webMethods Audit Logging Guide

Version 10.5

October 2019

WEBMETHODS

This document applies to webMethods Product Suite Version 10.5 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: WEBM-LG-105-20191015

Table of Contents

About this Guide **5**
Document Conventions 5
Online Information and Support 6
Data Protection 7

Types of Audit Logs **9**
Overview 10
IS Core Audit Logging 10
API Gateway Transaction Logging 10
Document Audit Logging 10
Error Audit Logging 11
Guaranteed Delivery Audit Logging 11
Messaging Audit Logging 11
Security Audit Logging 12
Service Audit Logging 12
Session Audit Logging 13
Process Audit Log 13
Business Process Audit Logging 14
Integration Process Audit Logging 14
Task Audit Logging 14
Additional Processing for Audit Log Entries 15
Globalization 15
webMethods Broker Client Logging 16
webMethods Adapter Logging 16

Setting Up IS Core Audit Logging **17**
Configure Audit Logging 18
Start the Logger Configuration 18
Enable the Logger 18
For the Service Logger, Choose the Logging Level 18
Identify the Destination 19
File Names and Locations 19
Database Error Handling 20
Configure Logging Mode (Synchronous or Asynchronous) 20
Configure Asynchronous Mode 22
Choose the Queue Provider 22
Choose Whether to Persist the Queue 22
Specify Maximum Queue Size 22
Specify the Connection Alias for the Universal Messaging Queue 23
Specify Reader Threads for the Universal Messaging Queue 24
Queue Fail-Fast Mode for Loggers that Use Universal Messaging Queues 24

Additional Fields for the Security Logger....25
Set the Log Level for the Messaging Logger....25
Set Up Additional Service Logging....26
Set Up Customized Service Logging in Designer....26
Write User-Defined Messages or Input Pipelines to the Integration Server Server Log....28
Write Custom Values for the Current Context to the Integration Server Server Log....28
Write User-Defined Messages to the IS Core Audit Log....29
Send Messages About Service Failures to Email Addresses....30
Perform Additional Processing on Audit Log Entries....31
Controlling the Level of Exception Logging Detail....31
Controlling Date-Time Stamp and Time Zone Details....32
Rotate Audit Logs Based on Size....32
Limit the Number of Audit Log Files Kept by Integration Server....33
Use Delimiters in a File-Based Audit Log....34
Receiving Notifications When Logging Fails....35
Fail-Fast Mode for Synchronous Logging....36
Queue Size Considerations for Fail-Fast Mode....38
Other Considerations for Using Fail-Fast Mode with Audit Logging....38

Viewing Audit Log Data....41
Overview....42
View the Audit Logs in Integration Server Administrator....43
View the Error Log....43
View the Guaranteed Delivery Log....44
View the Messaging Log....44
View the Security Log....46
View the Service Log....47
View the Session Log....48
View the API Gateway Transaction Logs....48
Change the Log Displays....51
Display Logged Data in Different Languages....51
Change the Display Permanently for All Logs....51
Change the Display Temporarily for a Particular Log....52

About this Guide

This guide explains how to configure webMethods Integration Server error, session, service, security, document, and guaranteed delivery audit logging, and how to view logged data. In addition, the guide briefly describes business process, task, and integration process audit logging, and points to the webMethods documentation that provides more detailed information for those types of logging.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format *folder.subfolder.service* , APIs, Java classes, methods, properties.
Italic	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.

Convention	Description
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at "http://documentation.softwareag.com". The site requires credentials for Software AG's Product Support site Empower. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to "empower@softwareag.com" with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at "https://empower.softwareag.com/".

You can find product information on the Software AG Empower Product Support website at "https://empower.softwareag.com".

To submit feature/enhancement requests, get information about product availability, and download products, go to "Products".

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the "Knowledge Center".

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at "https://empower.softwareag.com/public_directory.asp" and give us a call.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at "http://techcommunity.softwareag.com". You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

1 Types of Audit Logs

- Overview 10
- IS Core Audit Logging 10
- Session Audit Logging 13
- Process Audit Log 13
- Additional Processing for Audit Log Entries 15
- Globalization 15
- webMethods Broker Client Logging 16
- webMethods Adapter Logging 16

Overview

Audit logging for webMethods products provides important data you need to monitor webMethods system activity and correct problems. Integration Server maintains most of the audit logging data in the webMethods product suite. This chapter describes audit logging maintained by Integration Server.

IS Core Audit Logging

Integration Server writes document, webMethods API Gateway, error, guaranteed delivery, session, service, and security audit logging data to files or database tables collectively called the *IS Core Audit Log*. "Setting Up IS Core Audit Logging" on page 17 explains how to set up logging for the IS Core Audit Log.

API Gateway Transaction Logging

You can log API Gateway transaction events produced by the Log Invocation policy. These policies are enforced in API Gateway. You can use API Gateway transaction log entries to do the following:

- Identify the SOAP session, API, and instance of API Gateway on which the transaction events occurred.
- Track whether events completed successfully or failed.
- Record the content of request and response payloads for API calls.

> **Note:** You can only log request and response payloads if you are writing API Gateway transaction events to an external RDBMS.

For information about identifying the audit log as the destination and enforcing policies with API Gateway, see *webMethods API Gateway User's Guide*.

Document Audit Logging

When a trigger is configured for exactly-once processing and Integration Server cannot determine whether the current document is a copy of one the trigger has already processed, Integration Server logs the document to the external RDBMS as an *in doubt* document.

If a transient error occurs while Integration Server is publishing, delivering, or retrieving a document for a trigger, Integration Server logs the document to the external RDBMS as a *failed* document.

If Integration Server has tried repeatedly to publish or deliver a document for a trigger from its outbound store and failed, Integration Server logs the document to the external RDBMS as a *retries exceeded* document.

For complete information, see the Working with webMethods Messaging Triggers section in the *Publish-Subscribe Developer's Guide*.

Error Audit Logging

Error audit logging provides data about exceptions thrown by services running on Integration Server. You can use error log data to debug services. Sample stack trace information is shown below.

```
java.lang.NullPointerExceptionat
JpLogger.addScheduleID(JpLogger.java:46)at
java.lang.reflect.Method.invoke(Native Method)at
com.wm.app.b2b.server.JavaService.baseInvoke(JavaService.java:287at
com.wm.app.b2b.server.ServiceManager.invoke(ServiceManager.java:344)at
com.wm.app.b2b.server.comm.DefaultServerRequestHandler.handleMessage
DefaultServerRequestHandler.java:97)at
com.wm.app.b2b.server.HTTPMessageHandler.process(HTTPMessageHandler.java:167)at
com.wm.app.b2b.server.Dispatch.run(Dispatch.java:204)at
com.wm.util.pool.PooledThread.run(PooledThread.java:105)at
java.lang.Thread.run(Thread.java:498)
```

Guaranteed Delivery Audit Logging

If you configure the guaranteed delivery capability in Integration Server, guaranteed delivery audit logging provides data about guaranteed delivery transactions. You can use guaranteed delivery log entries to do the following:

- Track when transactions start and their current status.
- See the names of guaranteed delivery processes that are running.
- Track whether the processes completed successfully or failed.

For complete information about Integration Server's guaranteed delivery capability, see *webMethods Integration Server Administrator's Guide*.

Messaging Audit Logging

Messaging audit logging provides data about messages sent by Integration Server and/or received and processed by a trigger on Integration Server. The messaging log contains detailed logging that helps with debugging issues in your messaging implementation.

You can use messaging log entries to do the following:

- Track when Integration Server prepares to send and then sends a message to the messaging provider.
- Identify when or if published messages are written to the client side queue.

- Determine whether a trigger successfully received a message from the messaging provider.
- Track failures or successful completion of pre-processing and processing for a message.
- Identify message acknowledgements sent to the messaging provider.

The messaging audit log contains log entries written for Universal Messaging connection aliases and JMS connection aliases for which enhanced logging is configured and the messaging log is identified as the destination for the log entries. For each alias, you specify that the messaging audit logger writes log entries for messages sent and/or received by the alias. You can further refine the logging by indicating that logging occurs for messages sent to particular destinations or received by specific triggers. For information about configuring enhanced logging for messaging, see the *webMethods Integration Server Administrator's Guide*.

