webMethods Audit Logging Guide

Version 9.7

October 2014

This document applies to webMethods Product Suite Version 9.7 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-97-20141015

Table of Contents

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 storage locations for services on webMethods Integration Server, using the convention *folder.subfolder:service* .
UPPERCASE	Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+).
Italic	Identifies variables for which you must supply values specific to your own situation or environment. Identifies new terms the first time they occur in the text.
`Monospace font`	Identifies text you must type or messages displayed by the system.
{ }	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.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Documentation Installation

You can download the product documentation using the Software AG Installer. The documentation is downloaded to a central directory named _documentation in the main installation directory (SoftwareAG by default).

Online Information

Software AG Documentation Website

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

Software AG Empower Product Support Website

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 and certified samples, go to Products.

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

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, 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.

1 Concepts

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 guide describes audit logging maintained by Integration Server, as follows:

This type of logging...	Provides information about...	See page...
Error	Stack trace information about all errors that occur on Integration Server, including exceptions thrown by services.	Error Audit Logging
Session	Sessions opened on Integration Server by Software AG Designer, third-party clients, and other servers.	Session Audit Logging
Service	Services that run in Integration Server.	Service Audit Logging
Security	Security-related administrative and operational actions on Integration Server, such as modifications to authorizations and authentication, and attempts to access Integration Server resources or perform runtime events.	Security Audit Logging
Document	Documents that are in doubt, have failed, or have exhausted trigger retries on Integration Server.	Document Audit Logging
Guaranteed delivery	Guaranteed delivery transactions in Integration Server.	Guaranteed Delivery Audit Logging
Business process	Business processes modeled in Designer that run on Integration Servers.	Business Process Audit Logging

This type of logging...	Provides information about...	See page...
Task	Tasks designed in Designer that run on My webMethods Server. Tasks can be called from business processes or can run as standalone tasks.	Task Audit Logging
Integration process	Integration processes made up of a chain of services that run on Integration Servers.	Integration Process Audit Logging
Mediator transaction	webMethods Mediator transaction events produced by the Log Invocation run-time policy.	Mediator Transaction Logging

This chapter describes each type of audit logging. In addition, the chapter lists the default language used for log entries and describes the effect of your operating environment and webMethods language packs on log entries.

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

Note: webMethods Broker Document logging provides data from documents that Broker clients publish or subscribe to on Brokers. It should not be confused with the document audit logging provided with Integration Server. For more information about setting up webMethods Broker document logging, see *Administering webMethods Broker*.

