Summary:

Audit log location : \instances\instance\_name\logs\

\instances\instance\_name\config\auditing\transient.sql.errors.xml

watt.server. deprecatedExceptionLogging =true

watt.server.logs.dateStampFmt =time stamp(java.text.SimpleDateFormat )

watt.server.logs.dateStampTimeZone=time zone(java.util.TimeZone ).

watt.server.audit.logRotateSize=100KB

watt.server.audit.logFilesToKeep=10

watt.server.audit.file.fieldDelimiter=@

watt.server.audit.file.recordDelimiter=&

default:fixed-length

pub.event:addSubscriber

pub.event:auditError service.

watt.server.event.audit.async

watt.server.audit.failFastLoggers

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

IS Core Audit Logging:

API Gateway Transaction Logging

You can log API Gateway transaction events produced by the Log Invocation policy. These policies are re enforced in API Gateway.

You can use API Gateway transaction log entries to do :

* 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.
* Document Audit Logging : 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 IS. You can use error log data to debug services.
* Guaranteed Delivery Audit Logging : If you configure the guaranteed delivery capability in Integration Server, guaranteed delivery audit logging provides data about guaranteed delivery transactions. 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 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.

You can use messaging log entries to do the following:

* Track when IS 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.

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 IS and
* To access IS 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 Java services that run in IS. 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.

Session Audit Logging : provides data about sessions opened on IS by Designer, third-party clients, and other servers. You can use session log entries to do the following:

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

Audit log location : \instances\instance\_name\logs\

* API Gateway : AGW\_EVENT\_TXN\_yyyymmdd\_hhmmss.log
* Error : WMERROR\_yyyymmdd\_hhmmss.log
* Guaranteed delivery : WMTXIN\_yyyymmdd\_hhmmss.log
* Messaging : WMMESSAGING\_yyyymmdd\_hhmmss.log
* Security : WMSECURITY\_yyyymmdd\_hhmmss.log
* Service : WMSERVICE\_yyyymmdd\_hhmmss.log
* Session : WMSESSION\_yyyymmdd\_hhmmss.log

There are times when IS automatically reset the destination from Database to File (such as the Error, Session, Service, Security, Guaranteed Delivery, API Gateway transaction loggers, and so on)

* When IS 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.
* 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, IS will set the destination for the loggers back to Database.

Transient/non-transient error

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 IS\_directory \instances\instance\_name\config\ auditing\transient.sql.errors.xml file, IS considers the error to be transient.
* If the numeric code is not listed in that file, IS 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 Audit Logging

* Set the Log Level for the Messaging Logger
* Set Up Additional Service Logging
* Perform Additional Processing on Audit Log Entries

Graphical user interface, application

Description automatically generated

Enabled: indicate whether you want the logger to start writing log entries.

Level: choose the level of logging for services.

* 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; i.e. you cannot override it in Designer for individual services.

Destination: This could be any external RDBMSs or file.

database configuration

Maximum Retries: 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.

Wait Between Retries field.

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

Mode: This could be synchronous or asynchronous.

Synchronous:

* The logger writes entries directly to the destination.
* 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.

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

* Non-transient : Writes the entry to the FailedAudit.
* Transient: Switches to asynchronous mode for the entry.

Asynchronous:

* The logger writes entries to a queue, then later writes the entries from the queue to the destination.
* Each logger has its own queue.

logger tries to write an entry in asynchronous mode, and cannot write the entry to the destination because of an error.

* Non-transient: Writes the entry to the FailedAudit.
* Transient: Retries writing the entry to the destination at the interval specified on the Wait Between Retries field.
* If Transient errors occur on the retry attempts, and the Maximum Retries value is exceeded, the logger writes the entry to the FailedAudit.log.

Asynchronous Mode Configuration

* Queue Provider: logger should write to an internal queue LWQ, or to a Universal Messaging queue.
* Universal Messaging queues might provide better performance.
* The messaging logger cannot be configured to use a queue on Universal Messaging as the audit logging queue, it uses LWQ.

Guaranteed: 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.

Maximum Queue Size: specify the maximum number of entries the queue can hold. Specify numerals only.

If the queue reaches its maximum, the logger writes the log entries to a file called FailedAudit.

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.

Controlling the Level of Exception Logging Detail

Configure server configuration parameter watt.server.deprecatedExceptionLogging to control IS service exceptions logs details.

* parameter is set to false for detailed exception logging
* parameter is set to true for basic exception logging

Controlling Date-Time Stamp and Time Zone Details

* To change the Date-Time Stamp format of audit-log files, set
* watt.server.logs.dateStampFmt =format of time stamp .
  + The format 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 must be compatible with the java.util.TimeZone class.

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

Rotate Audit Logs Based on Size

By default, Integration Server rotates file-based audit logs daily.

You can configure server configuration parameter watt.server.audit.logRotateSize to rotate the audit log files by size in addition to rotating by day.

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 or invalid value(Admin UI Extended Settings wont allow) is specified for watt.server.audit.logRotateSize, Integration Server rotates the audit log files for file-based loggers daily only.

This parameter affects only the audit loggers configured write to a file.

Limit the Number of Audit Log Files Kept by IS

By default, Integration Server keeps audit log files for file-based audit loggers indefinitely.

server configuration parameter watt.server.audit.logFilesToKeep is used for limiting the number of audit log files.

This 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.

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.

If you set watt.server.serverlogFilesToKeep to 0, or -ve no, Integration Server keeps an unlimited number of audit log files

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.

Use Delimiters in a File-Based Audit Log

The fields for an entry in a file-based audit log can be fixed length(defaltand size of the fields determines) or character delimited.

In fixed-length, if the actual data for the field is less than the defined size, then Integration Server pads the field with whitespace to maintain constant field widths.

But this is not efficient with large volume of entries To address this resource inefficiency, you can change the audit loggers to write character-delimited log entries instead of fixed length.

* 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.
* These parameters affect all audit logs written to a file including the FailedAudit log.
* If no value is specified for the parameters, Integration Server writes log entries using the fixed-length format.
* A valid delimiter is any visible character sequence (0021-007E) from the US ASCII character set.
* The values for fieldDelimiter and recordDelimiter must be different.
* You must restart Integration Server for changes to take effect.

Receiving Notifications When Logging Fails

You can subscribe to audit error events to notify administrators when IS 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 inserting an audit record into the audit logging database.
* When IS 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.

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.

check the webMethods Integration Server Built-In Services Reference for detailed info.

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.

Fail-Fast Mode for Synchronous Logging

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.

With this IS eliminates the time a request spends making retry attempts thus improve performance.

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, all synchronous loggers with a database destination will have fail-fast capability.

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 IS 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.

When a synchronous logger encounters a transient error while attempting to write to the audit logging database, the ISCoreAudit function, 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 (in Maximum Retries field) for the logger.,

Additionally, if 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.

If connectivity to the database is not restored before IS exhausts the retries, Integration Server writes the audit log entry to the FailedAudit log.

Each synchronous logger has a queue that holds 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 must be large enough to hold all the audit logging records it may receive while the audit logging database is unavailable.

The logger writes the log entries to a file called FailedAudit\_yyyymmdd\_hhmmss.log when the queue reaches its maximum size.

If the database is available so that functional alias can establish a connection to it, but it is 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 IS 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 N and M are in seconds.

Example:

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

Best Practices:

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.

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(even enabled logger).