Security Audit Logging

Security audit logging provides data about security-related administrative and operational events that occur on Integration Server. Administrative events are configuration changes related to Integration Server security activities. Examples include enabling or disabling security audit logging; changes to authorization, authentication, port, or audit settings; SSL configuration, password restrictions; or root certificates. Operational events include attempts to log on to Integration Server and to access Integration Server services and documents.

You can use security log entries to do the following:

- Track security events that occurred, when they occurred, and by whom they were performed; includes log entries about enabling or disabling security auditing in general and for particular areas (for example, authentication).
- Track whether events completed successfully or failed.

Service Audit Logging

Service audit logging provides data about flow and coded (for example, Java) services that run in Integration Server. You can use service log entries and data to do the following:

- Track when services start, their status, and their duration.
- Track whether services completed successfully or failed.
- Record the client that called the service, and the Integration Server port on which the client connected.
- Resubmit services.

In Integration Server, you globally disable all logging for all services, globally enable one type of logging for all services, or enable customized logging on a service-by-service

basis. If you enable customized logging, you set up the customized logging for specific services in Designer. For each service, you can choose the following:

- When to log based on how the service is called. For example, you might choose to log only when the service is called by a client request or trigger, as opposed to by other services.
- On what status to log. For example, you might choose to log only when the service fails.
- Whether to store the service's input pipeline and, if so, when. For example, you might choose to log the input pipeline only when an error occurs. Storing the input pipeline allows you to resubmit the service later if necessary.

Note: Whether you enable or disable service logging in Integration Server and Designer, if error logging is enabled, Integration Server always writes error log entries when service errors occur. The data includes stack trace data about the errors.

You can augment service logging data using Integration Server built-in services. The built-in services do the following:

- Enable services to post user-defined progress messages to the Integration Server server log or the IS Core Audit Log. For example, you might have a service post messages to indicate that certain pieces of code ran successfully, or to record run-time values for variables so you can see how the values changed as the service ran.
- Enable services to write the pipeline to the Integration Server server log.

Session Audit Logging

Session audit logging provides data about sessions opened on Integration Server by Designer, third-party clients, and other servers.

You can use session log entries to do the following:

- Track when sessions start, their current status, and their duration.
- Record the client that initiated the session.

Process Audit Log

Integration Server writes business process, integration process, and task audit logging data to database tables collectively called the *Process Audit Log*. For instructions on setting up logging for the Process Audit Log, see *webMethods Monitor User's Guide* (business and integration process audit logging) and *webMethods Task Engine User's Guide* (task audit logging).

Business Process Audit Logging

Business process audit logging provides data for business processes modeled in Designer that run on Integration Servers. You can use business process log entries and data to do the following:

- Identify business processes.
- Record the path that business processes took at run time.
- Track when business processes and business process steps started, changed status, and ended.
- Track whether business processes and steps completed successfully or failed.
- Resubmit business processes at specified steps.

In Designer and the Monitor user interface, you specify the amount and type of data to log for each business process model version. In Designer, you can also specify process step input and output document fields for which to log run-time values. In the Monitor user interface, you can also choose to log process transitions so you can see the path the process took at runtime. For instructions on setting up business process logging, see the *Software AG Designer Online Help* and *webMethods Monitor User's Guide*.

Process Engines log audit data for business processes. The Process Engine is a package installed on every Integration Server that runs business process steps. For detailed information on the Process Engine and how it logs data, see *Administering webMethods Process Engine*.

Integration Process Audit Logging

You can log entries that track the progress and results of integration processes. Integration processes are processes made up of a chain of services that run on Integration Servers. To log entries, you have the services that make up the integration process call webMethods Monitor built-in services that create these entries. For complete information, see *webMethods Monitor User's Guide*.

Task Audit Logging

Tasks are created in Designer and run on My webMethods Server. You can log two types of audit data for tasks:

- For all tasks, you can use task log entries to track the following:
  - When tasks are queued.
  - When users accept or release tasks, suspend and resume tasks, and complete or cancel tasks.
  - Whether tasks completed successfully, failed, or expired.

Task Engines log audit data for tasks and send the data to Integration Server. The Task Engine is a feature installed on every My webMethods Server that runs tasks. For detailed information on the Task Engine and instructions on setting up task logging, see *webMethods Task Engine User's Guide*.

Users perform the actions listed above from the task list in My webMethods. For instructions on performing actions on tasks, see *webMethods Task Engine User's Guide*.

- For tasks that are called from business processes, you can write business process log entries. Tasks called from business processes are run as business process steps, so you can log the same data for a task that you can log for any other business step (see " Business Process Audit Logging", above). Process Engines log all business process entries.

Additional Processing for Audit Log Entries

If you want to perform additional processing on log entries, you can create an event handler. For example, you could create an event handler that sends service log entries to another log, such as the Event Log on a Windows system. For information, see the *webMethods Integration Server Built-In Services Reference* and the *webMethods Service Development Help*.

Globalization

If a webMethods product is equipped with webMethods language packs and some of those language packs correspond to the language used by the operating environment in which the product is running, the product writes its log entries in the language used by the operating system. If the product is equipped with no language packs or with language packs that do *not* correspond to the language used by the operating system, the product writes its log entries in U.S. English.

Suppose your operating environment uses Japanese as its language. You have installed language packs including the Japanese Language Packs on Integration Server, so Integration Server stores its own log entries in Japanese. You have not installed the Japanese Language packs on Trading Networks, so Integration Server stores Trading Networks log entries in U.S. English.

Note: Even if no language packs are installed on the webMethods product and the product is using U.S. English, Integration Server might store log entries from external sources, such as database drivers or adapter resources, in the language used by the operating environment in which the product is running.

webMethods Broker Client Logging

For instructions on configuring your system to log documents that webMethods Broker clients publish to or subscribe to on Brokers, see *Administering webMethods Broker*.

Note: webMethods Broker is deprecated.

webMethods Adapter Logging

For instructions on setting up logging for webMethods adapters, see the adapter guides.

2 Setting Up IS Core Audit Logging

- Configure Audit Logging 18
- Set the Log Level for the Messaging Logger 25
- Set Up Additional Service Logging 26
- Perform Additional Processing on Audit Log Entries 31
- Controlling the Level of Exception Logging Detail 31
- Controlling Date-Time Stamp and Time Zone Details 32
- Rotate Audit Logs Based on Size 32
- Limit the Number of Audit Log Files Kept by Integration Server 33
- Use Delimiters in a File-Based Audit Log 34
- Receiving Notifications When Logging Fails 35
- Fail-Fast Mode for Synchronous Logging 36

Configure Audit Logging

Integration Server writes to the IS Core Audit Log using *audit loggers*. Each type of logging data has its own logger. For example, the error logger writes the audit log entries for errors, the service logger writes audit log entries for services, and the document logger writes documents. Each logger has a default configuration, but you can reconfigure it. You do not have to disable a logger to reconfigure it; you can reconfigure an enabled logger.

Start the Logger Configuration

1. In Integration Server Administrator, go to the **Settings > Logging** page.

   Note: If your Integration Server license does not include security auditing, guaranteed delivery, or API Gateway, those loggers are unavailable.

2. In the **Logger List**, click a logger you want to set up.
3. Click **Edit *type* logger** and set the fields described below. When you are done, click **Save Changes** and then restart Integration Server.

Enable the Logger

In the **Enabled** field, indicate whether you want the logger to start writing log entries.

For the Service Logger, Choose the Logging Level

If you are configuring the service logger, in the **Level** field, choose the level of logging for services.

Value	Description
perSvc	Lets you set up customized logging on a service-by-service basis in Designer.
Brief	The logger writes start and failure or start and success log entries for every service every time the service is called, either directly (top-level) or by another service (nested).
Verbose	Same as **Brief**, except that the logger also writes the input pipeline in all cases.

The **brief** and **verbose** values are globally applied to services; if you choose one of those values, you cannot override it in Designer for individual services. Software AG

recommends using these values only in a development environment, when performing an extensive debugging effort.

Identify the Destination

In the **Destination** area, identify where the logger is to write entries. Any external RDBMSs will have been set up after installation (see *Installing Software AG Products*).

Note: The messaging audit logger always writes to a file and cannot be configured to write to a database.

In the **Maximum Retries** field, specify the maximum times the logger should retry writing the entry to the destination if the first attempt fails because of a transient error. A transient error is an error that arises from a temporary condition that might be resolved or corrected quickly, such as the unavailability of a resource due to network issues or failure to connect to a database.

