-only; can't write char	The user attempted to add a character into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a char to this message.
STCBytesMessage is read-only; can't write int	The user attempted to add an integer into a read-only message.	Select a message that has write capability. Use **clearBody()** to write an int to this message.
STCBytesMessage is read-only; can't write long	The user attempted to add a long into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a long to this message.
STCBytesMessage is read-only; can't write float	The user attempted to add a float into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a float to this message.
STCBytesMessage is read-only; can't write double	The user attempted to add a double into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a double to this message.
STCBytesMessage is read-only; can't write UTF	The user attempted to add an UTF format into a read-only message.	Select a message that has write capability. Use **clearBody()** to write an UTF format to this message.

Table 33 Java Error Messages (Continued)

Java Error Message	Problem or Possible Cause	Recommended Action
UTFDataFormatException: String is too big to convert to UTF	There is a limited size for an UTF format string (65535 bytes). The characters in the Java string are over what the UTF format can handle during a conversion.	The conversion must be completed using a string that the UTF format can handle.
STCBytesMessage is read-only, can't write byte[]	Bytes cannot be written to read-only messages.	Select a message that has write capability. Use **clearBody()** to write a byte[] to this message.
Byte array size not big enough for offset + length in STCBytesMessage	The byte array is not large enough for the API to handle the offset and length.	Increase the size of the byte array, or decrease the size of the offset and length.
STCBytesMessage is read-only, can't write Object	Objects cannot be written to read-only messages.	Select a message that has write capability. Use **clearBody()** to write an Object to this message.
Try to write object other than primitive wrappers in STCBytesMessage	The API can only write 7 or 8 primitive wrappers into the message.	You cannot do this. There is no recommended action.
No matched name in STCMapMessage	The key name cannot be mapped in the message. You cannot retrieve any value from this name.	Ensure that the key name is mapped in the message.
Type error when try to get boolean type in STCMapMessage	The message is looking for a boolean value. An integer cannot be used in place of a boolean type.	The key should match the boolean type. Ensure that a boolean value is used in the message.
Can't decode String to Byte in STCMapMessage	The string cannot be converted to a Byte value.	To fix the error, call the method **getString()**.
Can't decode String to Short in STCMapMessage	The string cannot be converted to a Short value.	To fix the error, call the method **getString()**.
Type error when try to get byte type in STCMapMessage	The value for this key cannot convert to byte type.	The value should match the byte type. Ensure that the correct value type is used for the message.

Table 33 Java Error Messages (Continued)

Java Error Message	Problem or Possible Cause	Recommended Action
Type error when try to read short type in STCMapMessage	The value for this key cannot convert to short type.	The value should match the short type. Ensure that the correct value is used for the message.
Type error when try to read char type in STCMapMessage	The value for this key cannot convert to character type.	The value should match the character type. Ensure that the correct value is used for this message.
Can't decode String to Integer in STCMapMessage	You cannot go from a String to an Integer.	Use the **getString()** method to retrieve the value from the key.
Type error when try to read int type in STCMapMessage	The value for this key cannot convert to integer type.	The value should match the integer type. Ensure that the correct value type is used for the message.
Can't decode String to Long in STCMapMessage	You cannot go from String to Long.	Use the **getString()** method to retrieve the value from the key.
Type error when try to read long type in STCMapMessage	The value for this key cannot convert to long type.	The value should match the long type. Ensure that the correct value type is used for the message.
Can't decode String to float in STCMapMessage	You cannot go from String to float.	Use the **getString()** method to retrieve the value from the key.
Type error when try to read float type in STCMapMessage	The value for this key cannot convert to float type.	The value should match the float type. Ensure that the correct value type is used for the message.
Can't decode String to double in STCMapMessage	You cannot go from String to a double.	Use the **getString()** method to retrieve the value from the key.
Type error when try to read double type in STCMapMessage	The value for this key cannot convert to double type.	The value should match the double type. Ensure that the correct value type is used for the message.
Type error when try to read String type in STCMapMessage	The value for this key cannot convert to String type.	The value should match the String type. Ensure that the correct value type is used for the message.

Table 33 Java Error Messages (Continued)

Java Error Message	Problem or Possible Cause	Recommended Action
Type error when try to read byte[] type in STCMapMessage	The value for this key cannot convert to byte[] type.	The value should match the byte[] type. Ensure that the correct value type is used for the message.
STCMapMessage is read-only; can't write name/boolean	The user attempted to add a name using a boolean string into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a name/boolean to this message.
STCMapMessage is read-only; can't write name/byte	The user attempted to add a name using a byte string into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a name/byte to this message.
STCMapMessage is read-only; can't write name/short	A short byte string cannot be written into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a name/short to this message.
STCMapMessage is read-only; can't write name/char	A character string cannot be written into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a name/char to this message.
STCMapMessage is read-only; can't write name/int	An integer string cannot be written into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a name/int to this message.
STCMapMessage is read-only; can't write name/long	A long byte string cannot be written into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a name/long to this message.
STCMapMessage is read-only; can't write name/float	A float byte string cannot be written into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a name/float to this message.