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)
```

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 and the Integration Server port on which the client connected.

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.

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.

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 *Publish-Subscribe Developer's Guide*.

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*.

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*.

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.

Integration Process Audit Logging

You can log entries that track the progress and results of integration processes. To do so, 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*.

Mediator Transaction Logging

You can log Mediator transaction events produced by the Log Invocation run-time policy. This policy is enforced in Mediator, but you identify the audit log as the destination for the events in CentraSite. You can use Mediator transaction log entries to do the following:

- Identify the SOAP session, virtual service, and instance of Mediator on which the transaction events occurred.
- If the Identify Consumer policy action is defined for the virtual service, identify the IP address and name of the service consumer.
- Track whether events completed successfully or failed.
- Record the content of request and response payloads for service calls.

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

For information about identifying the audit log as the destination, see the CentraSite documentation. For information about enforcing policies with Mediator, see *Administering webMethods Mediator*.

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.

2 Setting Up Audit Logging

Overview

Integration Server writes error, session, service, security, document, guaranteed delivery, and Mediator transaction audit logging data to files or database tables collectively called the *IS Core Audit Log*. This chapter explains how to set up logging for the IS Core 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). For instructions on configuring your system to log documents that Broker clients publish to or subscribe to on Brokers, see *Administering webMethods Broker*.

Configure Audit Logging

Integration Server writes to the IS Core Audit Log using *audit loggers*. Each type of logging data has its own audit logger. For example, the error audit logger writes the audit log entries for errors, the service audit 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. The page lists the audit logger for each type of data.

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

2. In the **Logger List**, click a logger you want to set up. The page shows all settings for that logger.
3. Click **Edit *type* logger** and set the fields described below.

Enable the Logger

In the **Enabled** area, indicate whether you want the logger to write audit log entries. The default is **Yes** for all loggers except the Security logger.

Choose the Logging Level for the Service Logger

In the Level area, choose the level of logging for services. The default is **perSvc**.

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.

Choose the Mode for Writing Log Entries

In the **Mode** area, choose the mode for the logger to use to write log entries. The default is **Synchronous**.

Value	Description
Synchronous	The logger writes log entries directly to the destination (see "Identify the Destination" on page 18). 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.
Asynchronous	The logger writes log entries to a queue, and the queue later writes the entries to the destination. Each logger has its own queue.

Indicate Whether to Persist the Queue

In the Guaranteed area, indicate whether Integration Server is to persist the queue on disk when the logger is logging asynchronously. The default is **No**.

Value	Description
No	Integration Server maintains the queue in memory. This option provides better logging performance. However, if Integration Server shuts down abnormally, the log entries in the queue will be lost
Yes	Integration Server persists the queue on disk. This option is safer but can adversely affect logging performance.

Specify the Maximum Queue Size

For asynchronous logging, in the **Maximum Queue Size** field, specify the maximum number of log entries the logger's queue can hold. The default is 100000. 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. If the logger is writing to a database, the queue's insertion of logged data into the 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 Mediator logger cannot write request and response payloads to this file.

Identify the Destination

In the **Destination** area, identify where the logger is to write log entries.

Value	Description
Database	The logger writes log entries to an external RDBMS. You must also set the **Maximum Retries** and **Wait Between Retries** fields (see "Specify the Maximum Retries for Database Destination" on page 20 and "Specify the Wait Between Retries for Database Destination" on page 22). **Note:** When set to **Database**, there are times when Integration Server will automatically reset the destination to **File**. For example, when

Value	Description
	Integration Server starts 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, Mediator transaction loggers, etc.) 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 loggers destination back to **Database**.
File	Unless otherwise noted, the logger writes entries to a file in the *Integration Server_directory*\instances*instance_name* \logs directory, as follows: **Note:** This value is unavailable for the Document and Process Engine loggers, which can only write to an external RDBMS.

Logger	Log File Name
Error	WMERROR_*yyyymmdd_hhmmss* .log
Session	WMSESSION_*yyyymmdd_hhmmss* .log
Service	WMSERVICE_*yyyymmdd_hhmmss* .log
Security	WMSECURITY_*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) WMTXOUT_*yyyymmdd_hhmmss* .log (outbound transactions file, on the host machine of an Integration Server that is submitting guaranteed delivery requests)
Mediator transaction	MED_EVENT_TXN_*yyyymmdd_hhmmss* .log **Note:** The logger writes the file to the *Integration Server_directory*\instances*instance_name* \logs\Mediator directory.

Note: The IS Core Audit Log is configured either during or after Integration Server installation. For Mediator transaction events, the MediatorEvents database component is installed. For details, see *Installing webMethods and Intelligent Business Operations Products*.