Also set the **Wait Between Retries** field.

Note: If the logger is logging data asynchronously (see below) and the logging queue has many audit records in it, the elapsed time between logging attempts may be longer than the value in the **Wait Between Retries** field.

Note: The **Maximum Retries** field and the **Wait Between Retries** field do not apply to the messaging audit logger because the messaging audit logger cannot be configured to write to a database.

File Names and Locations

If you set the destination to **File**, the logger will write entries to a file in the *Integration Server_directory*\instances*instance_name* \logs directory, as described below, unless otherwise noted.

Logger	Log File Name
API Gateway transaction	AGW_EVENT_TXN_*yyyymmdd_hhmmss* .log **Note:** The logger writes the file to the *Integration Server_directory*\instances*instance_name* \logs \APIGateway directory.
Error	WMERROR_*yyyymmdd_hhmmss* .log
Guaranteed delivery	WMTXIN_*yyyymmdd_hhmmss* .log (inbound transactions log file, on the host machine of the Integration Server that is handling guaranteed delivery requests)

Logger	Log File Name
	WMTXOUT_*yyyymmdd_hhmmss* .log (outbound transactions file, on the host machine of an Integration Server that is submitting guaranteed delivery requests)
Messaging	WMMESSAGING_*yyyymmdd_hhmmss* .log
Security	WMSECURITY_*yyyymmdd_hhmmss* .log
Service	WMSERVICE_*yyyymmdd_hhmmss* .log
Session	WMSESSION_*yyyymmdd_hhmmss* .log

Database Error Handling

There are times when Integration Server will automatically reset the destination from **Database** to **File** (for example, when Integration Serverstarts up and the Audit Log function cannot connect to the external RDBMS or when the Audit Log function connects to the external RDBMS but it becomes unavailable for subsequent sessions). In such cases, Integration Server will change the destination for all loggers that are capable of writing to a file (such as the Error, Session, Service, Security, Guaranteed Delivery, API Gateway transaction loggers, and so on) to **File**. The loggers that can only write to RDBMS (such as the Document and Process Engine loggers) will become unavailable. When Integration Server restarts and the connection to the database is restored, Integration Server will set the destination for the loggers back to **Database**.

During audit logging, database errors are returned from the database drivers as SQLExceptions that contain a numeric code that represents an error or warning. If the numeric code is listed in the *Integration Server_directory*\instances*instance_name* \config \auditing\transient.sql.errors.xml file, Integration Server considers the error to be transient. If the numeric code is *not* listed in that file, Integration Server considers the error to be non-transient. If you discover transient errors that are not listed in the transient.sql.errors.xml file, modify the file to include numeric codes for those errors.

Configure Logging Mode (Synchronous or Asynchronous)

In the **Mode** field, choose whether the logger is to write entries to the destination synchronously or asynchronously. In synchronous mode, the logger writes entries directly to the destination. In asynchronous mode, the logger writes entries to a queue, then later writes the entries from the queue to the destination. Each logger has its own queue.

You might use synchronous logging when you have an application that requires some type of auditing to succeed, and you do not want to proceed without knowing that the auditing occurred. Synchronous mode is faster for a logger writing to a database under

load. In contrast, synchronous mode might be slower for a logger writing to a file under load.

When the logger tries to log in synchronous mode, one of the following occurs:

- The logger writes the entry to the destination.
- The logger cannot write the entry to the destination because of an error.

Error Type	Action
Non-transient	Writes the entry to the FailedAudit_*yyyymmdd_hhmmss* .log in the *Integration Server_directory*\instances*instance_name* \logs directory.
Transient	Switches to asynchronous mode for the entry.

When the logger tries to write an entry in asynchronous mode, one of the following occurs:

- The logger writes the entry to the destination.
- The logger cannot write the entry to the destination because of an error.

Error Type	Action
Non-transient	Writes the entry to the FailedAudit_*yyyymmdd_hhmmss* .log.
Transient	Retries writing the entry to the destination at the interval specified on the **Wait Between Retries** field. One of the following occurs: ■ The logger writes the entry to the destination on one of the retry attempts. ■ Transient errors occur on the retry attempts, and the **Maximum Retries** value is exceeded. The logger writes the entry to the FailedAudit_*yyyymmdd_hhmmss* .log.

Note: The Service logger cannot write the input pipeline to this file, and the API Gateway logger cannot write request and response payloads to the FailedAudit_*yyyymmdd_hhmmss* .log file.

Configure Asynchronous Mode

Choose the Queue Provider

In the **Queue Provider** field, indicate whether the logger should write to an internal queue, sometimes called a light-weight queue, or to a Universal Messaging queue. Universal Messaging queues might provide better performance.

Note: The messaging audit logger uses the internal, light-weight queue as the audit logging queue. The messaging logger cannot be configured to use a queue on Universal Messaging as the audit logging queue.

Universal Messaging queue names adhere to one of the following conventions depending on whether the **Client Prefix Is Shared** check box is selected for the Universal Messaging connection alias used by the logger:

- If the **Client Prefix Is Shared** check box is selected for the Universal Messaging connection alias, the Universal Messaging queue name follows the naming convention:

  wm/is/audit/clientPrefix/*logger_name* Queue (for example, wm/is/audit/myClientPrefix/SessionQueue)

- If the **Client Prefix Is Shared** check box is *not* selected for the Universal Messaging connection alias, the Universal Messaging queue name follows the naming convention:

  wm/is/audit/*logger_name* Queue (for example, wm/is/audit//SessionQueue)

Note: When the Universal Messaging connection alias is configured to share the client prefix, multiple Integration Servers in a stateful or stateless cluster can use the alias to use the same Universal Messaging queue as the logging queue. Additionally, this logging queue can be shared across a Universal Messaging realm. That is, loggers on multiple Integration Servers can write log entries to and read log entries from one Universal Messaging queue shared across a Universal Messaging realm.

Choose Whether to Persist the Queue

In the **Guaranteed** field, indicate whether to persist the queue on disk or maintain it in memory. Persisting the queue on disk is safer but can adversely affect logging performance. Maintaining the queue in memory provides better logging performance, but if Integration Server shuts down abnormally, the log entries in the queue will be lost.

Specify Maximum Queue Size

In the **Maximum Queue Size** field, specify the maximum number of entries the queue can hold. Specify numerals only (for example, do not include commas or periods).

Choose a value that accommodates your system's average volume for log entries. If your logging volume has sudden spikes, the queue can usually catch up by writing the

pending entries during lulls. Make sure the Integration Server host machine has enough disk space or memory to accommodate the largest possible size of the queue as specified in this field, as well as the requirements of other applications the Integration Server is hosting. The queue's insertion of logged data into a database is constrained by the database's availability and connections limit.

If the queue reaches its maximum, the logger writes the log entries to a file called FailedAudit_*yyyymmdd_hhmmss* .log in the *Integration Server_directory*\instances *instance_name* \logs directory. You can scan the file to find events that were not logged.

Note: The Service logger cannot write the input pipeline to this file, and the API Gateway logger cannot write request and response payloads to this file.

If a logger uses a JDBC functional alias for which fail-fast mode is configured, the queue for the loggers must be large enough to hold all the audit logging records it may receive while the audit logging database is unavailable. For more information about fail-fast mode for audit logging, see "Queue Size Considerations for Fail-Fast Mode" on page 38.

Specify the Connection Alias for the Universal Messaging Queue

If the logger will use a Universal Messaging queue, select the Universal Messaging connection alias for the logger to use. If the alias does not exist yet, click **Connection Alias** to create the alias. For information about creating a Universal Messaging connection alias, see the Working with Messaging Connection Aliases section in *webMethods Integration Server Administrator's Guide*.

The connection alias is a webMethods messaging connection alias that contains configuration information for establishing a connection to a Universal Messaging server. The logger can use a webMethods connection alias that is also used with webMethods messaging triggers, or you can create a Universal Messaging connection alias specifically for use with audit logging.

Note: If you want multiple Integration Servers to use the same Universal Messaging queue as the logging queue across a Universal Messaging realm, the Universal Messaging connection alias used for logging must have the **Client Prefix Is Shared** check box selected. This ensures that Universal Messaging includes the client prefix in the namespace portion of the Universal Messaging queue name. Each of the Integration Servers in the cluster must use the same Universal Messaging connection alias.