Table 33 Java Error Messages (Continued)

Java Error Message	Problem or Possible Cause	Recommended Action
STCMapMessage is read-only; can't write name/double	A double byte string cannot be written into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a name/double to this message.
STCMapMessage is read-only; can't write name/String	A String cannot be written into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a name/String to this message.
STCMapMessage is read-only; can't write name/byte[]	A byte[] string cannot be written into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a name/byte[] to this message.
STCMapMessage is read-only; can't write name/byte[] with offset and length	A byte[] with an offset and length cannot be written into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a name/byte[] with offset and length to this message.
Byte array size not big enough for offset + length in STCMapMessage	The byte array is not large enough for the API to handle the offset and length.	Increase the size of the byte array, or decrease the size of the offset and length.
STCMapMessage is read-only; can't write name/Object	An Object byte string cannot be written into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a name/Object to this message.
Try to write object other than primitive wrappers in STCMapMessage	The API can only write 7 or 8 primitive wrappers into the message.	You cannot do this. There is no recommended action.
Type error when try to get boolean type in STCMessage	The value for this key cannot convert to boolean type.	The value should match the boolean type. Ensure that the correct value type is used for the message.
Can't decode String to Byte in STCMessage	The value for this key cannot decode to Byte.	Use the **getString()** method to retrieve the value from the key.

Table 33 Java Error Messages (Continued)

Java Error Message	Problem or Possible Cause	Recommended Action
Type error when try to get byte type in STCMessage	The value for this key cannot convert to byte.	Use the **getString()** method to retrieve the value from the key.
Can't decode String to Short in STCMessage	The value for this key cannot decode to Short.	Use the **getString()** method to retrieve the value from the key.
Type error when try to read short type in STCMessage	The value for this key cannot convert to short type.	Use the **getString()** method to retrieve the value from the key.
Can't decode String to Integer in STCMessage	The value for this key cannot decode to Integer.	Use the **getString()** method to retrieve the value from the key.
Type error when try to read int type in STCMessage	The value for this key cannot convert to Integer.	Use the **getString()** method to retrieve the value from the key.
Can't decode String to Long in STCMessage	The value for this key cannot decode to Long.	Use the **getString()** method to retrieve the value from the key.
Type error when try to read long type in STCMessage	The value for this key cannot convert to long type.	Use the **getString()** method to retrieve the value from the key.
Can't decode String to float in STCMessage	The value for this key cannot decode to float.	Use the **getString()** method to retrieve the value from the key.
Type error when try to read float type in STCMessage	The value for this key cannot convert to float.	Use the **getString()** method to retrieve the value from the key.
Can't decode String to double in STCMessage	The value for this key cannot decode to double.	Use the **getString()** method to retrieve the value from the key.
Type error when try to read double type in STCMessage	The value for this key cannot convert to double type.	Use the **getString()** method to retrieve the value from the key.
Type error when try to read String type in STCMessage	The value for this key cannot convert to String type.	Use the **getByte()** method to retrieve the value from the key.
STCMessage is read-only; can't write boolean	The user attempted to add a boolean string into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a boolean to this message.

Table 33 Java Error Messages (Continued)

Java Error Message	Problem or Possible Cause	Recommended Action
STCMessage is read-only; can't write byte	The user attempted to add a byte string into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a byte to this message.
STCMessage is read-only; can't write short	The user attempted to add a short string into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a short to this message.
STCMessage is read-only; can't write int	The user attempted to add an integer string into a read-only message.	Select a message that has write capability. Use **clearBody()** to write an int to this message.
STCMessage is read-only; can't write long	The user attempted to add a long string into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a long to this message.
STCMessage is read-only; can't write float	The user attempted to add a float string into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a float to this message.
STCMessage is read-only; can't write double	The user attempted to add a double string into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a double to this message.
STCMessage is read-only; can't write String	The user attempted to add a String into a read-only message.	Select a message that has write capability. Use **clearBody()** to write a String to this message.
STCMessage is read-only; can't write Object	The user attempted to add an Object into a read-only message.	Select a message that has write capability. Use **clearBody()** to write an Object to this message.
Try to write object other than primitive wrappers in STCMapMessage	The API can only write 7 or 8 primitive wrappers into the message.	You cannot do this. There is no recommended action.

