Oracle® Fusion Middleware

Oracle® Fusion Middleware Administering the Store-and-Forward Service for Oracle WebLogic Server

15c (15.1.1.0.0)
G31940-01
October 2025

Oracle Fusion Middleware Oracle® Fusion Middleware Administering the Store-and-Forward Service for Oracle WebLogic Server, 15c (15.1.1.0.0)

G31940-01

Contents

Preface

Audience i
Documentation Accessibility i
Diversity and Inclusion i
Related Documentation i
Conventions ii

1 Understanding the Store-and-Forward Service

The WebLogic SAF Environment 1
- The SAF Service 1
- SAF Service Agents 1
  - SAF Agent Configuration Parameters 2
  - Persistent Store Rules When Targeting SAF Agents to a Cluster 2
- Using SAF with WebLogic JMS 3
- SAF and Cross Domain Security 3

When to Use the SAF Service 3
Configuring a Basic SAF Service 4
Designing SAF Agents 5
- Setting a Message Time-To-Live Duration and Message Delivery Failure Policy 5
- Logging Failed Message Deliveries 5
- Setting Delivery Retry Attempts 5
- Using Message Quotas, Thresholds, and Paging 6
- Boot-Time Recovery 6
- Migratable SAF Agents for Highly Available Messaging 6

2 Configuring SAF for JMS Messages

SAF Resources in a JMS Module 1
- Imported SAF Destinations 1
- Remote SAF Context 2
- SAF Error Handling 2
- The SAF JMS Picture 3

Creating JMS SAF Resources 3

Designing SAF for JMS Messages 4
Selecting a Quality-of-Service (QOS) Level 4
How SAF Handles Delivery Modes 4
Using Message Unit-of-Order 4
Transactional Messages 5
Message Compression Across SAF Boundaries 5
SAF to a Distributed Destination 5
SAF Load Balancing 5
Sending Messages to an Imported Destination 5
SAF Agent Forwarding Messages to a Remote Destination 6
Using the JMSReplyTo Field with SAF 6
Securing SAF Destinations 7
Connections and ConnectionFactories 7

3 Monitoring and Managing SAF Agents

Monitoring SAF Agents 1
Managing Message Operations on SAF Agents 1

4 Troubleshooting WebLogic SAF

Debugging WebLogic SAF 1
Enabling Debugging 1
Enable Debugging Using the Command Line 1
Enable Debugging Using the WebLogic Remote Console 1
Enable Debugging Using the WebLogic Scripting Tool 2
Changes to the config.xml File 3
SAF Debugging Scopes 3
Request Dyeing 4
SAF Message Life Cycle Logging for JMS Messages 4
Events in the SAF Message Life Cycle 4
Message Log Location 4
Enabling SAF Message Logging 5
SAF Message Log Content 5
SAF Message Log Record Format 5
Sample Log File Records 6
Message Stored Event 6
Message Forwarded Event 7
Message Expired Event 7
Message Removed Event 7
Managing SAF Agent Log Files 8
Rotating Message Log Files 8

Renaming Message Log Files 8
Limiting the Number of Retained Message Log Files 8
Frequently Asked Questions About JMS SAF 8

Preface

This document is a resource for system administrators responsible for configuring, managing, and monitoring the WebLogic Store-and-Forward service for use with WebLogic JMS and Web Services Reliable Messaging (WSRM).

Audience

The information in this document is relevant to production phase administration, monitoring, or performance tuning topics. This document does not address the pre-production development or testing phases of a software project.

It is assumed that the reader is familiar with WebLogic Server system administration. This document emphasizes the value-added features provided by WebLogic Store-and-Forward (SAF) and key information about how to use WebLogic Server features and facilities to maintain WebLogic Server in a production environment.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Documentation

Samples and Tutorials

Oracle provides a variety of code examples and tutorials that show WebLogic Server configuration and API use, and provide practical instructions on how to perform key

development tasks. For more information, see Sample Applications and Code Examples in *Understanding Oracle WebLogic Server*.

SAF Specific Configuration and Maintenance Information

For comprehensive guidelines for developing, deploying, and monitoring WebLogic Server applications, see the following documents:

- *Administering JMS Resources for Oracle WebLogic Server* contains instructions for configuring and managing JMS resources, such as JMS servers, JMS modules, and the Path Service.
- Using the WebLogic Persistent Store in *Administering the WebLogic Persistent Store* provides information about the benefits and usage of the system-wide Persistent Store.
- Tuning WebLogic JMS Store-and-Forward in *Tuning Performance of Oracle WebLogic Server* provides information on how to get the best performance from Store-and-Forward (SAF) applications.

New WebLogic Server Features

For a comprehensive listing of the new WebLogic Server features introduced in this release, see *What's New in Oracle WebLogic Server*.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Understanding the Store-and-Forward Service

The WebLogic store-and-forward (SAF) service enables Oracle WebLogic Server to deliver messages reliably between applications that are distributed across WebLogic Server instances. For example, with the SAF service, an application that runs on or connects to a local WebLogic Server instance can reliably send messages to an endpoint that resides on a remote server. If the destination is not available at the moment the messages are sent, either because of network problems or system failures, then the messages are saved on a local server instance and are forwarded to the remote endpoint once that endpoint becomes available.

The WebLogic SAF Environment

The WebLogic SAF environment includes the SAF service and SAF agents, and may also include JMS SAF, web services reliable messaging, and Cross-Domain Security.

The SAF Service

The SAF service enables WebLogic Server to deliver messages reliably between applications that are distributed across WebLogic Server instances. For example, with the SAF service, an application that runs on or connects to a local WebLogic Server instance can reliably send messages to an endpoint that resides on a remote server. If the destination is not available at the moment the messages are sent, either because of network problems or system failures, then the messages are saved on a local server instance, and are forwarded to the remote endpoint once it becomes available.

WebLogic JMS utilizes the SAF service to enable local JMS message producers to reliably send messages to remote JMS queues or topics, as described in Using SAF with WebLogic JMS.

SAF Service Agents

There are two sides involved in the process of storing and forwarding messages: a local sending side and a remote receiving endpoint. SAF agents are responsible for store-and-forwarding messages between these local sending and remote receiving endpoints. A SAF agent can be configured to have only sending capabilities, receiving capabilities, or both.

JMS SAF only requires a sending agent on the sending side for JMS messages. Whereas, WSRM SAF requires both a *sending agent* and a *receiving agent*.

- **Sending agent** — Used for JMS messages and WSRM. If message persistence is required, a sending agent stores messages to a persistent storage, forwards messages to the receiving side, and re-transmits messages when acknowledgements do not come back in time.
- **Receiving agent** — Used only for WSRM. Receiving agents detect and eliminate duplicate messages sent by a sending agent, and delivers messages to the final destination.

> **Note**
>
> In the case of JMS messages, the JMS server associated with a remote exported JMS destination on the receiving side manages duplicate elimination.

SAF Agent Configuration Parameters

A SAF agent is a configurable object that is similar to a JMS server in that it manages message persistence, paging parameters, and thresholds and quotas. The working behavior of the SAF service is controlled by a number of configurable parameters on SAF agents:

- General configuration parameters, including:
  - select persistent storage
  - setting message paging defaults
  - specify delivery retry settings
  - determine window size
  - specify message acknowledgment intervals (WSRM only)
- Threshold and quota parameters for controlling the message throughput of SAF agents.

  See Designing SAF Agents.
- Monitoring capabilities for SAF agents, remote endpoints, and conversations. You also have the capability to manage incoming, forwarding, and receiving message operations on SAF agents and to remote endpoints.