Integration Server creates a session pool for each Universal Messaging connection alias that is used by a logger. The session pool contains the Universal Messaging sessions used by the logger to write an entry to the Universal Messaging queue. For example, if the error logger uses aliasOne and the service and session loggers use aliasTwo, Integration Server creates a session pool for aliasOne that is used only by the error logger, and a session pool for aliasTwo that is shared by the service and session loggers. You specify the minimum and maximum size of the session pools used to contain Universal Messaging sessions on the watt.server.audit.um.sessionPool.min and

watt.server.audit.um.sessionPool.max server configuration parameters. For instructions on setting the parameters, see *webMethods Integration Server Administrator's Guide*.

Note: The **Connection Alias** field does not apply to the messaging audit logger because the messaging audit logger cannot be configured to use a Universal Messaging queue as the audit logging queue.

Specify Reader Threads for the Universal Messaging Queue

If the logger will use a Universal Messaging queue, in the **Number of Readers** field, specify the number of reader threads the logger can use to retrieve entries from the queue and write them to the destination. Each reader thread creates a session with the Universal Messaging server at Integration Server start up.

If you have a standalone Integration Server, set the logger to a number greater than 0. If you have a clustered or non-clustered group of Integration Servers, you might be able to improve performance by designating only one or a few loggers of each type across the cluster or group as readers. In this case, set non-reader loggers to 0, and reader loggers to a number greater than 0.

If the number of reader threads you specify is insufficient, the queue will fill up and the logger will write subsequent entries to the FailedAudit_*yyymmdd_hhmmss* .log file.

Note: The **Number of Readers** field does not apply to the messaging audit logger because the messaging audit logger cannot be configured to use a Universal Messaging queue as the audit logging queue. The messaging audit logger always uses the internal queue as the audit logging queue.

Queue Fail-Fast Mode for Loggers that Use Universal Messaging Queues

Integration Server provides a queue fail-fast mode for loggers that use Universal Messaging queues. In queue fail-fast mode, loggers writes all entries to their internal queues rather than to their Universal Messaging queues. Loggers configured as readers pause their reader threads.

Integration Server enters queue fail-fast mode when Integration Server cannot connect to the Universal Messaging server at startup, or when the connection to the Universal Messaging server is lost after Integration Server has started running. Integration Server tries to reconnect to the Universal Messaging server. For a connection failure at startup, Integration Server tries to reconnect at the interval specified on watt.server.audit.um.sessionPool.retryInterval server configuration parameter. For instructions on setting the parameter, see *webMethods Integration Server Administrator's Guide*.

After the connection is established or re-established, Integration Server exits fail-fast mode and the loggers resume logging. Reader loggers drain their internal queue by writing entries from the queues to the destination.

Additional Fields for the Security Logger

If you are configuring the Security logger, set the additional fields below.

By default, the security logger writes security events that occur after Integration Server has completed its startup sequence. In the **Generate Audit Data** on **Startup** area, choose whether the logger should also write security events that occur during Integration Server's startup sequence.

Note: Writing security events during startup makes the startup sequence significantly slower.

In the **Generate Auditing Data on** area, choose when Integration Server should log security events as described in the following table.

Value	Description
Success	Only when the event completes successfully.
Failure	Only when the event fails.
Success or Failure	Regardless of outcome.

In the **Security Areas to Audit** area, select the areas for which to log security-related events.

Set the Log Level for the Messaging Logger

If you are configuring the messaging logger, you may set the logging level for the messaging logger to a level other than the default level of Info. Each logging level includes the indicated type of messages plus all messages from the levels above it (for example, the Trace level includes Info and Debug messages). The debugging level determines how much detail the message audit logger records in the messaging audit log.

Specify one of the following as the log level for the messaging logger:

Logging Level	Description
Info	Main sending and receiving actions such as success or failure of sent or received messages.
Debug	Details of sending and receiving activities such as preprocessing and message acknowledgment.

Logging Level	Description
Trace	Further details of sending and receiving activities, such as the start and end of polling the messaging provider for additional messages. **Note:** Trace-level messages are provided for JMS messaging only.

Note: The logging level for the messaging audit logger can be different than the logging level for the 0168 Messaging (Enhanced Logging) server log facility for the server log.

Set Up Additional Service Logging

If you selected **perSvc** logging for the Services logger, you must set up customized logging in Designer for every service you want to audit.

You can augment any type of service logging by using Integration Server built-in services to write user-defined messages to the Integration Server server log or the IS Core Audit Log.

Set Up Customized Service Logging in Designer

For each service, you can choose the following logging options. For complete information on working with services, see the *Software AG Designer Online Help*.

- Whether to log and, if so, when, as follows:
  - Every time the service is called, whether by a client request, trigger, or another service.
  - Only when the service is called by a client request or a trigger (that is, when the service is a top-level service).
- The statuses in the service's execution on which to log - when the service fails, fails or succeeds, or starts and fails or succeeds.
- Whether to store the service's input pipeline and, if so, always or only when an error occurs. Storing the input pipeline allows you to resubmit the service later if necessary.

  Note: You can only log input pipelines if you are writing service data to an external RDBMS.

- Whether to log select fields from the service signature.

Note: If any of the selected fields you log from the service signature require a greater length than the default of 512 characters, you can modify the length of the STRINGVALUE column in the WMSERVICECUSTOMFLDS table to accommodate a larger value. Keep the following points in mind when increasing the column length:

- If the data written to the STRINGVALUE column contains multibyte characters, data can be truncated in the middle of a character. To avoid this, Integration Server truncates the last character boundary before the maximum length of the field, which could result in the data contained in the column being slightly smaller than the maximum value set in the audit logging database. To ensure that characters are not truncated, use the watt.server.audit.dbEncoding server configuration parameter to specify the character set used by the audit logging database. For more information about watt.server.audit.dbEncoding, see *webMethods Integration Server Administrator's Guide*.
- Integration Server checks the database for column width by obtaining the metadata and examining the CHAR_OCTET_LENGTH field of the column. If the database vendor does not supply a CHAR_OCTET_LENGTH value for the column, Integration Server uses the default length of 512 characters for the STRINGVALUE column.

You must restart Integration Server for the new length to take effect.

- Whether to associate a custom value with an auditing context. The custom value can be used to search for service audit records in the Integration Server.

To improve service logging performance, do the following:

- Set up customized logging for top-level services only. Avoid logging nested services.
- Log on service failure or log on service failure or success. Only choose to log on service failure or success *and* start when you need the greatest possible quality of service.
- Logging the pipeline can negatively affect performance, especially if the pipeline contains large objects, because Integration Server has to make a copy of the pipeline every time the service is invoked. Store the input pipeline only for top-level services, and only when absolutely necessary (for example, on failure only). Remove all unnecessary data from the pipeline to minimize the volume of data to store.
- The audit log entries that the Process Engine can write for business process steps that run services convey the same information as the audit log entries you can write for services. In addition, the Process Engine can store the input pipeline for services that are run by process steps. To improve logging performance, avoid logging the same information twice by coordinating audit logging for services that are invoked by process steps.

Note: When coordinating logging, keep in mind that when a service is run by a process step, it is actually called by a wrapper service, making it a nested service rather than a top-level service.

Write User-Defined Messages or Input Pipelines to the Integration Server Server Log

You can have running services post user-defined progress messages to the Integration Server server log. For example, you might have a service post messages to indicate that certain pieces of code ran successfully, or to record run-time values for variables so you can see how the values changed as the service ran. To do so, you make the service call the Integration Server built-in service pub.flow:debugLog.

You can also have running services write input pipelines to the Integration Server server log. To do so, you make the service call the Integration Server built-in service pub.flow:tracePipeline.

You can write this information regardless of how you have configured service audit logging. For instructions on using these services, see the *webMethods Integration Server Built-In Services Reference* and the *webMethods Service Development Help*. For information on the Integration Server server log, see *webMethods Integration Server Administrator's Guide*.

Write Custom Values for the Current Context to the Integration Server Server Log

You can write custom values associated with auditing contexts to the server log. If Integration Server is configured to write service audit data to a database, you have the option of using these custom values as search criteria to locate and view specific logged service data. You search logged audit data using the webMethods Monitor.