Table 33 Java Error Messages (Continued)

Java Error Message	Problem or Possible Cause	Recommended Action
Can only acknowledge a received message	A message cannot acknowledge a sent message by using the acknowledge() method.	There is no recommended action.
STCStreamMessage is write-only; can't read boolean	The user attempted to read a boolean string in a write-only message.	Select a message that has read capability. Use the **reset()** method to retrieve the value from the key.
STCStreamMessage is EOF	The client has reached the end of file for this stream; there are no additional values housed within the stream.	If you want to retrieve data again, use the **reset()** method.
Reading bytes field not complete in STCStreamMessage	The byte array must read all of the byte before it can retrieve the next value. If the byte array is not large enough to read all of the byte, it will not retrieve the next value.	Provide a larger byte array.
Type error when try to read boolean type in STCStreamMessage	The value for this key cannot convert to boolean type.	Use the **readString()** method to retrieve the value from the key.
STCStreamMessage is write-only; can't read byte	The user attempted to read a byte string in a write-only message.	Select a message that has read capability. Use the **reset()** method to read the value from the key.
Can't decode String to Byte in STCStreamMessage	The value for this key cannot decode to Byte.	Use the **readString()** method to retrieve the value from the key.
Type error when try to read byte type in STCStreamMessage	The value for this key cannot convert to byte type.	Use the **readString()** method to retrieve the value from the key.
STCStreamMessage is write-only; can't read short	The user attempted to read a short string in a write-only message.	Select a message that has read capability. Use the **reset()** method to read the value from the key.

Table 33 Java Error Messages (Continued)

Java Error Message	Problem or Possible Cause	Recommended Action
Can't decode String to Short in STCStreamMessage	The value for this key cannot decode to Short.	Use the **readString()** method to retrieve the value from the key.
Type error when try to read short type in STCStreamMessage	The value for this key cannot convert to short type.	Use the **readString()** method to retrieve the value from the key.
STCStreamMessage is write-only; can't read char	The user attempted to read a character string in a write-only message.	Select a message that has read capability. Use the **reset()** method to read the value from the key.
Type error when try to read char type in STCStreamMessage	The value for this key cannot convert to char type.	Use the **readString()** method to retrieve the value from the key.
STCStreamMessage is write-only; can't read int	The user attempted to read an integer string in a write-only message.	Select a message that has read capability. Use the **reset()** method to read the value from the key.
Can't decode String to Integer in STCStreamMessage	The value for this key cannot decode to Integer type.	Use the **readString()** method to retrieve the value from the key.
Type error when try to read int type in STCStreamMessage	The value for this key cannot convert to int type.	Use the **readString()** method to retrieve the value from the key.
STCStreamMessage is write-only; can't read long	The user attempted to read a long string in a write-only message.	Select a message that has read capability. Use the **reset()** method to read the value from the key.
Can't decode String to Long in STCStreamMessage	The value for this key cannot decode to Long.	Use the **readString()** method to retrieve the value from the key.
Type error when try to read long type in STCStreamMessage	The value for this key cannot convert to long type.	Use the **readLong()** method to retrieve the value from the key.
Can't decode String to float in STCStreamMessage	The value for this key cannot decode to float.	Use the **readString()** method to retrieve the value from the key.

Table 33 Java Error Messages (Continued)

Java Error Message	Problem or Possible Cause	Recommended Action
Type error when try to read float type in STCStreamMessage	The value for this key cannot convert to float type.	Use the **readString()** method to retrieve the value from the key.
STCStreamMessage is write-only; can't read double	The user attempted to read a double string in a write-only message.	Select a message that has read capability. Use the **reset()** method to read the value from the key.
Can't decode String to double in STCStreamMessage	The value for this key cannot decode to double.	Use the **readString()** method to retrieve the value from the key.
Type error when try to read double type in STCStreamMessage	The value for this key cannot convert to double type.	Use the **readString()** method to retrieve the value from the key.
STCStreamMessage is write-only; can't read String	The user attempted to read a String in a write-only message.	Select a message that has read capability. Use the **reset()** method to read the value from the key.
Type error when try to read String type in STCStreamMessage	The value for this key cannot convert to String type.	Use the **readBytes()** method to retrieve the value from the key.
STCStreamMessage is write-only; can't read byte[]	The user attempted to read a byte[] in a write-only message.	Select a message that has read capability. Use the **reset()** method to read the value from the key.
Type error when try to read byte[] type in STCStreamMessage	The value for this key cannot convert to byte[] type.	Use the **readString()** method to retrieve the value from the key.
STCStreamMessage is read-only; can't write boolean	The user attempted to add a name using a boolean string in a read-only message.	Select a message that has write capability. Use **clearBody()** to write a boolean to this message.
STCStreamMessage is read-only; can't write byte	The user attempted to add a name using a byte string in a read-only message.	Select a message that has write capability. Use **clearBody()** to write a byte to this message.