Specify the Maximum Retries for Database Destination

If the destination is set to **Database**, in the **Maximum Retries** field, specify the maximum number of retry attempts Integration Server makes to log the audit data to the database if the initial attempt fails because of a transient error. The default is 3 retries. How Integration Server proceeds with retries depends on the configured logger mode:

- For synchronous logging, if the initial attempt to write to the database fails because of a transient error, Integration Server attempts to write the audit log entry asynchronously. Integration Server follows the asynchronous retry processing described in "Overview of Synchronous Logging" on page 20. Note that the logger mode does not change from synchronous to asynchronous. Integration Server attempts asynchronous logging only for the audit log entry for which synchronous logging failed. The logger will use synchronous logging for subsequent log entries.
- For asynchronous logging, if the initial attempt to write to the database fails because of a transient error, Integration Server waits the amount of time specified in **Wait Between Retries** field. Then Integration Server makes another attempt to write the audit log entry to the database. If Integration Server exhausts all the retry attempts and the retry attempts fail because of transient errors, Integration Server writes the entry to a file called FailedAudit_*yyyymmdd_hhmmss* .log in the *Integration Server_directory*\instances*instance_name* \logs directory.

If the initial attempt to write the audit log entry to the database fails because of a non-transient error, Integration Server writes the audit log entry to the 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 Mediator logger cannot write request and response payloads to this file.

Overview of Synchronous Logging

The following table provides an overview of how Integration Server performs synchronous logging for a logger, including the asynchronous retry processing that occurs when the log entry cannot be written because of a transient error.

Step	Description
1	Integration Server attempts to write the audit log entry to the database synchronously. One of the following occurs:

Step	Description
	■ Integration Server writes the audit log entry to the database successfully. ■ A transient error occurs. Integration Server writes an audit entry to the logger's queue. Integration Server proceeds to step **2**, to attempt to log the data asynchronously. ■ A non-transient error occurs. Integration Server writes the log entry to FailedAudit_*yyyymmdd_hhmmss* .log in the *Integration Server_directory*\instances*instance_name* \logs directory. Integration Server makes no further attempts to write the entry to the database.
2	Integration Server makes an initial attempt to log the data asynchronously. One of the following occurs: ■ Integration Server writes the log entry to the database successfully. ■ A transient error occurs. Integration Server writes the log entry to the retry queue proceeds to step **3**, to make retry attempts to log the data asynchronously. ■ A non-transient error occurs. Integration Server writes the log entry to FailedAudit_*yyyymmdd_hhmmss* .log in the *Integration Server_directory*\instances*instance_name* \logs directory. Integration Server makes no further attempts to write the entry to the database.
3	Integration Server begins asynchronous retry processing by making the first retry attempt to write the audit log entry to the database. One of the following occurs: ■ Integration Server writes the log entry to the database successfully. ■ A transient error occurs. Integration Server waits the length of time specified in the **Wait Between Retries** field and then makes another retry attempt. Integration Server repeats step 3 until writing the log entry successfully or making the maximum retry attempts. If the maximum retry attempts are made and writing to the database still fails because of a transient error, Integration Server proceeds to step **4**. ■ A non-transient error occurs. Integration Server writes the audit log entry to FailedAudit_*yyyymmdd_hhmmss* .log in the *Integration Server_directory*\instances*instance_name* \logs

Step	Description
	directory. Integration Server makes no further attempts to write the entry to the database.
4	When Integration Server makes the maximum number of allowed retries and writing to the database still fails because of a transient error, Integration Server writes the audit log entry to FailedAudit_*yyyymmdd_hhmmss* .log in the *Integration Server_directory*\instances*instance_name* \logs directory. Integration Server makes no further attempts to write the entry to the database.

What Errors Are Considered Transient?

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. Within the context of audit logging, whether or not an error received from the database is considered transient is determined by the contents of the following file:

Integration Server_directory\instances*instance_name* \config\auditing \transient.sql.errors.xml

During audit logging, database errors are returned from the database drivers as SQLExceptions which contain a numeric code that represents an error or warning.

- If the numeric code is listed in the transient.sql.errors.xml file, Integration Server considers the error to be transient.
- If the numeric code is *not* listed in the transient.sql.errors.xml file, Integration Server considers the error to be non-transient.

If you discover other transient errors that are not in the transient.sql.errors.xml file, modify the file to include numeric codes for those errors.

Specify the Wait Between Retries for Database Destination

If the destination is set to **Database**, in the **Wait Between Retries** field, specify the number of seconds the logger should wait between tries to connect to the RDBMS. The default is 5 seconds.

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

Additional Fields for the Security Logger

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

Choose Whether to Generate Auditing Data on Startup

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. The default is **No**.

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

Choose When to Generate Security Auditing Data

In the **Generate Auditing Data on** area, choose when Integration Server should log security events. The default is **Success** or **Failure**.

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

