Framework Deployment Guide

Communication Session Failures

4/24/2025

Communication Session Failures

Contents

In a distributed interaction management solution, components must communicate continuously with each other and with some external resources. A communication session with a required resource can fail for any of these reasons:

- Failure of the resource itself
- Problem with the hardware where the resource is located
- Network connectivity problem between the two points
- Forced termination of the connection that has not shown any activity for a specified amount of time

Any time a solution component cannot communicate with a required resource, the solution may not be able to perform its required function.

After a failure is detected, the fault correction procedure normally consists of repeated attempts to regain access to either the resource in question or to a redundant resource, if one is available.

Each underlying communication protocol is typically equipped with functions that monitor open communication sessions. When a failure is detected, the communication software signals an abnormal condition to the interacting processes. This detection mechanism is fully supported in the Genesys solution, whose connection layer translates system messages into appropriate events on the application level.

However, communication protocols do not always provide adequate detection times. The TCP/IP stack, for example, may take several minutes to report a failure associated with a hardware problem (such as when a computer goes down or a cable is disconnected). This delay presents a serious challenge to the availability of any interaction management solution.

Software Exceptions

A *software exception* is an interruption in the normal flow of a program caused by an internal defect. An operating system generates exceptions in response to illegal operations that a software program attempts to perform. After generating an exception, the operating system terminates the process, which may make unavailable all solutions that use the functionality of this component.

Genesys provides an exception-handling function that monitors the exceptions that the operating system generates. The function attempts to prevent application termination by skipping the program block from which the exception originated. In most cases, this action amounts to losing one processing step with respect to a single interaction in favor of preventing an application failure.

Although the function attempts to prevent application termination, it still reports the exception with the highest priority marking. This ensures that operators know about the exception and can take appropriate measures.

You can configure the number of times during which the function tries to prevent an application from failing if it continues to generate the same exception. If this threshold is exceeded, the exception-handling function abandons the recovery procedure, allowing the operating system to terminate the application. This termination can then be detected and corrected by external fault-management functions.

By default, the exception-handling function is enabled in any daemon application; six exceptions

occurring in 10 seconds will not cause an application to terminate. To change these parameters or disable the exception handling, use a corresponding command-line parameter when starting an application.

Application Failures

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

Because the application that fails cannot perform any functions, you must use an external mechanism for both detection and correction of faults of this type. In Framework, the Management Layer is this mechanism. For information about the architecture and components in the Management Layer, see the *Framework Management Layer User's Guide*.

Configuration Server Failure Because of Memory Starvation

When Configuration Server responds to client requests with data, the responses are stored in Configuration Server memory until they are sent. The rate at which they are send depends on several factors, such as:

- load on Configuration Server
- network throughput
- ability of the client to receive and process the data

In some cases, the unsent messages might accumulate in memory. In severe cases, they could accumulate to the point where Configuration has to terminate unexpectedly because it has used 100% of memory.

To resolve this, you can impose flow control by limiting how much memory is used by unsent mail. When this limit is reached, Configuration Server stops processing client requests. When the backlog of unsent requests starts to clear and its memory usage drops below the imposed limit, Configuration Server starts process client requests again, in the order in which they were received.

Flow control is activated by two configuration options. **max-client-output-queue-size** provides flow control for communications for a single client. **max-output-queue-size** defines flow control for all clients.

> **Warning**
> Be very careful when using this option, as it effectively stops Configuration Server until all of its output buffers drop below the specified limit. Use this option only as a last resort.

Refer to the *Framework Configuration Options Reference Manual* for detailed descriptions about

these options.

Remote Site Failures

Starting in release 8.0, each Solution Control Server in a Distributed Solution Control Server environment can detect the failure of a remote site controlled by another Solution Control Server. Refer to the *Framework Management Layer User's Guide* for more information.