Table 33 Java Error Messages (Continued)

Java Error Message	Problem or Possible Cause	Recommended Action
STCStreamMessage is read-only; can't write short	The user attempted to add a name using a short string in a read-only message.	Select a message that has write capability. Use **clearBody()** to write a short to this message.
STCStreamMessage is read-only; can't write char	The user attempted to add a name using a character string in a read-only message.	Select a message that has write capability. Use **clearBody()** to write a char to this message.
STCStreamMessage is read-only; can't write int	The user attempted to add a name using an integer string in a read-only message.	Select a message that has write capability. Use **clearBody()** to write an int to this message.
STCStreamMessage is read-only; can't write long	The user attempted to add a name using a long string in a read-only message.	Select a message that has write capability. Use **clearBody()** to write a long to this message.
STCStreamMessage is read-only; can't write float	The user attempted to add a name using a float string in a read-only message.	Select a message that has write capability. Use **clearBody()** to write a float to this message.
STCStreamMessage is read-only; can't write double	The user attempted to add a name using a double string in a read-only message.	Select a message that has write capability. Use **clearBody()** to write a double to this message.
STCStreamMessage is read-only; can't write String	The user attempted to add a name using a String in a read-only message.	Select a message that has write capability. Use **clearBody()** to write a String to this message.
STCStreamMessage is read-only; can't write byte[]	The user attempted to add a name using a byte[] string in a read-only message.	Select a message that has write capability. Use **clearBody()** to write a byte[] to this message.

Table 33 Java Error Messages (Continued)

Java Error Message	Problem or Possible Cause	Recommended Action
STCStreamMessage is read-only; can't write byte[] with offset and length	The user attempted to add a name using a byte[] string with offset and length in a read-only message.	Select a message that has write capability. Use **clearBody()** to write a byte[] with an offset and length to this message.
Size of byte array not big enough in STCSteamMessage	The client provided an offset and length that are larger than the byte array.	Make the offset and length smaller than the byte array, or make the byte array larger than the offset and length.
STCStreamMessage is read-only; can't write Object	The user attempted to add a name using an Object string in a read-only message.	Select a message that has write capability. Use **clearBody()** to write an Object to this message.
Try to write object other than primitive wrappers in STCStreamMessage	The API can only write 7 or 8 primitive wrappers into the message.	You cannot do this. There is no recommended action.
STCTextMessage is read-only; can't write text	The user attempted to add a name using a text string in a read-only message.	Select a message that has write capability. Use **clearBody()** to write text to this message.
Consumer connection denied by server	The client created a consumer that the server refused to accept.	The client must create a unique name that the server can accept. Note that the server cannot create this name.
Failure to read message in consumer socket stream	There was an internal formatting error.	Contact SeeBeyond's support personnel, as this is an internal fix only.
e.toString()	This is a system error. You lost connection to the server; your system resource is lost or failed.	Restart the client or restart the server.
Incorrect message format	A message was created using an incorrect format.	Ensure that messages are created using the correct format.
Can't create producer socket	The server refused to accept this producer.	Call SeeBeyond technical support.

Table 33 Java Error Messages (Continued)

Java Error Message	Problem or Possible Cause	Recommended Action
Publish failed	The publish command initiated during a non-transactional session and the server refused to publish this message.	This message is reserved for future use.
ioe.toString()	This is a system-level error: the connection has been lost or the socket failed.	Try the following: - Restart the client or the server. - Close the session. - Close the publisher or subscriber. - Create another publisher or subscriber.
Can't create session	The server did not create a session.	Reserved for future use.
Duplicate durable subscription to destination	When the user attempted to create another durable subscription, he used the same name that was provided to the API.	No duplicate names are allowed. Ensure that the name for the durable subscription is unique.
Failed to unsubscribe	The server refused to unsubscribe. If the name does not match the name in the subscriber server list of durable subscriber names, an error is sent to the client.	Check the client to ensure that it has the correct durable subscriber name.
Commit failed	The server rejected the commit.	Try resubmitting the commit. If this fails, restart everything (the program, the consumer, the connection, and the session).
Rollback failed	The server rejected the rollback.	Try resubmitting the rollback. If this fails, restart everything (the program, the consumer, the connection, and the session).

Location of the Java method: eventSend()

Information on the Java method **eventSend()** is now located in the *e*Gate Integrator User's Guide*.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V