  See Monitoring and Managing SAF Agents.
- JMS message logging capabilities for imported SAF destinations. This is an external view of the events that a JMS message traverses through once it has been stored by a SAF agent and eventually forwarded to a remote JMS destination.

  For more information about JMS message logging options for SAF destinations, see SAF Message Life Cycle Logging for JMS Messages.

For more information about delivery retry settings, logging, and paging defaults, see Designing SAF Agents.

For information about configuration parameters for SAF agents in the WebLogic Remote Console, see Create a Store-and-Forward Agent in the *Oracle WebLogic Remote Console Online Help* and click **Show Advanced Fields** to see all of the options.

Persistent Store Rules When Targeting SAF Agents to a Cluster

When targeting an SAF agent to a standalone server, you can either use the server's default persistent store or select an explicitly configured store. However, the following persistent store selection rules apply in order to properly target SAF agents in a cluster.

- When targeted to a cluster (static or dynamic), an agent instance is automatically generated on each server instance in the cluster. In this configuration, we recommend that an SAF agent use a custom persistent store that is also targeted to the same cluster. For more information, see Migratable SAF Agents for Highly Available Messaging.
- When targeted to a static cluster and no persistent store is configured, an SAF agent uses the default persistent store of each WebLogic server instance.

- When targeted to a migratable target, an SAF agent must use a custom persistent store *and* it must be targeted to the same migratable target used by the custom store. An SAF agent, JMS server, and custom store can share a migratable target. See Understanding Migratable Targets In a Cluster in *Administering Clusters for Oracle WebLogic Server*.

> **Note**
>
> To preserve SAF message consistency, WebLogic Server prevents you from retargeting an existing SAF agent to a migratable target. Instead, you must delete the existing SAF agent and configure a new one with the same values and target it to a migratable target.

Using SAF with WebLogic JMS

The JMS store-and-forward uses a single sending SAF agent to provide highly-available JMS message production. For example, a JMS message producer connected to a local server instance can reliably forward messages to a remote JMS destination, even though that remote destination may be temporarily unavailable when the message was sent. JMS SAF is transparent to JMS applications; therefore, JMS client code still uses the existing JMS APIs to access remote destinations. See Configuring SAF for JMS Messages.

To ensure that a sending SAF agent for imported JMS destinations does not introduce a single point of failure for dependent JMS applications in a cluster, WebLogic Server can be configured to migrate SAF agents to a healthy server instance in the cluster. See Migratable SAF Agents for Highly Available Messaging.

SAF and Cross Domain Security

SAF services do not require you to configure Cross Domain Security. However, you will need to configure Cross Domain Security for all the domains your process communicates with *if* Cross Domain Security is configured on one domain *and* the membership of the Uniform Distributed Destinations are imported through SAF changes. A best practice is to keep all the domains used by your process symmetric, with respect to Cross Domain Security configuration— that is, all domains use Cross Domain Security (or are on the appropriate exception lists) or none of the domains have Cross Domain Security configured. See Configuring Cross-Domain Security in *Administering Security for Oracle WebLogic Server*.

When to Use the SAF Service

The SAF service should be used when forwarding JMS or WSRM messages between WebLogic Server 9.*x* or later domains.

The SAF service can deliver messages:

- Between two stand-alone server instances.
- Between server instances in a cluster.
- Across two clusters in a domain.
- Across separate domains.

When not to use the SAF service:

- Forwarding messages to instances of WebLogic Server prior to version 9.0.

- Interoperating with third-party JMS products (for example, MQSeries).

  For these tasks, you should use the WebLogic Messaging Bridge. See Understanding the Messaging Bridge in *Administering the WebLogic Messaging Bridge for Oracle WebLogic Server*.

- When using temporary destinations with the `JMSReplyTo` field to return a response to a request.

Additionally, when you use JMS SAF, an application can receive messages only and directly from a remote server that is available.

Configuring a Basic SAF Service

The main tasks for implementing a basic SAF service include creating and configuring the SAF sending and receiving agents. Additional tasks may be required for WSRM, WebLogic JMS, and the persistent store.

To implement the SAF service in a domain, complete the following tasks:

1. Configure a SAF agent in the sending-side cluster or server instance(s). There are a number of ways to create SAF agents:

   - The WebLogic Remote Console enables you to configure, modify, target, monitor, and delete SAF agents in your environment.

     For step-by-step instructions of the SAF agent configuration tasks, see Create a Store-and-Forward Agent in the *Oracle WebLogic Remote Console Online Help*.

   - WebLogic Java Management Extensions (JMX) enables you to access the `SAFAgentMBean` and `SAFAgentRuntimeMBean` MBeans to create and manage SAF agents. See Overview of WebLogic Server Subsystem MBeans in *Developing Custom Management Utilities Using JMX for Oracle WebLogic Server*.

   - The WebLogic Scripting Tool (WLST) enables you to create and manage JMS servers and JMS system resources. See Using WLST to Manage JMS Servers and JMS System Resources in *Administering JMS Resources for Oracle WebLogic Server*.

2. When using WSRM, even if you configure a SAF agent with sending *and* receiving capability on the sending side, you *must* also configure a SAF agent with receiving capability on the receiving-side cluster or server instance.

3. When configuring a SAF agent, you can accept the server's default store or configure a store if you want a dedicated store for SAF messages. When targeting a SAF agent to a cluster, you must accept the server's default store. You can also target a SAF agent to a migratable target for high availability, as described in Persistent Store Rules When Targeting SAF Agents to a Cluster.

   See Using the WebLogic Persistent Store in *Administering the WebLogic Persistent Store*.

4. For WebLogic JMS, configure SAF Imported Destination, SAF Context Handling, and SAF Error Handling (optional) objects in a JMS module, as described in Configuring SAF for JMS Messages.

5. Optionally, configure a Path Service for JMS messages when the sending-side is a cluster and the JMS producer is associated with a *message unit-of-order*. A message unit-of-order enables JMS message producers to group ordered messages into a single unit. See Using Message Unit-of-Order in *Developing JMS Applications for Oracle WebLogic Server*.

   The Path Service is a persistent map that can be used to store the mapping of a group of messages to a messaging resource such as a SAF agent. See Using the WebLogic Path Service in *Administering JMS Resources for Oracle WebLogic Server*.

Designing SAF Agents

When designing and configuring SAF agents for forwarding messages, you should take into consideration the requirements of your operational environment, such as the time needed by a SAF agent to reliably deliver a message, the logging of failed message deliveries, setting delivery retry attempts, and more.

Setting a Message Time-To-Live Duration and Message Delivery Failure Policy

The reliability of SAF is time based. You can configure the duration that a message needs to be delivered by a SAF agent reliably using the Time-To-Live parameter. When the configured Time-To-Live duration expires, the sending agent removes the message from its storage and discontinues attempts to retransmit the message to the receiving side. It is up to the application to decide what to do with the messages that fail to be delivered when its Time-To-Live expires.

For information on how JMS handles failed message deliveries, see SAF Error Handling.

Messages can fail to be delivered for the following reasons:

- Network outage
- The endpoint does not exist (not configured)
- The endpoint is down
- Endpoint quota failure
- Security denial
- Required QOS is not supported
- The WSRM conversation times out (the conversation is idle for too long).

Logging Failed Message Deliveries

If the Logging parameter is enabled for the sending agent, it logs a message in the server log for every failed message. This is an alternative for applications that do not have their own failure handling or their failure handling cannot complete. See Understanding WebLogic Logging Services in *Configuring Log Files and Filtering Log Messages for Oracle WebLogic Server*.