To write custom values for the current context to the server log, use the Integration Server built-in service pub.flow:setCustomContextID.

The pub.flow:setCustomContextID service is stored in the WmPublic package. Its input parameter is described below. For instructions about using this service, see the *webMethods Integration Server Built-In Services Reference* .

Input Parameters

id **String** Optional. The custom value for the current auditing context. Specify a value that you want to associate with the auditing context.

Output Parameters

None.

Write User-Defined Messages to the IS Core Audit Log

If you are storing service audit data in an external RDBMS, and you have installed the Process Engine, you can have services post user-defined progress messages to the IS Core Audit Log. For example, you might have a service post messages to indicate that certain pieces of code ran successfully, or to record run-time values for variables so you can see how the values changed as the service ran. To do so, you make the service call the Integration Server built-in service pub.prt.log:logActivityMessages.

Note: You view these messages in Monitor.

The pub.prt.log:logActivityMessages service is stored in the WmPRT package. Its input and output parameters are described below.

Input Parameters

FullMessage String Optional. Complete message to record in the IS Core Audit Log. The message can be up to 1024 bytes.

Note: If messages recorded in the IS Core Audit Log require more than 1024 characters, you can modify the length of the FULLMESSAGE column in the WMSERVICEACTIVITYLOG table to accommodate a larger value. Keep the following points in mind when increasing the column length:

- If the data written to the FULLMESSAGE column contains multibyte characters, data can be truncated in the middle of a character. To avoid this, Integration Server truncates the last character boundary before the maximum length of the field, which could result in the data contained in the column being slightly smaller than the maximum value set in the audit logging database. To ensure that characters are not truncated, use the watt.server.audit.dbEncoding server configuration parameter to specify the character set used by the audit logging database. For more information about watt.server.audit.dbEncoding, see *webMethods Integration Server Administrator's Guide*.
- Integration Server checks the database for column width by obtaining the metadata and examining the CHAR_OCTET_LENGTH field of the column. If the database vendor does not supply a CHAR_OCTET_LENGTH value for the column, Integration Server uses the default length of 1024 characters for the FULLMESSAGE column.

You must restart Integration Server for the new length to take effect.

BriefMessage String Optional. Shortened version of the full message. The message can be up to 240 bytes.

EntryType String Type of message.

Message Type	Description
`Message`	Informational. No action is needed.
`Warning`	A warning message. The service can complete successfully even if the circumstance causing the warning is not addressed.
`Error`	An error message. An error message will not stop the service or put it in an error state. However, the service cannot complete successfully until the circumstance causing the error is resolved.

Output Parameters

None.

Send Messages About Service Failures to Email Addresses

You can configure Integration Server to automatically send notifications to a specified e-mail address each time a service fails. These service failures are the stack track data written to the error log. In a development environment, you might direct these messages to the developer. In a production environment, you might direct them to the Integration Server administrator.

To send messages about service failures to e-mail addresses

1. In Integration Server Administrator, go to the **Settings > Resources** page and click **Edit Resource Settings**.
2. In the **SMTP Server** field, type the server name or IP address of the SMTP server to use to send the messages.
3. In the **Internal Email** field, type the e-mail address to which to send messages about critical log entries. Typically, you would specify the email address for the Integration Server administrator.
4. In the **Service Email** field, type the e-mail address to which to send messages about service failures. In a development environment, you might direct these messages to

the developer. In a production environment, you might direct these messages to the Integration Server administrator.

5. By default, Integration Server uses character set UTF-8 for the messages. If you want to use a different character set, identify the character set in the **Default Email Charset** field.
6. Click **Save Changes**.
7. By default, Integration Server connects to port 25 on the specified SMTP server. Also by default, when sending a message, Integration Server provides its own address (the From Address) as Integration-Server@*localhost* , where *localhost* is the Integration Server host machine. If you want to change either of these properties, follow these steps:
   a. In Integration Server Administrator, go to the **Settings > Extended** page. Integration Server Administrator displays a list of Integration Server configuration properties you can change using this method.
   b. Click **Edit Extended Settings**. In the **Extended Settings** box, set the properties as follows:

To change this property	Set this parameter
SMTP server port	`watt.server.smtpServerPort=`*`port to use`*
Integration Server's **From Address**	`watt.server.email.from=`*`new From Address to use`*

   c. Click **Save Changes**, then restart Integration Server.

Perform Additional Processing on Audit Log Entries

If you want to perform additional processing on log entries, you can create an event handler. For example, you could create an event handler that sends service log entries to another log, such as the Event Log on a Windows system. For information, see the *webMethods Integration Server Built-In Services Reference* and the *webMethods Service Development Help*.

Controlling the Level of Exception Logging Detail

You can control how the Integration Server logs service exceptions, and to what level of detail, by setting the watt.server.deprecatedExceptionLogging parameter.

If this parameter is set to `false` (detailed exception logging), the **Stack Trace** column of the error log shows the innermost stack trace (that is, the stack trace that points to the source of the problem).

If this parameter is set to `true` (basic exception logging), the stack trace is often truncated and the cause of the exception becomes more difficult to trace. For this reason, Software AG recommends that you do not set this parameter to true unless you are executing services that catch exceptions and do not re-issue them.

For more information about this parameter, see *webMethods Integration Server Administrator's Guide*.

Controlling Date-Time Stamp and Time Zone Details

In the audit log files, you can control the format entries, such as time stamp and time zone.

- To change the Date-Time Stamp format, set `watt.server.logs.dateStampFmt =format of time stamp`. The format of the date-time stamp must be compatible with the java.text.SimpleDateFormat class. .
- To change the format of the Time Zone, set `watt.server.logs.dateStampTimeZone=time zone`. The format of the time zone must be compatible with thejava.util.TimeZone class.

> **Note:** If this property is not set, Integration Server uses the time zone of the hosting Integration Server.

For more information about these parameters, see *webMethods Integration Server Administrator's Guide*.

Rotate Audit Logs Based on Size

By default, Integration Server rotates file-based audit logs daily. While daily rotation might be sufficient for audit logs that do not regularly contain a lot of data or are not frequently written to, some audit logs might increase rapidly in size. For example, if the service logger has a logging level of verbose, the service logger generates an extensive amount of data that is then written to the service log, specifically WMSERVICE_*yyyymmdd_hhmmss* .log In addition to consuming resources, a large audit log file can be difficult to examine. To avoid this, you can configure Integration Server to rotate the audit log files by size in addition to rotating by day.

Integration Server provides a server configuration parameter that you can use to specify the size limit for the file-based audit logs. When watt.server.audit.logRotateSize is set to a valid value, Integration Server rotates an audit log file when it reaches that size or at midnight, whichever occurs first. For example, suppose that the error logger writes to a file destination. If watt.server.audit.logRotateSize is set to 100KB and at midnight the error log file size is 80KB, Integration Server still rotates the error log at midnight. When Integration Server rotates an audit log file, Integration Server starts a new audit log file for the logger, using the time stamp of the first log entry as the date and time portion of the log file name.

There is no default value for the watt.server.audit.logRotateSize parameter. If no value is specified for watt.server.audit.logRotateSize, Integration Server rotates the audit log files for file-based loggers daily only. If an invalid value is specified, Integration Server proceeds as if no value was specified for the parameter. If you set the value using the **Extended Settings** page in Integration Server Administrator, validation prevents an invalid value from being saved.

The watt.server.audit.logRotateSize parameter affects only the audit loggers configured write to a file. The parameter does not affect audit loggers configured to write to a database.

Note: Integration Server always rotates the audit log for a logger when the current audit log reaches the maximum size as set by the watt.server.audit.logRotateSize parameter. Integration Server also rotates the audit log for a logger after server start up and daily, after midnight. Integration Server rotates the audit logs after the logger writes data to the log. Some audit loggers write data to the log immediately after startup or midnight. Other audit loggers, such as the error logger, do not necessarily write audit log data at start up or at midnight. Instead, Integration Server rotates the log when the logger writes data to the log file.

For more information about the watt.server.audit.logRotateSize parameter, see *webMethods Integration Server Administrator's Guide*.

Limit the Number of Audit Log Files Kept by Integration Server

By default, Integration Server keeps audit log files for file-based audit loggers indefinitely. Because audit log files can rapidly become large or numerous when using more verbose logging levels or under heavy loads, you might want to limit the number audit log files that Integration Server keeps for each logger on the file system. Limiting the number of files is sometimes referred to as pruning.

