ApplinX Log Files

ApplinX provides a number of different tools that can help to log, trace and analyze the application's performance and functionality.

- Server Log
- Connection Pool Log (Windows only)
- Process Tracing
- Logging a Message from within a Procedure
- Path Procedure Failure Log
- GCT Trace File
- Framework Log
- Monitoring the Performance
- JavaScript Logger Engine
- Debugging/Analyzing the Web Application's Code
- Printlet Log

Server Log

The Server Log includes information as to the Server's activities and problems. The contents of the server log file are defined according to the settings configured in the Server Parameter>Log node. The Server Log can be accessed either from ApplinX Administrator, from ApplinX Designer or via an external browser: ApplinX Administrator can be used by administrators and/or developers. Administrators who do not have ApplinX Designer will access the Server Log from the Designer. Administrators who do not have ApplinX Designer or ApplinX Administrator will access the Server Log via an external browser.

There are several possible levels and each level includes the levels above it. For example, the Debug level also logs Normal, Warnings and Errors Only levels. Refer to the following links for further details regarding:

Configuring the Server Log

Viewing the Server Log

Connection Pool Log (Windows only)

A log used for fine-tuning connection pool parameters or identifying problems of the different connection pools is written within the server log. The server log should be set to no less that the **Info** log level, in order to see connection pool logs. Once the project is in the production phase, **Error** level is the recommended level to use. It is possible to set a different detail of logging for each connection pool: None, Errors, Warnings, Information and Details.

Refer to Connection Pool, General Parameters for further details.

Process Tracing

The Application Process Tracing (typically used with SOA applications) is used to provide a log of performance times of processes such as procedures, paths and programs. This can give a more specific indication (pinpoint the exact process) as to the cause of application performance problems. The application process tracing information can be saved as a txt and/or csv file. These files are located in the ApplinX>log directory.

Refer to Logs for Tracing Application Processes for details on configuring and analyzing process tracing logs.

Refer to Process Tracing Parameters in the Application Configuration Parameters for a description of all the parameters.

Logging a Message from within a Procedure

As part of the procedure work flow definitions, it is possible to define that at some point in the procedure a message will be sent to the log. This is implemented using the Log Message Node (Refer to Working with Procedure Nodes and to General Nodes (Relevant for both Flow and Path Procedures), Path Procedure Nodes and Flow Procedure Nodes.

Path Procedure Failure Log

ApplinX enables generating a log that includes debug data regarding procedures that fail in runtime. The log includes a snapshot in ASCII characters of the screen that caused the failure.

To log data regarding procedures that fail in runtime

1. In the <APPLINX HOME>/config/log/gxlog_config.xml file under the com.sabratec.util.flow.error_tracking category tag, change the <level value="off"/> line to be commented and uncomment the <!-level value="debug"/--> line.

2. Restart the ApplinX Server.

   The log will be created in the <APPLINX HOME>/log directory, and the file names will have the following format: debugging_error_in_%I_%t.log, %I being identifying information about the user and procedure name, and %t being the timestamp. The location and name of the file can be changed in the gxlog_config.xml file.

GCT Trace File

The GCT Trace File log enables recording a file, which traces the connection communication (connection pool or user) between the ApplinX server and the host, for each connection. It is possible to define whether a single trace file will be created, replacing the previously saved file or whether the data will be saved to a new file on every new connection or session. Identifying the separately saved files is possible by inserting identifying parameters in the file name (the session ID, creation time and/or connection ID). It is highly recommended to create a separate file for each session/connection or creation time. Note that trace files can be created from within the session definition overriding the application definition. This is recommended as it does not conflict with other existing sessions.

Refer to Recording Trace Files for details regarding recording trace files.

Refer to the Session View for details regarding overriding application properties per session and also for details on navigating within a session in replay mode.

Framework Log

The Framework Log lists the errors and exceptions and can then be used to debug the framework. When is production, the log should be used with discretion. Open the Framework Configuration Editor as detailed in Configuring your Web Application and configure the Log node:

File name

The log is written to this file.

Append to existing file

Selecting this check box determines whether when restarting the Web server, the log file will be overwritten or a new file will be created.

Log level

The contents of the log file are as detailed as this property defines, where every level includes the levels above it. For example, the Debug level also logs Normal, Warnings and Errors Only levels. Available values: "Normal", "Warnings", "Errors only" and "Debug" (by default Normal is selected).

Log history

Determines the number of backups saved before overwriting the old log files. For example: 10 means "save the last 10 log files, in addition to the current one, then start to overwrite".

Max. file size

Starts a new log file after the current file has been filled to the maximum file size.

Monitoring the Performance

The Performance Monitor monitors user actions and is used for analyzing framework performance. It includes log comments for every framework request. A performance log line contains the following data: Session Id, Screen Name, total request time, total server side divided into: attach, sendKeys, instant generation, detach, total page client load time. When is production, the log should be used with discretion.

Note:
Framework performance is recorded only for Instant pages and for generated pages that include PF/ENTER keys. Custom code mapped to server side buttons/links can not be monitored.

The Performance monitor is typically used in Web enabled applications. Open the Framework Configuration Editor as detailed in Configuring your Web Application and configure the Performance Monitor log:

Enable performance monitoring

Determines whether to create a performance log.

File name

The log is written to this file.

Write performance log per session ID

When selected, creates a performance log for each session.

Trace all/specific sessions

Determines whether to trace all sessions or only the session specified here.

Short/Detailed description

Short description means that the information is displayed in a tabular structure. Detailed description displays the information as a paragraph.

JavaScript Logger Engine

The JavaScript logger engine is used to log JavaScript errors (primarily) as well as to debug specific modules of the JavaScript engine. The engine logs JavaScript errors which occur to users of the Web application. These errors are logged to the framework JavaScript log, with the following user information: Session ID, IP, browser details.

Refer to JavaScript Logger Engine for further details.

Debugging/Analyzing the Web Application's Code

It is possible to use the ApplinX framework log to analyze and debug the code in the Web application. This is implemented by making changes in the config/gx_logConfig.xml file.

Refer to Investigating the Web Application's Code for further details.

Printlet Log

The Printlet Log logs information regarding the printlet activity. This information can be displayed by clicking on the Show Log button within the applet. There are two log level which can be set: normal (default) or debug. The loglevel parameter within the printlet determines the log level.