Setting Delivery Retry Attempts

The sending agent needs to connect to the receiving side in order to forward messages over, yet there are times that the connection may not be available. When an attempt to send a message fails, the sending agent must retry until it succeeds. Similarly, if the desired QOS is Exactly-Once or At-Least-Once, the sending agent must keep sending a message until it receives an acknowledgement for that message.

To control the frequency of attempts and the interval between two subsequent attempts, you can configure the default values for Retry Delay Base, Retry Delay Multiplier, and Retry Delay Maximum parameters. The Retry Delay Multiplier must be greater or equal to 1, and the Retry Delay Maximum value must greater or equal to the Retry Delay Base value. By default, the Retry Delay Multiplier value is set to 1, which means there's a fixed interval, defined by the Retry Delay Base value, between two successive attempts and that Retry Delay Maximum will

be ignored. If the Retry Delay Multiplier is greater than 1, an exponential back-off algorithm will be used to adjust the retry intervals.

The delays will be exponentially increased, starting from the Retry Delay Base value, and will be multiplied by the Retry Delay Multiplier value each time. The amount of delay will not be increased any more once the Retry Delay Maximum value is reached. Once there is a successful attempt, the Retry Delay Multiplier value is reset to the Retry Delay Base value.

Using Message Quotas, Thresholds, and Paging

Persistent messages are saved in the persistent store on the sending side until they are successfully forwarded to and acknowledged by the receiving side. However, non-persistent messages pending for delivery exist in-memory on the sending side, and all of the history records also exist in-memory on the receiving side. If the remote side is not available for a long time, the pending non-persistent messages could use up the sending side server's memory and even take the server down. By configuring quotas for each SAF agent, you can protect the server from running out of memory. Once the quota is about to be exceeded, the SAF agent will reject any new requests.

You can also configure SAF agents to page out messages or history records to a paging directory before the agent reaches the quotas. Paging will be triggered by certain conditions specified as thresholds in the SAF agent's configuration. The persistent store for messages or history records is also used for paging purposes.

The SAF agent threshold and quota parameters and relationship are the same as for JMS destinations and JMS servers.

Boot-Time Recovery

When a WebLogic Server instance reboots, the messages that were not sent before the server instance went down are recovered from the server's persistent store. The sending agent then attempts to send those messages to the remote side if they have not expired. Similarly, on the receiving side, the history records are recovered during reboot.

Migratable SAF Agents for Highly Available Messaging

To ensure that a sending SAF agent for imported JMS destinations does not introduce a single point of failure for dependent JMS applications in the cluster, WebLogic Server can be configured to automatically or manually migrate SAF agents to a healthy server instance in a cluster.

You can use cluster targeted SAF agents and automatically migrate the agents and their custom stores to healthy servers in your WebLogic cluster. See the section Simplified JMS Cluster and High Availability Configuration in *Administering JMS Resources for Oracle WebLogic Server*.

For service migration, you can also use a *migratable target*, which serves as a grouping of pinned JMS-related services, such as JMS servers, SAF agents, and persistent stores, and that migratable target is hosted on only one physical server in a cluster. Such hosted services can be automatically migrated from the current unhealthy hosting server to a healthy active server with the help of the Health Monitoring subsystem. JMS services hosted by a migratable target can also be manually migrated, either in response to a server failure or as part of regularly scheduled maintenance. When the migratable target is migrated, all pinned services hosted by that target are also migrated.

See Service Migration in *Administering Clusters for Oracle WebLogic Server*.

2

Configuring SAF for JMS Messages

The JMS store-and-forward (SAF) service builds on the WebLogic SAF service to provide highly available JMS message production. For example, a JMS message producer connected to a local server instance can reliably forward messages to a remote JMS destination, even though that remote destination may be temporarily unavailable when the message was sent. JMS SAF is transparent to JMS applications; therefore, JMS client code still uses the existing JMS APIs to access remote destinations.

SAF Resources in a JMS Module

When configuring the SAF resources for a JMS module, you must configure the SAF imported destinations, the remote SAF context, and the SAF error handling resources, in a JMS system module or application module. JMS configuration resources are stored outside of the WebLogic domain as module descriptor files, which are defined by XML documents that conform to the `weblogic-jmsmd.xsd` schema. JMS modules also provide the configuration of SAF resources that allow WebLogic Server to store and forward JMS messages. See Understanding JMS Resource Configuration in *Administering JMS Resources for Oracle WebLogic Server*.
After your JMS SAF resources are configured, a configured SAF sending agent forwards messages to the receiving side; retransmits messages when acknowledgements do not come back in time; and, if message persistence is required, temporarily stores messages in persistent storage. See Understanding the Store-and-Forward Service.

JMS SAF is transparent to JMS applications. Existing JMS applications can take advantage of this feature without any code changes. In fact, you only need to configure imported JMS destinations within JMS modules, which then associate remote JMS destinations to local JNDI names. JMS client code still uses the existing JMS APIs to access the imported destinations. JMS store-and-forward is only for message production; therefore, JMS clients still need to consume messages directly from imported destinations.

The following topics explain how to configure the SAF resources for a JMS module:

Imported SAF Destinations

A SAF *destination* (queue or topic) is a local representation of a JMS destination (queue or topic) in a JMS module that is associated with a remote server instance or cluster. Such remote destinations are *imported* into the local cluster or server instance so that the local server instance or cluster can send messages to the remote server instance or cluster. All JMS destinations are automatically exported by default, unless the Export SAF Destination parameter on the destination is explicitly disabled.

A collection of imported SAF destinations is called *SAF imported destinations*. Each collection of imported destinations is associated with a SAF *remote context*. They can also share the same JNDI prefix, time-to-live default (message expiration time), and SAF error handling object.

When a JMS producer sends messages to a SAF destination, these messages are stored on the SAF destination for future delivery. A SAF agent forwards the messages to the remote JMS destination (that the imported destination represents) when the destination is reachable, using the SAF Remote Context.

Remote SAF Context

A remote SAF context defines the URL of the remote server instance or cluster where the JMS destination is exported from. It also contains the security credentials to be authenticated and authorized in the remote cluster or server. A SAF remote context configuration is required to use imported destinations. A remote SAF context can be re-used by multiple SAF imported destination configurations.

Here's an example of an URL used when a remote SAF context defines a single, remote managed server from which it will import standalone JMS destinations:

```
<URL>
t3://123.0.0.1:7001
</URL>
```

To import a distributed destination from a remote cluster you need to supply a comma-delimited list of DNS Server names or IP addresses. Here's an example of an URL used when a remote SAF context defines a remote cluster from which it will import distributed destination members:

```
<URL>
t3://123.0.0.1:7001,123.0.0.1:7002,123.0.0.1:7003
</URL>
```

For more information about specifying the initial point of contact with a WebLogic Cluster, see Using WebLogic JNDI in a Clustered Environment in *Developing JNDI Applications for Oracle WebLogic Server*.

SAF Error Handling

SAF error handling resources define the action to be taken when the SAF service fails to forward messages to a remote destination. SAF error handling resources are not required for imported SAF destinations, but are recommended as a best practice.