Integration Server provides the watt.server.audit.logFilesToKeep server configuration parameter for limiting the number of audit log files that Integration Server maintains on the file system for each logger, including the current log file for the logger. When the number of log files for an audit logger reaches the established limit, Integration Server deletes the oldest audit log file each time Integration Server rotates the audit log for the logger at midnight or due to size limitations. Integration Server also prunes the number of audit log files for an audit logger when creating a new log file for the audit logger. For example, after Integration Server start up, the audit logger writes log data to a new file. If, after creating the new audit log file, Integration Server determines the number of log files for the logger exceeds the maximum allowed, Integration Server deletes the oldest audit log file.

Following are some examples of values for watt.server.audit.logfilesToKeep and the resulting Integration Serverbehavior:

- If you set watt.server.audit.logFilesToKeep to *n* , Integration Server keeps the current audit log file for a logger and up to *n-1* archived audit log files. For example, if you set watt.server.audit.logFilesToKeep to 30 and the session logger is configured to write to a file, Integration Server keeps the current session log file and up to 29 archived session log files.
- If you set watt.server.audit.logFilesToKeep to 1, Integration Server keeps the current audit log file for a logger and no previous audit log files. When Integration Server rotates the audit log for a logger that writes to a file, Integration Server does not keep the previous audit log for the logger
- If you set watt.server.serverlogFilesToKeep to 0, or any value less than 1, Integration Server keeps an unlimited number of audit log files for each audit logger that writes to a file

The default value of watt.server.audit.logFilesToKeep is 0, indicating that there is no limit to the number of audit log files that Integration Server maintains for an audit logger that write to file.

The watt.server.audit.logFilesToKeep parameter affects only the audit loggers configured to write to a file with the exception of the FailedAudit logs. The parameter does not affect audit loggers configured to write to a database.

For more information about the watt.server.audit.logFilesToKeep parameter, see *webMethods Integration Server Administrator's Guide*.

Use Delimiters in a File-Based Audit Log

The fields for an entry in a file-based audit log can be fixed length or character delimited. By default, Integration Server uses fixed-length fields where the size of the fields determines the length of the fields. For each log entry, Integration Server writes the maximum number of characters for each field. If the actual data for the field is less than the defined size, then Integration Server pads the field with whitespace. This helps to maintain constant field widths which results in a log file with a human-readable format. However, when a log file has a large volume of entries, the fixed-length fields can quickly consume disk resources. To address this resource inefficiency, you can change the audit loggers to write character-delimited log entries instead of fixed length.

Integration Server includes the following server configuration parameters that you can use to specify the field and record delimiters for entries in a file-based audit log:

- watt.server.audit.file.fieldDelimiter. Specifies the character sequence to delimit fields within a log record in a file-based audit log.
- watt.server.audit.file.recordDelimiter. Specifies the character sequence to delimit records in a file-based audit log.

Keep the following information in mind when setting character delimiters for fields and records for file-based audit logs.

- The watt.server.audit.file.fieldDelimiter and watt.server.audit.file.recordDelimiter parameters affect all audit logs written to a file including the FailedAudit log. The parameters do not affect audit logs written to a database.
- There is no default value for the watt.server.audit.file.fieldDelimiter and watt.server.audit.file.recordDelimiter parameters. If no value is specified for the parameters, Integration Server writes log entries using the fixed-length format where each field is the maximum length.
- A valid delimiter is any character sequence where a character is any visible (printing) character (0021-007E) from the US ASCII character set.
- The values for watt.server.audit.file.fieldDelimiter and watt.server.audit.file.recordDelimiter must be different.
- Integration Server writes character-delimited audit log entries only if both watt.server.audit.file.fieldDelimiterandwatt.server.audit.file.recordDelimiter are set to valid values and the values are different. If the values are not valid and are not different, Integration Server uses the fixed-length format for the file based audit logs.
- When specifying delimiter character sequences, do not use a character sequence likely to appear in a log entry.
- Whether a file-based audit log uses a fixed-length format or uses a character-delimited format does not affect how Integration Server Administrator displays the log file.
- After switching between fixed-length format and character-delimited format, Integration Server Administrator cannot display entries from the previous format.
- You must restart Integration Server for changes to the watt.server.audit.file.fieldDelimiter and watt.server.audit.file.recordDelimiter to take effect.
- To revert to fixed-length field entries for file-based audit logs, delete the values of the watt.server.audit.file.fieldDelimiter and watt.server.audit.file.recordDelimiter parameters and restart Integration Server.

Receiving Notifications When Logging Fails

You can subscribe to audit error events to notify administrators when Integration Server cannot write audit logging information to the IS Core Audit Log. *Audit error events* occur in the following instances:

- When a SQLException is encountered while trying to insert an audit record into the audit logging database.
- When Integration Server initializes and cannot connect to the audit logging database.
- When the Service logger is configured to retry failed auditing attempts, the audit error event is fired for the initial failure and each subsequent failure.

When you subscribe to an audit error event, you can supply a filter to limit the events that your event handler receives. The filter applies to the concatenated values of the *destination* and *errorCode* fields. The following table shows how you can use filters to limit the events that your event handler will receive:

This filter	Limits the events that the event handler receives to
YourSearchTerm	Events that contain *onlyYourSearchTerm* .
**YourSearchTerm*	Events that contain *YourSearchTerm* at the end of the audit error event value.
*YourSearchTerm**	Events that contain *YourSearchTerm* at the beginning of the audit error event value.
YourSearchTerm	Events that contain *YourSearchTerm* anywhere within the audit error event value.

You subscribe to audit error events using pub.event:addSubscriber and then define the specifications for the audit error event handlers with the pub.event:auditError service. For more information about these services, see *webMethods Integration Server Built-In Services Reference* .

You can indicate whether event handlers for audit error events are invoked synchronously or asynchronously by using the watt.server.event.audit.async server configuration parameter. For more information, see *webMethods Integration Server Administrator's Guide*.

Fail-Fast Mode for Synchronous Logging

A logger is configured to use fail-fast mode when all of the following are true:

- The logger is synchronous.
- The logger writes data to a database.
- The logger uses a JDBC functional alias associated with a JDBC pool alias.
- Fail-fast is enabled for the JDBC functional alias.
- The logger is specified in the watt.server.audit.failFastLoggers server configuration property or the server configuration parameter is empty. When watt.server.audit.failFastLoggers is empty and fail-fast is enabled in the ISCoreAudit functional alias, Integration Server, all synchronous loggers with a database destination will have fail-fast capability.

Fail-fast mode is a capability of a JDBC functional alias that allows requests for a database connection to fail immediately if a previous request failed because of a

transient error. By causing requests that will not succeed to fail quickly, Integration Server eliminates the time a request spends making retry attempts.

When fail-fast mode is enabled and the JDBC pool alias used by the JDBC functional alias cannot establish a connection to the database, the JDBC functional alias enters fail-fast mode. In fail-fast mode:

- All attempts by an Integration Server component that uses the JDBC functional alias to get a database connection will return immediately with a SQLException.
- The JDBC function monitors database connectivity. When database connectivity is restored, the JDBC functional alias exits fail-fast mode. Integration Server components that use the JDBC functional alias to obtain a database connection will obtain a connection when one is requested.

Fail-fast mode can improve performance when used with synchronous audit logging. When a synchronous logger encounters a transient error while attempting to write to the audit logging database, the ISCoreAudit function, which is used by the logger, attempts to reconnect and write to the database after waiting the amount of time specified in the **Wait Between Retries** field for the logger. The ISCoreAudit function continues to retry connecting and writing to the database until it succeeds or the makes the maximum number of retries specified for the logger, which is specified in the **Maximum Retries** field. Additionally, if the transient error occurs because the database is unavailable, each attempt to connect to the database will pause while the logger waits for the connection attempt to fail. All of the waiting contributes to synchronous audit logging becoming very slow when the audit logging database is unavailable.

However, if the ISCoreAudit function has fail-fast enabled, the first time that audit logging fails because the database is unavailable, the affected loggers stop trying to write to the database. Any loggers that use the functional alias to request a database connection return with an exception immediately when the logger makes a request for a database connection. The logger does not make any attempts to retry. After connectivity is restored, the ISCoreAudit functional alias exits fail-fast mode The affected loggers resume normal operation. As a result of using fail-fast mode, client threads that generate audit records do not experience the pauses associated with database connection timeouts.