Choose Security Areas to Audit

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

Complete the Logger Configuration

When you are done settings the fields, click **Save Changes** and then restart Integration Server.

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.
  - 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.

- 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.

Set to...	To indicate that the message is...
`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. This is the default.

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...	Set this...
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). This is the default setting.

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

You can control the format of the time stamps, including the time zone, of entries in the audit log files by setting the following parameters:

To change this...	Set this...
Date-Time Stamp Format	`watt.server.logs.dateStampFmt =`*`format of time stamp`* The format of the date-time stamp must be compatible with the java.text.SimpleDateFormat class.
Time Zone	`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*.

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*.

3 Viewing Audit Log Data

Overview

You can use Integration Server Administrator, Monitor, or both to view audit log data.

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
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
Mediator transaction**	No	No

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

** To view logged data for Mediator transactions, you must open the log file manually or look up the data in the MED_EVENT_TXN table. For more information, see "View the Mediator Transaction Logs" on page 37.

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. The fields in the error log are listed below.

Column	Detail
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** **Current Context**	Context information Monitor uses to connect related entries from different logs.

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. The fields in the guaranteed delivery log are listed below.

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 Security Log

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

Column	Details
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* .

Column	Details
	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.
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. The fields in the service log are listed below.

Column	Details
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**).

Column	Details
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	Details
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.

Column	Details
Age	Duration of the session, in milliseconds.
Session ID	A string the server generates to uniquely identify each session.

View the Mediator Transaction Logs

You cannot view logs for Mediator transactions using the Integration Server Administrator or Monitor. You can view logs for Mediator 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\Mediator directory.
- **The audit table.** You can open the audit table using your RDBMS editor. The table name is MED_EVENT_TXN. For more information, see the documentation for your RDBMS editor.

The columns in the Mediator transaction log are listed below.

Flat File Column Name	Database Column Name	Details
Session ID	**SESSION_ID**	SOAP invocation session.
Service Name	**SERVICE_NAME**	Name of the virtual service that generated the log entry.
Target Name	**TARGET_NAME**	Name of the Mediator instance reporting the event.
Consumer	**CONSUMER_NAME**	Service consumer name associated with the call. This is included when an Identify Consumer policy action is defined for the virtual service.
Consumer IP	**CONSUMER_IP**	IP address of the service consumer. This is included when an Identify Consumer policy action is defined for the virtual service.

Flat File Column Name	Database Column Name	Details
Request Status	**STATUS**	Current status of the request (**Success** or **Failure**).
Response Payload	**RESPONSE**	Response payload. This field is written only if you use the Mediator database component. It cannot be written to the Mediator log file.
Request Payload	**REQUEST**	Request payload. This field is written only if you use the Mediator database component. It cannot be written to the Mediator log file.
Total Roundtrip Time	**TOTAL_TIME**	Time in milliseconds required to invoke the service provider. This time includes the overhead incurred by Mediator. Overhead includes security overhead for encryption, decryption, and load-balance retries.
Provider Roundtrip Time	**PROVIDER_TIME**	Time in milliseconds required for Mediator to invoke a service provider and receive a response.
Insert Timestamp	**INSERTTIMESTAMP**	Date and time the entry was written to the log. This is calculated by the RDBMS.
Timestamp	**AUDITTIMESTAMP**	Date and time the audit entry was created. This is calculated by the Integration Server.
Root Context Id	**ROOTCONTEXTID**	Globally unique identifier (GUID) for the Mediator transaction event.
Parent Context Id	**PARENTCONTEXTID**	
Context Id	**CONTEXTID**	
Message Id	**MSGID**	GUID used as the primary key for the row.

Flat File Column Name	Database Column Name	Details
Server Id	**SERVERID**	ID of the host machine that produced the audit record. 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.

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.

1. In Integration Server Administrator, go to the **Settings > Extended** page and click **Show and Hide Keys**. Integration Server Administrator displays a list of the Integration Server configuration properties you can change using this method.
2. Select the check box next to each property you want to change, as follows:

If you want to change...	Select this property...
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

3. Click **Save Changes**. Integration Server Administrator displays the selected properties in the **Extended Settings** box.
4. Click **Edit Extended Settings**. In the **Extended Settings** box, set the properties as follows:

To change...	Property	Set to...
Number of log entries shown	watt.server.log.maxEntries	Positive integer
Refresh interval for log display	watt.server.log.refreshInterval	Positive integer

To change...	Property	Set to...
Time stamp for log entries to UTC	watt.server.audit.displayLogs.contextTime	false

5. Click **Save Changes**. Changes take effect immediately.

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.