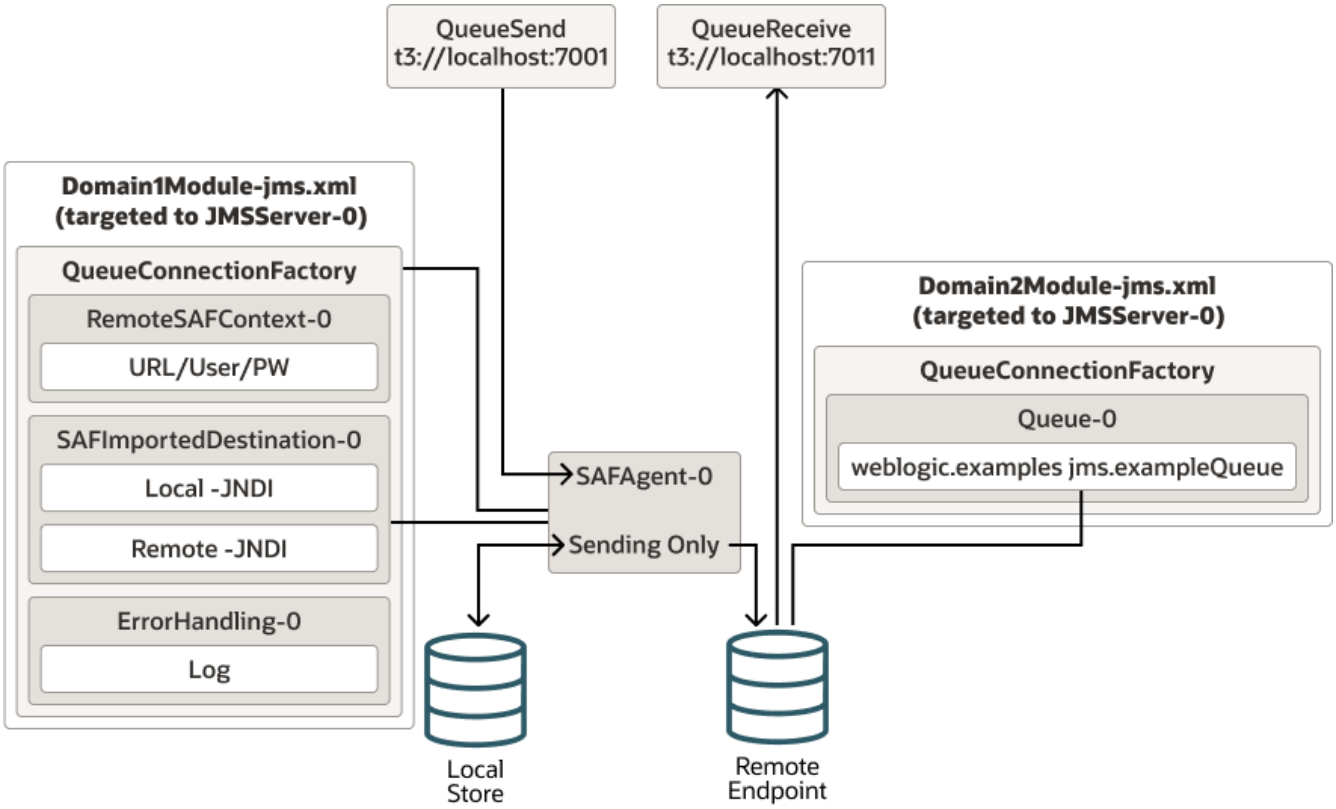
Configuration options include the following parameters:

- Error Handling Policy:
  - Discard – By default, expired messages are simply removed from the system. The removal is not logged and the message is not redirected to another location.
  - Log – Removes expired messages and writes an entry to the server log file indicating that the messages were removed from the system. You define the actual information that will be logged in the Log Format field.
  - Redirect – Moves expired messages from their current location into the Error Destination defined for imported SAF destinations.
  - Always-Forward – Ignores the SAF Default Time-to-Live value set on the imported destination and the expiration time set on the message, and so forwards the message to a remote destination even after it has expired. This options is useful for situations where an application has expiration policies set up on the remote destination, and they want that messages still go through the expiration process on the remote destination.
- Log Format – If you selected the Log policy in previous step, use this field to define what information about the message is logged.
- Error Destination – If you select the Redirect policy, use this field to select a local SAF destination where you want expired messages to be redirected.

The SAF JMS Picture

Figure 2-1 illustrates how JMS messages that are produced to the `QueueSend` queue in Domain1Module-jms.xml module in Domain1 are forwarded by a SAF agent to the `QueueReceive` queue in the Domain2Module-jms.xml module in remote Domain2.

Figure 2-1 Store-and-Forward JMS Messages



Creating JMS SAF Resources

WebLogic Server supports several different methods for creating the SAF resources in a JMS module. These methods include the following:

- The WebLogic Remote Console enables you to configure, modify, target, monitor, and delete JMS system modules and JMS system resources in your environment.

  For JMS SAF resource configuration tasks, see Create a Store-and-Forward Agent in the *Oracle WebLogic Remote Console Online Help*.
- The WebLogic Scripting Tool (WLST) enables you to create and manage JMS servers and JMS system resources. See Using WLST to Manage JMS Servers and JMS System Module Resources in *Administering JMS Resources for Oracle WebLogic Server*.
- WebLogic Java Management Extensions (JMX) enables you to access the `SAFImportedDestinationsBean`, `SAFRemoteContextBean`, and `SAFErrorHandlingBean` management MBeans to create and manage the SAF resources in JMS modules. See Overview of WebLogic Server Subsystem MBeans in *Developing Custom Management Utilities Using JMX for Oracle WebLogic Server*.

- JMS Module Helper Extension APIs enable you to locate JMS runtime MBeans, as well as methods to manage (locate/create/delete) JMS system module configuration resources in a given module. See Using JMS Module Helper to Manage Applications in *Developing JMS Applications for Oracle WebLogic Server* or `JMSModuleHelper` in the *Java API Reference for Oracle WebLogic Server*.

Designing SAF for JMS Messages

When you design and configure WebLogic SAF for storing and forwarding JMS messages, you should give careful consideration the specific settings for quality-of-service, message delivery modes, message unit-of-order, transaction rollback, load balancing, and so on, that are appropriate for your operational environment.

The following topics explain how to design and configure WebLogic SAF for storing and forwarding JMS messages.

Selecting a Quality-of-Service (QOS) Level

Persistent JMS messages are always forwarded with Exactly-Once QOS provided by the SAF service. For non-persistent messages, three different QOS levels can be defined on imported SAF queues and topics:

- Exactly-once—The highest QOS guarantees that a message is forwarded to the remote endpoint once and only once. With Exactly-once, messages survive server crashes and network down time, while guaranteeing one occurrence of each message at the endpoint.
- At-least-once—Guarantees that a message is forwarded to the remote endpoint at least once, but with the possibility of duplicates. With At-least-once, multiple copies of a message might show up on the remote endpoint because of network failures or server crashes that occur when the message is in transit.
- At-most-once—The lowest QOS guarantees that each message is forwarded to the remote endpoint only once, if at all. It does not guarantee that a message is forwarded to the endpoint. With At-most-once, messages may get lost because of network failures or server crashes. No duplicate messages will reach the endpoint.

How SAF Handles Delivery Modes

A SAF application can also specify a delivery mode for each message, as follows:

- Persistent messages are saved in the persistent store on the sending side until they are successfully forwarded to and acknowledged by the receiving side.
- Non-persistent messages are kept in memory on the sending side until the receiving side acknowledges them. This means that non-persistent messages can be lost if the sending side crashes.

Using Message Unit-of-Order

Within a cluster, a JMS producer can be associated with a *message unit-of-order*, which enables a stand-alone message producer, or a group of producers acting as one, to group messages into a single unit with respect to the processing order. See Using Message Unit-of-Order in *Developing JMS Applications for Oracle WebLogic Server*.

Imported SAF destinations can use either a Hash Map or a Path Service to group ordered messages in a cluster. However, as a best practice, you should configure a Path Service. The Path Service is a persistent map that can be used to store the mapping of a group of

messages to a messaging resource such as a SAF agent. See Using the WebLogic Path Service in *Administering JMS Resources for Oracle WebLogic Server*.