Note: For the initial audit log entry that cannot be written to the database because of a transient error that prevents a connection to the database, Integration Server retries writing the entry according to the **Maximum Retries** and **Wait Between Retries** properties configured for the logger. If connectivity to the database is not restored before Integration Server exhausts the retries, Integration Server writes the audit log entry to the FailedAudit log. That is, Integration Server writes the audit log entry that triggers fail-fast mode for the logger to the FailedAudit log if Integration Server makes the maximum number of retry attempts before connectivity to the database is restored.

Queue Size Considerations for Fail-Fast Mode

An important consideration when using fail-fast mode is the size of the queue for the logger. Each synchronous logger has a queue that will hold audit records received while the logger's JDBC function is in fail-fast mode. Loggers in fail-fast mode continue to accept, and queue up, audit logging records. The queues for each of these loggers must be sized appropriately. The **Maximum Queue Size** field on the **Settings > Logging > Logger Details** page must be large enough to hold all the audit logging records it may receive while the audit logging database is unavailable. If the queue reaches its maximum size, the logger writes the log entries to a file called FailedAudit*_yyyymmdd_hhmmss* .log in the *Integration Server_directory*\instances*instance_name* \logs directory. You can scan the file to find records that were not logged.

Other Considerations for Using Fail-Fast Mode with Audit Logging

The audit logging database may become unavailable because there is a problem with the database itself or because there has been an interruption in connectivity to the database.

- If a database problem caused the database to be unavailable, an attempt to write to the database usually fails quickly, which causes the functional alias to enter fail-fast mode quickly.
- If a network problem caused the database to be unavailable, it may take several seconds to a few minutes for an attempt to write to the database to fail. The first attempt to write to the database after network connectivity is lost causes the functional alias to enter fail-fast mode. However, while the functional alias is waiting for the initial failure to return, additional client requests will wait. Each operating system has settings that control how long to wait for a response on a network connection. You may need to adjust these network settings in your operating system to avoid a long wait before the functional alias enters fail-fast mode

If the database is still available and the functional alias can establish a connection to it, but the database has become slow or unresponsive, there are two connection properties that you can set on the database URL to avoid long pauses:

- `LoginTimeout` specifies the maximum number of seconds that Integration Server will wait for a new connection to be returned from the database.
- `QueryTimeout` specifies the maximum number of seconds that a read or write to the database must complete in before it is rolled back.

These properties are appended to the database URL as follows:

```
;LoginTimeout=N;QueryTimeout=M
```

Where N is the number of seconds for connections to timeout and M is the number of seconds for queries or updates to timeout, respectively.

For example:

```
jdbc:wm:dbProvider://your-db-host:1521;ServiceName=your-db-svc;LoginTimeout=5;QueryTimeout=3
```

These properties are set in the **Database URL** field of the **Settings > JDBC Pools > Connection Aliases** page.

Note: Software AG recommends that you create a separate JDBC Pool Alias for use by the ISCoreAudit JDBC Functional Alias. This way, the `LoginTimeout` and `QueryTimeout` settings will affect audit logging only.

3 Viewing Audit Log Data

- Overview 42
- View the Audit Logs in Integration Server Administrator 43
- View the API Gateway Transaction Logs 48
- Change the Log Displays 51

Overview

The following table lists the type of Audit log data that you can view using Integration Server Administrator, Monitor, or both.

Audit Log Data	View using Integration Server Administrator?	View using Monitor?
Documents	No	✓ Yes
Errors	✓ Yes	✓ Errors for logged services, documents, and processes
Guaranteed delivery	✓ Yes	No
Messaging	✓ Yes	No
Security	✓ Yes	No
Services	✓ All except logged input pipelines and user-defined messages in the IS Core Audit Log	✓ Yes
Sessions	✓ Yes	No
Business processes	No	✓ Yes
Tasks*	No	No
Integration processes	No	✓ Yes
API Gateway transaction**	No	No

* For information on viewing logged data for tasks, see *webMethods Task Engine User's Guide*.

Audit Log Data	View using Integration Server Administrator?	View using Monitor?
** To view logged data for API Gateway transactions, you must open the log file manually or look up the data in the AGW_EVENT_TXN table. For more information, see "View the API Gateway Transaction Logs" on page 48.		

Monitor links related logged data in its display; for example, for a business process or business process step, you can see all relevant service, error, and user-defined message entries. You can also perform a variety of actions from Monitor; for example, if you logged input pipelines for services, you can edit the pipelines and resubmit the services, and you can archive or delete audit log data. For complete information, see *webMethods Monitor User's Guide*.

Integration Server Administrator does not link related data for you. You must look through the individual logs for related data yourself. This chapter explains how to view audit logs in Integration Server Administrator and change various aspects of the log displays.

View the Audit Logs in Integration Server Administrator

By default, Integration Server Administrator displays the most recent entries in the logs.

View the Error Log

In Integration Server Administrator, go to the **Logs > Error** page to view the error log.

Column	Description
Time Stamp	Date and time the entry was written to the log.
Service Name	Name of the service in which the error occurred.
Service Stack	Parent services for the service in which the error occurred.
Error Message	Message that describes the error that occurred.
Stack Trace	Trace that shows the call sequence leading to the error. To expand the display of stack trace data, select the **Expand Stack Trace Data** check box in the **Log display controls** area and click **Refresh**.
Root Context **Parent Context**	Context information Monitor uses to connect related entries from different logs.

Column	Description
Current Context	

Note: For more information about interpreting the error log and using the log to help debug services, see *webMethods Integration Server Administrator's Guide*.

View the Guaranteed Delivery Log

In Integration Server Administrator, go to the **Logs > Guaranteed Delivery** page to view the guaranteed delivery log.

Column	Details
Time Stamp	Date and time the entry was written to the log.
Status	Current status of the transaction (**Start** or **Stop**).
Message	Name of the guaranteed delivery process that is running.
Error Message	If the transaction failed, message that describes the error that occurred.
Root Context **Parent Context** **Current Context**	Context information Monitor uses to connect related entries from different logs.

Integration Server writes guaranteed delivery log entries to two logs, one for inbound transactions and one for outbound transactions. By default, Integration Server Administrator displays the most recent entries in the inbound guaranteed delivery transactions log. You can switch to the log entries in the outbound transactions log by clicking **View Guaranteed Delivery Outbound Log**.

View the Messaging Log

In Integration Server Administrator, go to the **Logs > Messaging** page to view the messaging log.

Column	Description
Time Stamp	Date and time the entry was written to the log.

Column	Description
Mode	Indicates whether the message logger wrote the log entry on behalf of a message producer or consumer for webMethods messaging or JMS messaging. The messaging log uses the abbreviation MSG for webMethods messaging. For example MSG_PRODUCER and MSG_CONSUMER are the modes for webMethods messaging producer and consumer, respectively.
Message	Text of the logged message.
UUID	Universally unique identifier for the message being sent or received. ■ In webMethods messaging, you can set a value for the *_env/uuid* field when publishing the document. You can only use a user-defined UUID for messages sent to Universal Messaging by the pub.publish:publish service. If you do not set a UUID value or if you are using a service for a user-defined UUID cannot be set, Integration Server assigns one during publication. ■ For JMS messaging, you can set a UUID in the *JMSMessage/properties/uuid* field when sending the JMS message. For JMS messaging, the *uuid* field is optional. Integration Server does not generate or assign a UUID if one is not specified.
Alias	Name of the messaging connection alias associated with the sent or received message. The messaging connection alias can be a Universal Messaging connection alias or a JMS connection alias.
Destination	Destination to which the message was sent. ■ For webMethods messaging, the destination is the name of the channel to which the message was sent. The name of the channel corresponds to the publishable document type for which Integration Server published an instance document. ■ For JMS messaging, the destination contains the JNDI lookup name for the queue or topic to which the JMS message was sent.

Column	Description
Message ID	The ID assigned to the message by the messaging provider. ■ For webMethods messaging, the message ID is the Universal Messaging event ID which Universal Messaging generates when it first receives the message. ■ For JMS messaging, the message ID is the JMSMessageID which is generated on behalf of the JMS provider.
Trigger	Name of the trigger that received the message.