Transactional Messages

If an application message is in a transaction, saving the message in the persistent storage becomes part of the user transaction to preserve Exactly-Once semantics.

In particular, the message will be removed from the persistent storage as part of the transaction rollback if the application decides to rollback the transaction. However, forwarding is *not* part of the application transaction. The sending agent will not forward a transactional message until the transaction commits. Within a transaction, message ordering is preserved based on when the messages are sent.

Message Compression Across SAF Boundaries

JMS store-and-forward can compress messages when they are forwarded between different clusters. A message compression threshold can be set programmatically using a JMS API extension to the `WLMessageProducer` interface, or administratively by either specifying a Default Compression Threshold value on a connection factory or on a JMS remote SAF context.

For more information, on using message compression for JMS messages, see Compressing Messages in the *Tuning Performance of Oracle WebLogic Server*.

When an uncompressed message that exceeds the remote SAF context's compression threshold value is about to be forwarded across the SAF boundary, it is compressed. The message stays compressed until it is received by the remote endpoint. If the message has already been compressed when it crosses the SAF boundary because the compression is turned on the connection factory, the message will stay compressed across SAF boundary no matter if the SAF compression is triggered or not.

SAF to a Distributed Destination

A remote endpoint can be a distributed destination. In general, messages to a remote distributed destination are stored and forwarded using the same model as messages that are forwarded to remote standalone destinations. However, load balancing is handled differently than other distributed destinations, see SAF Load Balancing.

See Configuring Distributed Destinations in *Administering JMS Resources for Oracle WebLogic Server*.

SAF Load Balancing

Load balancing determines the message route taken by a message when it is sent to an imported SAF destination or forwarded to a remote destination. There are two situations to consider when configuring WebLogic Server to determine a message route for SAF messages:

Sending Messages to an Imported Destination

Messages sent by a JMS client are load balanced to an imported destination using one of the following message routes:

- For a single SAF agent, load balancing is handled in the same manner as sending to a regular JMS destination.

- For multiple SAF agents, the imported destination behaves similarly to a distributed destination. The connection factory load balancing setting load balances messages among the SAF agents. See Connections and ConnectionFactories.
- You can choose whether messages should be distributed among all running members (`Per-Member` - the default), or only to a single member per JVM (`Per-JVM`) using the `Exactly Once Load Balancing` policy. See Load-Balancing Heuristics in *Administering JMS Resources*.

SAF Agent Forwarding Messages to a Remote Destination

When forwarding messages, each SAF agent forwards messages independently. In this case, SAF agents forwarding messages are not load balanced, the message route is determined by the destination type:

- If the destination is a regular non-distributed destination, there is no load balancing or fail over.
- If the target destination is a distributed destination, the message route and failover behavior is determined by the configuration of the SAF agent for a non-Unit -of-Order `Exactly-once` message, where the message is either `persistent` or `non-persistent` with a QoS of `ExactlyOnce`. The message route is determined by round-robin load balancing messages among a list of available distributed destination members. If a member becomes unavailable, that member is removed dynamically from the list. If a member becomes available, it is dynamically added to the list.

  From a single message perspective, there is no failover. If the SAF agent fails to deliver a message to a member destination, there is no attempt to redeliver to another member; instead, the SAF agent periodically retries delivery to the original member. The message "sticks" to its target member to preserve the `ExactlyOnce` QoS.

  For even load balancing of messages across multiple JVMs, the candidate members for non-Unit -of-Order `Exactly-Once` QoS forwarded messages can be restricted to a single member of the distributed destination on each remote JVM. This behavior is configurable using the `Imported Destination ExactlyOnceLoadBalancingPolicy` attribute or a command-line system property. See Load-Balancing Heuristics in *Administering JMS Resources* for details.
- For all other message types, the message route is determined by the settings of the default connection factory (`loadBalancingEnabled=true` and `serverAffinityEnabled=true`).

> **Note**
>
> Default JMS connection factory settings cannot be changed or overridden.

Using the JMSReplyTo Field with SAF

Generally, JMS applications can use the `JMSReplyTo` header field to advertise its temporary destination name to other applications. However, the use of temporary destinations with a `JMSReplyTo` field is not supported for SAF imported destinations.

See Using Temporary Destinations in *Developing JMS Applications for Oracle WebLogic Server*.

Securing SAF Destinations

The following security measures apply to SAF imported destinations.

- Secure roles and policies can be set on imported SAF queues and topics.
- The SAF service does not preserve a message's `JMSXUserID` across SAF boundaries. A `JMSXUserID` is a system generated property that identifies the user sending the message. `JMSXUserID` is defined in the JMS Specification, published at http://www.oracle.com/technetwork/java/jms/index.html.

Connections and ConnectionFactories

JMS messages route from a client, through its connection host, and then on to a JMS destination instance hosted by a WebLogic JMS server or SAF Agent. Clients use JNDI to obtain a connection factory, and then use the connection factory to create a JMS connection to WebLogic. Each connection instance is pinned to a particular WebLogic server (the "connection host").

A connection factory defines configuration parameters for a client connections, and must be hosted on the same server or cluster as the client's destinations. WebLogic JMS provides default connection factories, and custom connection factories. Default connection factories are not configurable, while custom connection factories are configurable but cannot be configured with the same JNDI name as a default connection factory. A connection factory can be hosted on one or more servers in a WebLogic cluster:

- Default connection factories are always hosted on every server in their cluster.
- Custom connection factory hosts are set based on their target configuration.

A client's connection factory instance is actually a WebLogic RMI replica-aware stub that implicitly keeps track of all active servers that host the connection factory. As servers become available and unavailable in a cluster, the connection factory instance is automatically updated with this information. When a client creates a connection using a connection factory instance, one of these servers becomes the client's connection host, and the connection host stays the same for the life of the connection.

There are three connection factory settings that tune distributed and imported destination producer load balancing behavior:

- `Server Affinity`— Specifies whether send requests load balance only among members local to the connection host when present, instead of all distributed or imported destination members. Defaults to enabled.
- `Load Balancing`—Specifies whether or not non-anonymous producers are load-balanced among members on a per-call basis. If enabled, messages are load-balanced across all members on every `send()` or `publish()`. If disabled, a load-balance decision is made on the `first send()` or `publish()` and all subsequent messages are sent to the same destination. Defaults to enabled.
- `Producer Load Balancing Policy`- Specifies whether messages should be distributed among all running members `Per-Member` (the default), or only to a single member per JVM `Per-JVM`. See Load-Balancing Heuristics in *Administering JMS Resources*.

> **Note**
>
> `Server Affinity`, `Load Balancing` and `Producer Load Balancing Policy` are ignored when:
>
> - producing to regular destinations.
> - used with Unit-of-Order and Unit-of-Work messages to honor a strict ordering quality of service.

3

Monitoring and Managing SAF Agents

You can use the WebLogic Remote Console to monitor runtime statistics and manage message operations for your SAF agents.

Monitoring SAF Agents

You can monitor statistics on active SAF agents defined in your domain using the WebLogic Remote Console. As an alternative, you can also do this programmatically using the `SAFAgentRuntimeMBean`.

The WebLogic Remote Console provides monitoring capabilities for SAF agents, remote endpoints, and connections. The following information can be viewed on each SAF agent:

- For WSRM only, all of the SAF receiving agents that a SAF agent has talked to.
- For WSRM only, all of the conversations that have occurred.
- For WSRM and JMS, all of the endpoints that an SAF agent has sent messages to.

The following statistics information will be recorded on each sending agent:

- Current, total, and high count of messages (per agent and per remote endpoint)
- Total number of messages that has failed (per conversation, agent and remote endpoint)
- Current, total, and high count of conversations (per agent and per remote endpoint)
- Total uptime, downtime, and last time connected or disconnected for a remote agent.

See Monitor an SAF Agent in the *Oracle WebLogic Remote Console Online Help*.

Managing Message Operations on SAF Agents

The WebLogic Remote Console also provides runtime message management capabilities for SAF agents.

For information, see Monitor an SAF Agent in the *Oracle WebLogic Remote Console Online Help*.

4

Troubleshooting WebLogic SAF

You can create, collect, analyze, archive, and access diagnostic data generated by a running server and the applications deployed within its containers. This data provides insight into the runtime performance of servers and applications and enables you to isolate and diagnose faults when they occur. WebLogic SAF takes advantage of this service to provide enhanced runtime statistics, notifications sent to imported SAF queues and topics, message life cycle logging JMS messages, and debugging to help you keep your WebLogic domain running smoothly.
See Monitoring SAF Agents.

Debugging WebLogic SAF

WebLogic Server provides special-purpose debugging features to help you isolate WebLogic SAF performance issues, such as a set of registered WebLogic SAF debugging scopes, and the ability to trace the flow of an individual application request through the SAF service. The topics that follow describe these features, and also explain the different ways you can enable debugging.

Enabling Debugging

You can enable debugging by setting the appropriate `ServerDebug` configuration attribute to `true`. Optionally, you can also set the server `StdoutSeverity` to `Debug`.

You can modify the configuration attribute in any of the following ways.

Enable Debugging Using the Command Line

Set the appropriate properties on the command line. For example,

```
-Dweblogic.debug.DebugSAFSendingAgent=true
-Dweblogic.log.StdoutSeverity="Debug"
```

This method is static and can only be used at server startup.

Enable Debugging Using the WebLogic Remote Console

Use the WebLogic Remote Console to set the debugging values:

1. In the **Edit Tree**, go to **Environment**: **Servers**.
2. Select the server for which you want to enable or disable debugging.
3. Then, go to the **Debug**: **Messaging** page.
4. Select the check box for the debug scope or attributes you want to modify.
5. Click **Save**.
6. Open the Shopping Cart and then click **Commit Changes**.

   Not all changes take effect immediately—some require a restart.

   This method is dynamic and can be used to enable debugging while the server is running.

Enable Debugging Using the WebLogic Scripting Tool

Use the WebLogic Scripting Tool (WLST) to set the debugging values. For example, the following command runs a program for setting debugging values called `debug.py`:

```
java weblogic.WLST debug.py
```

The main scope, `weblogic`, does not appear in the graphic; `saf` is a sub-scope within `weblogic`. Note that the fully-qualified `DebugScope` for `DebugSAFSendingAgent` is `messaging.saf.admin`.

The debug.py program contains the following code:

```
user='user1'
password='password'
url='t3://localhost:7001'
connect(user, password, url)
edit()
cd('Servers/myserver/ServerDebug/myserver')
startEdit()
set('DebugSAFSendingAgent','true')
save()
activate()
```

Note that you can also use WLST from Java. The following example shows a Java file used to set debugging values:

```
import weblogic.management.scripting.utils.WLSTInterpreter;
import java.io.*;
import weblogic.jndi.Environment;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;

public class test {
        public static void main(String args[]) {
       try {
              WLSTInterpreter interpreter = null;
              String user="user1";
              String pass="pw12ab";
              String url ="t3://localhost:7001";
              Environment env = new Environment();
              env.setProviderUrl(url);
              env.setSecurityPrincipal(user);
              env.setSecurityCredentials(pass);
              Context ctx = env.getInitialContext();

              interpreter = new WLSTInterpreter();
              interpreter.exec
                     ("connect('"+user+"','"+pass+"','"+url+"')");
              interpreter.exec("edit()");
              interpreter.exec("startEdit()");
              interpreter.exec
                     ("cd('Servers/myserver/ServerDebug/myserver')");
              interpreter.exec("set('DebugSAFSendingAgent','true')");
              interpreter.exec("save()");
              interpreter.exec("activate()");

       } catch (Exception e) {
       System.out.println("Exception "+e);
       }
```

```
        }
}
```

Using the WLST is a dynamic method and can be used to enable debugging while the server is running.

Changes to the config.xml File

Changes in debugging characteristics, through console, or WLST, or command line are persisted in the `config.xml` file.

This sample `config.xml` fragment shows a transaction debug scope (set of debug attributes) and a single SAF attribute.

Example 4-1 Example Debugging Stanza for SAF

```
<server>
<name>myserver</name>
<server-debug>
<debug-scope>
<name>weblogic.transaction</name>
<enabled>true</enabled>
</debug-scope>
<debug-saf-sending-agent>true</debug-saf-sendingagent>
</server-debug>
</server>
```

SAF Debugging Scopes

The following are registered debugging scopes for WebLogic SAF.

- `DebugSAFStore` (scope `weblogic.messaging.saf.store`) – prints limited information about SAF's use of the store.
- `DebugSAFReceivingAgent` (scope `weblogic.messaging.saf.receivingagent`) – prints information about the SAF receiving side.
- `DebugSAFSendingAgent` (scope `weblogic.messaging.saf.sendingagent`) – prints information about the SAF sending side.
- `DebugSAFVerbose` (scope `weblogic.messaging.saf.verbose`) – prints detailed (internal) information.
- `DebugSAFManager` (scope `weblogic.messaging.saf.manager`) – prints information about SAF management (setting up conversations between agents).
- `DebugSAFAdmin` (scope `weblogic.messaging.saf.admin`) – prints information about SAF administration (pause/resume).

> **Note**
>
> For debugging JMS SAF destinations, you can use the DebugJMSSAF (scope `weblogic.jms.saf`) WebLogic JMS scope, which prints information about JMS SAF (store-and-forward) destinations.

For information about WebLogic JMS debugging scopes, see Troubleshooting WebLogic JMS in *Administering JMS Resources for Oracle WebLogic Server*.

Request Dyeing

Another option for debugging is to trace the flow of an individual (typically "dyed") application request through the SAF service. See Configuring the Dye Vector via the DyeInjection Monitor in *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

SAF Message Life Cycle Logging for JMS Messages

The message life cycle is an external view of the events that a JMS message traverses once it has been stored by a SAF agent and eventually forwarded to a remote JMS destination. Message life cycle logging provides you with easy access to information about the existence and status of stored and forwarded JMS messages within the context of a given SAF agent. In particular, each message log contains information about basic life cycle events such as message storing, forwarding, and expiration. See Understanding WebLogic Logging Services in *Configuring Log Files and Filtering Log Messages for Oracle WebLogic Server*.
SAF message logging is enabled by default when you create a SAF agent; however, you must specifically enable it on the imported SAF destinations in JMS modules targeted to the SAF agent. Logging can occur on a continuous basis and over a long period of time. It can be also be used in realtime mode while the SAF agent is deployed, or in an offline fashion when the SAF agent is down.

For information about viewing and configuring message logging, see Log Messages in the *Oracle WebLogic Remote Console Online Help*.

Events in the SAF Message Life Cycle

When SAF message life cycle logging is enabled for an imported SAF queue or topic, a record is added to the SAF agent's message log file each time a message meets the conditions that correspond to a basic message life cycle event. The life cycle events that trigger a SAF message log entry are as follows:

- Stored – This event is logged when a producer sends a message to an imported SAF queue or topic.
- Forwarded – This event is logged when SAF forwards a message to a remote queue or topic.
- Removed – This event is logged when messages are manually purged from a remote endpoint via a SAF agent.
- Expired – This event is logged when a message reaches the expiration time specified on the imported SAF destination.