Note: You can search the messaging log for all log entries for a particular UUID. On the **Logs > Messaging** page, enter the UUID for which you want to search in the **UUID** field. Then, click **Refresh**.

View the Security Log

In Integration Server Administrator, go to the **Logs > Security** page to view the security log.

Column	Description
Time Stamp	Date and time the entry was written to the log.
Message	Text that explains the security event that occurred.
Server Id	Integration Server on which the security event occurred. This is necessary information when Integration Servers are clustered and writing to a shared RDBMS. The ID can be *DNSname:port* or *IPaddress:port* . **Note:** The *port* is always the Integration Server's primary port, even if the event occurred on a different (non-primary) Integration Server port.
Client Id	Network IP address for the client from which the security event was performed.
User Id	Integration Server user name under which the client connected to perform the security event.

Column	Description
Security Event Type	Category for the security event that occurred (authentication, authorization, certificates, configuration, and so on).

View the Service Log

In Integration Server Administrator, go to the **Logs > Service** page to view the service log.

Column	Description
Time Stamp	Date and time the entry was written to the log.
User Id	Integration Server user name of the client that called the service that generated the log entry.
Server Id	Integration Server on which the service that generated the log entry ran. This is necessary information when Integration Servers are clustered and writing to a shared RDBMS. The ID can be *DNSname:port* or *IPaddress:port* . **Note:** The *port* is always the Integration Server's primary port, even if a service executed on a different (non-primary) Integration Server port.
Service Name	Service that generated the log entry.
Resubmittable	Whether you can resubmit the service from Monitor. You can resubmit a service if it is a top-level (as opposed to nested) service and the service's input pipeline was logged.
Status	Current status of the service (**Started**, **Retried**, **Ended**, or **Failed**).
Duration	Length of time the service ran (in milliseconds).
Error Message	If the service failed, message that describes the error that occurred.
Root Context **Parent Context** **Current Context**	Context information Monitor uses to connect related entries from different logs.

For information about viewing service log entries in Monitor, see *webMethods Monitor User's Guide*.

View the Session Log

In Integration Server Administrator, go to the **Logs > Session** page to view the session log. The fields in the session log are listed below.

Column	Description
Time Stamp	Date and time the entry was written to the log.
Server Id	Integration Server on which the session occurred. This is necessary information when Integration Servers are clustered and writing to a shared RDBMS. The ID can be *DNSname:port* or *IPaddress:port* . **Note:** The field lists the Integration Server's primary port, even if the session occurred on a different (non-primary) Integration Server port.
User Id	Integration Server user name under which the client connected for the session.
Client Id	IP address of the machine from which the client request was submitted. The word "system" appears for session requests from Integration Server for operations such as running a scheduled service or refreshing the display.
Session State	Current status of the session (**Started**, **Expired**, or **Ended**).
RPCs	Number of services the client has called so far during the session.
Age	Duration of the session, in milliseconds.
Session ID	A string the server generates to uniquely identify each session.

View the API Gateway Transaction Logs

You cannot view logs for API Gateway transactions using the Integration Server Administrator or Monitor. You can view logs for API Gateway transactions only by manually opening either:

- **The log file (the flat file).** You can open the log file using a text editor. The log file is located in the *Integration Server_directory*\instances*instance_name* \logs \APIGateway directory.
- **The audit table.** You can open the audit table using your RDBMS editor. The table name is AGW_EVENT_TXN. For more information, see the documentation for your RDBMS editor.

The following table describes the columns in the API Gateway transaction log.

Flat File Column Name	Database Column Name	Description
API Id	**API_ID**	Unique identifier of the API that produced the audit record.
API Name	**API_NAME**	Name of the API that generated the log entry.
API Version	**API_VERSION**	Version of the API that generated the log entry.
Application Id	**APPLICATION_ID**	Unique identifier of the application associated with the call.
Application IP	**APPLICATION_IP**	IP address of the application associated with the call. This is included when the Identify and Authorize Application policy is applied for the API.
Application	**APPLICATION_NAME**	Name of the application associated with the call. This is included when the Identify and Authorize Application policy is applied for the API.
Native Endpoint	**NATIVE_ENDPOINT**	Endpoint URL of the API that generated the log entry.
Partner ID	**PARTNER_ID**	Unique identifier of the partner that produced the audit record.
Provider Roundtrip Time	**PROVIDER_TIME**	Time in milliseconds required for API Gateway to invoke an API provider and receive a response.

Flat File Column Name	Database Column Name	Description
Request Payload	**REQUEST**	Request payload. This field is written only if you use the API Gateway database component. It cannot be written to the API Gateway log file.
Response Payload	**RESPONSE**	Response payload. This field is written only if you use the API Gateway database component. It cannot be written to the API Gateway log file.
Service Name	**SERVICE_NAME**	Name of the API that generated the log entry.
Request Headers	**REQUEST_HEADERS**	Request header in the incoming request.
Response Headers	**RESPONSE_HEADERS**	Response header in the outgoing response.
Request Status	**STATUS**	Current status of the request (**Success** or **Failure**).
Target Name	**TARGET_NAME**	Name of the API Gateway instance reporting the event.
Total Roundtrip Time	**TOTAL_TIME**	Time in milliseconds required to invoke the API provider. This time includes the overhead incurred by API Gateway. Overhead includes security overhead for encryption, decryption, and load-balance retries.
Query Params	**QUERY_PARAMS**	Query parameters that are present in the incoming REST request.
Correlation ID	**CORRELATIONID**	Unique identifier that can be used to query the log.
Error Origin	**ERROR_ORIGIN**	Origin of the error - API or API Gateway.

Flat File Column Name	Database Column Name	Description
Custom Field	**CUSTOM_FIELD**	Custom field that is specified for a transaction event.

Change the Log Displays

You can change the display of log pages in Integration Server Administrator. You can:

- Display logged data in different languages.
- Change various aspects of the display for all logs permanently.
- Change various aspects of a particular log's display temporarily.

Display Logged Data in Different Languages

Note: The changes in this section will also affect the Integration Server server log, described in *webMethods Integration Server Administrator's Guide*.

This section applies only to logged data that is stored in files.

If you want to view logged data in a language other than English, you might have to adjust your text editor or command shell. Integration Server writes the files in the Unicode UTF-8 encoding. These files do not contain a Byte Order Mark (BOM, Unicode character U+FEFF). If the files contain non-ASCII data, such as log entries written in non-U.S. English, you might have to adjust the character encoding used by your text editor or command shell so you can view the log entries.

On a UNIX system, you can adjust the character encoding by changing your locale setting (LC_ALL) to the appropriate UTF-8 encoded locales. For example, to view Japanese characters in a text editor or command shell on a Solaris system, you might change your locale setting to ja_JP.UTF-8.

On a Windows system, because the files do not contain the BOM character, text editors such as Notepad might not detect the UTF-8 encoding correctly. Adjust the encoding manually so you can view the files. To view the logs in the cmd shell, you can use the command `chcp 65001`.

Change the Display Permanently for All Logs

By default, the number of log entries shown for logs in Integration Server Administrator is 35 and the refresh interval is 90 seconds. You can change these defaults.

Important: Significantly increasing the number of entries displayed or decreasing the refresh interval can slow system performance. Changing these properties

will also affect the Integration Server server log, described in *webMethods Integration Server Administrator's Guide*.

Also by default, the time stamps in the log entries default to local time and display the time zone. You can change this to the Coordinated Universal Time (UTC) that is recorded for the entries in the IS Core Audit Log database component.

To change parameter	Server configuration
Number of log entries shown	watt.server.log.maxEntries
Refresh interval for log display	watt.server.log.refreshInterval
Time stamp for log entries to UTC	watt.server.audit.displayLogs.convertTime
Date format to use in log files	watt.server.dateStampFmt

For more information on server configuration parameters mentioned in the above table, see *webMethods Integration Server Administrator's Guide*

Change the Display Temporarily for a Particular Log

To change the display for a particular log temporarily, use the **Log display controls** area at the top of the log display page and then click **Refresh**. The changes remain until you change them again, or until you shut down Integration Server, whichever comes first.

Note: If Integration Server is storing logged data in an external RDBMS, most log pages offer **From:** and **To:** fields that let you choose the entries to display using a date range. However, using date ranges can slow system performance.