Message Log Location

The message log is stored under your domain directory. For example:

```
USER_DOMAIN\servers\servername\logs\safagents\saf_agent_name\jms\jms.messages.log
```

In the preceding directory path, *USER_DOMAIN* is the root directory of your domain, typically `c:\Oracle\Middleware\user_projects\domains\` *USER_DOMAIN*, which is parallel to the directory in which WebLogic Server program files are stored, typically `c:\Oracle\Middleware\wlserver_12.1`.

Enabling SAF Message Logging

You can enable or disable SAF message logging for an SAF queue and SAF topic using the WebLogic Remote Console.

WebLogic Java Management Extensions (JMX) enable you to access the SAFAgentRuntimeMBean and SAFAgentRuntimeMBean MBeans to manage SAF message logs. See Overview of WebLogic Server Subsystem MBeans in *Developing Custom Management Utilities Using JMX for Oracle WebLogic Server*.

When you enable SAF message logging, you can specify whether the log entry will include all the message header fields or a subset of them; all system-defined message properties or a subset of them; all user-defined properties or a subset of them. You may also choose to include or to exclude the body of the message. See MessageProducer and MessageConsumer in *Developing JMS Applications for Oracle WebLogic Server*.

SAF Message Log Content

Each record added to the log includes basic information such as the message ID and correlation ID for the subject message. You can also configure the SAF agent to include additional information such as the message type and user properties.

SAF Message Log Record Format

Except where noted, all records added to the SAF Message Life Cycle Log for JMS messages contain the following pieces of information in the order in which they are listed:

- Date – The date and time the message log record is generated.
- Transaction identifier – The transaction identifier for the transaction with which the message is associated
- WLS diagnostic context – A unique identifier for a request or unit of work flowing through the system. It is included in the SAF message log to provide a correlation between events belonging to the same request.
- Raw millisecond value for "Date" – To aid in troubleshooting high-traffic applications, the date and time the message log record is generated is displayed in milliseconds.
- Raw nanosecond value for "Date" – To aid in troubleshooting high-traffic applications, the date and time the message log record is generated is displayed in nanoseconds.
- JMS message ID – The unique identifier assigned to the message.
- JMS correlation ID – A user-defined identifier for the message, often used to correlate messages about the same subject.
- JMS destination name – The fully-qualified name of the destination server for the message.
- SAF message life cycle event name – The name of the message life cycle event that triggered the log entry.
- JMS user name – The name of the user who (produced? consumed? received?) the message.
- JMS message forwarded identifier – This information is included in the log only when the message life cycle event being logged is the "Message Forwarded" event. If the message forwarded was on a SAF queue, the log will include information about the origin of the remote destination and the OAM identifier for the remote destination known to the SAF

agent. If the remote destination is a durable subscriber, the log will also include the client ID for the connection and the subscription name.

The syntax for the message forwarded identifier is as follows:

```
MC:CA(…):OAMI(wls_server_name.jms.connection#.session#.consumer#)
```

where:

- MC stands for message consumer.
- CA stands for client address.
- OAMI stands for OA&M identifier.
- CC stands for connection consumer, when applicable.

If the consumer is a durable subscriber the additional information will be shown using the following syntax:

```
DS:client_id.subscription_name[message consumer identifier]
```

where DS stands for durable subscriber.

- JMS message content – This field can be customized on a per destination basis. However, the message body will not be available.
- JMS message selector – This information is included in the log only when the message life cycle event being logged is the "Message Forwarded" event. The log will show the "Selector" argument from the JMS API.

Sample Log File Records

The sample log file records that follow show the type of information that is provided in the log file for each of the message life cycle events. Each record is a fixed length, but the information included will vary depending upon relevance to the event and on whether a valid value exists for each field in the record. The log file records use the following syntax:

```
####<date_and_time_stamp> <transaction_id> <WLS_diagnostic_context> <date_in_
milliseconds> <date_in_nanoseconds> <JMS_message_id> <JMS_correlation_id> <JMS_
destination_name> <life_cycle_event_name> <JMS_user_name> <consumer_identifier>
 <JMS_message_content> <JMS_message_selector>
```

> **Note**
>
> If you choose to include the JMS message content in the log file, note that any occurrences of the left-pointing angle bracket (<) and the right-pointing angle bracket (>) within the contents of the message will be escaped. In place of a left-pointing angle bracket you will see the string "<" and in place of the right-pointing angle bracket you will see ">" in the log file.

Message Stored Event

```
####<Nov 3, 2005 11:11:02 AM EST> <> <> <1131034262637> <391826>
 <ID:<864239.1131034262172.0>> <>
 <SenderSafmodule!safDestinations2!xsafQ2@SendingAgent@D1S1> <Stored> <system>
<> <&lt;?xml version="1.0" encoding="UTF-8"?&gt;

&lt;mes:WLJMSMessage xmlns:mes="http://www.oracle.com/WLS/JMS/
Message"&gt;&lt;mes:Header&gt;&lt;mes:JM
```

```
SDeliveryMode&gt;NON_
PERSISTENT&lt;/mes:JMSDeliveryMode&gt;&lt;mes:JMSExpiration&gt;0&lt;/mes:JMSExpi
ration&gt;&lt;mes:JMSPriority&gt;5&lt;/mes:JMSPriority&gt;&lt;mes:JMSRedelivered
&gt;false&lt;/mes:JMSRedelivered&gt;&lt;mes:JMSTimestamp&gt;1131034262172&lt;/me
s:JMSTimestamp&gt;&lt;mes:Properties/&gt;&lt;/mes:Header&gt;&lt;mes:Body&gt;&lt;
mes:Text&gt;EndPointExpireAllWithMessageForward(NP):0&lt;/mes:Text&gt;&lt;/mes:B
ody&gt;&lt;/mes:WLJMSMessage&gt;> <>
```

Message Forwarded Event

```
####<Nov 3, 2005 11:11:30 AM EST> <> <> <1131034290391> <277193>
 <ID:<864239.1131034288312.0>> <>
 <SenderSafmodule!safDestinations3!safErrorQueue@SendingAgent@D1S1>
<Forwarded> <<WLS Kernel>> <> <&lt;?xml version="1.0" encoding="UTF-8"?&gt;

&lt;mes:WLJMSMessage
xmlns:mes="http://www.oracle.com/WLS/JMS/Message"&gt;&lt;mes:Header&gt;&lt;mes:
JMSDeliveryMode&gt;NON_
PERSISTENT&lt;/mes:JMSDeliveryMode&gt;&lt;mes:JMSExpiration&gt;0&lt;/mes:JMSExpi
ration&gt;&lt;mes:JMSPriority&gt;5&lt;/mes:JMSPriority&gt;&lt;mes:JMSRedelivered
&gt;false&lt;/mes:JMSRedelivered&gt;&lt;mes:JMSTimestamp&gt;1131034288312&lt;/me
s:JMSTimestamp&gt;&lt;mes:Properties/&gt;&lt;/mes:Header&gt;&lt;mes:Body&gt;&lt;
mes:Text&gt;EndPointExpireAllWithMessageRedirect(NP):0&lt;/mes:Text&gt;&lt;/mes:
Body&gt;&lt;/mes:WLJMSMessage&gt;> <>
```

Message Expired Event

```
####<Nov 3, 2005 1:04:25 PM EST> <> <> <1131041065929> <42424>
 <ID:<130865.1131041065828.0>> <>
 <udd-saf!safDestinations!safRemoteQueue@SendingAgent@D1C1S1> <Expired>
<<WLS Kernel>> <> <&lt;?xml version="1.0" encoding="UTF-8"?&gt;

&lt;mes:WLJMSMessage xmlns:mes="http://www.oracle.com/WLS/JMS/
Message"&gt;&lt;mes:Header&gt;&lt;mes:
JMSDeliveryMode&gt;PERSISTENT&lt;/mes:JMSDeliveryMode&gt;&lt;mes:JMSExpiratio
n&gt;1131041065848&lt;/mes:JMSExpiration&gt;&lt;mes:JMSPriority&gt;7&lt;/mes:JM
SPriority&gt;&lt;mes:JMSRedelivered&gt;false&lt;/mes:JMSRedelivered&gt;&lt;mes:J
MSTimestamp&gt;1131041065828&lt;/mes:JMSTimestamp&gt;&lt;mes:Properties/&gt;
&lt;/mes:Header&gt;&lt;mes:Body&gt;&lt;mes:Text&gt;testSAFLogging_
Expired&lt;/mes:Text&gt;&lt;/mes:Body&gt;&lt;/mes:WLJMSMessage&gt;> <>
```

Message Removed Event

```
####<Nov 3, 2005 11:11:08 AM EST> <> <> <1131034268206> <803337>
 <ID:<864239.1131034262172.0>> <>
 <SenderSafmodule!safDestinations2!xsafQ2@SendingAgent@D1S1> <Removed> <<WLS
 Kernel>> <> <&lt;?xml version="1.0" encoding="UTF-8"?&gt;

&lt;mes:WLJMSMessage xmlns:mes="http://www.oracle.com/WLS/JMS/
Message"&gt;&lt;mes:Header&gt;&lt;mes:JMS
DeliveryMode&gt;NON_
PERSISTENT&lt;/mes:JMSDeliveryMode&gt;&lt;mes:JMSExpiration&gt;0&lt;/mes:JMSExpi
ration&gt;&lt;mes:JMSPriority&gt;5&lt;/mes:JMSPriority&gt;&lt;mes:JMSRedelivered
&gt;false&lt;/mes:JMSRedelivered&gt;&lt;mes:JMSTimestamp&gt;1131034262172&lt;/me
s:JMSTimestamp&gt;&lt;mes:Properties/&gt;&lt;/mes:Header&gt;&lt;mes:Body&gt;&lt;
mes:Text&gt;EndPointExpireAllWithMessageForward(NP):0&lt;/mes:Text&gt;&lt;
/mes:Body&gt;&lt;/mes:WLJMSMessage&gt;> <>
```

Managing SAF Agent Log Files

After you create a SAF agent, you can configure criteria for moving (rotating) old log messages to a separate file. You can also change the default name of the log file.

Rotating Message Log Files

You can choose to rotate old log messages to a new file based on a specific file size or at specified intervals of time. Alternately, you can choose not to rotate old log messages; in this case, all messages will accumulate in a single file and you will have to erase the contents of the file when it becomes too large.

If you choose to rotate old messages whenever the log file reaches a particular size you must specify a minimum file size. After the log file reaches the specified minimum size, the next time the server checks the file size it will rename the current log file and create a new one for storing subsequent messages.

If you choose to rotate old messages at a regular interval, you must specify the time at which the first new message log file is to be created, and then specify the time interval that should pass before that file is renamed and replaced.

See Rotate Log Files in the *Oracle WebLogic Remote Console Online Help*.

Renaming Message Log Files

Rotated log files are numbered in order of creation. For example, the seventh rotated file would be named myserver.log00007. For troubleshooting purposes, it may be useful to change the name of the log file or to include the time and date when the log file is rotated. To do this, you add `java.text.SimpleDateFormat` variables to the file name. Surround each variable with percentage (%) characters. If you specify a relative pathname when you change the name of the log file, it is interpreted as relative to the server's root directory.

Limiting the Number of Retained Message Log Files

If you choose to rotate old message log files based on either file size or time interval, you may also wish to limit the number of log files this SAF agent creates for storing old messages. After the server reaches this limit, it deletes the oldest log file and creates a new log file with the latest suffix. If you do not enable this option, the server will create new files indefinitely and you will have to manually clean up these files.

Frequently Asked Questions About JMS SAF

Get help with WebLogic SAF by reviewing the set of frequently asked questions.

This section answers commonly asked questions about how JMS SAF operates in a WebLogic domain.

Q. Which sending agent is picked by SAF for a JMS producer?

A. WebLogic Sever's cluster load balancing is used to pick a sending agent for a given JMS producer. Once a SAF agent is picked, it is used for the life of the JMS producer.

Q. How do JMS clients find a SAF destination?

A. A SAF destination can be found the same way a non-SAF JMS destination is found.

- JMS clients can look up a SAF Destination in JNDI
- `createDestination` API

  JMS clients can also use the `createDestination` API to find JMS Destination. JMS clients have to use the SAF destination name in a JMS module. The `Name` must be a fully-qualified name delimited by exclamation points (!). For example:

  ```
  <EAR Name>!<JMS Module Name>!<ImportedDestinationsName>!<SAFQueue or SAFTopic Name>
  ```

Q. Can a JMS producer sending messages to a JMS SAF imported destination be associated with a JMS Unit-of-Order?

A. Yes. For information about the Unit-of-Order feature, See Using Message Unit-of-Order in *Developing JMS Applications for Oracle WebLogic Server*.

Q. Why does my JMS producer associated with a JMS Unit-of-Order fail to send messages if the sending-side is a cluster?

A. In order to use JMS Unit-of-Order with SAF, you must configure the Path Service for the sending-side cluster. See Using the WebLogic Path Service in *Administering JMS Resources for Oracle WebLogic Server*.

Q. Do different JMS producers in the same Unit-of-Order pick up the same Sending Agent?

A. Yes. JMS SAF uses the Path Service to route to the same Sending Agent.

Q. Can a consumer be attached to a JMS SAF Imported Destination?

A. No. JMS consumers can only be attached to actual JMS destinations.

Q. Can a distributed destination be imported?

A. Yes, it can be imported using its JNDI name.

Q. Where do I configure Server Affinity, Load Balancing Enabled, and Forward Delay for a distributed destination that is imported in a sending cluster?

A. Server Affinity and Load Balancing Enabled are configured in the JMS connection factory on which the JMS producer was created. A JMS connection factory creates a JMS connection; a JMS connection creates a JMS session; a JMS session creates a JMS producer. Forward Delay is configured on the JMS distributed destination.

Q. Are the Server Affinity and Load Balancing parameters configured on a JMS connection factory in the sending cluster or server honored on the receiving cluster or server where the JMS Destination resides?

A. Yes. These attributes on the sending cluster or server are honored on the receiving cluster or server. For information about Server Affinity and Load Balancing for distributed destinations, see Configuring Distributed Destination Resources in *Administering JMS Resources for Oracle WebLogic Server*.

Q. Do XA transactions on the sending-side of a cluster ever cross the JMS SAF boundary? In other words, can the receiving-side participate with a transaction from the sending- side?

A. No. Messages are not forwarded until the transaction is committed.

Q. Does JMS SAF preserve the order of messages sent in a JMS session from a sending-side to a JMS destination?

A. Yes.

Q. In the SAF Remote Context, should I configure a Principal Name or a Username/Password?

A. You can configure the Remote SAF Context anyway you want. Username and Passwords are stored in the JMS module, and the principal name is stored in a Credential Mapper configured in the sending-side domain.

Q. The membership of my UDD changed and now my SAF does not seem to work properly. What's happening?

A. Make sure all the domains your process communicates with are configured symmetrically with regards to Cross Domain Security. If one of the domains has Cross Domain Security configured, then all must be configured with Cross Domain Security (or be on the appropriate exception lists). See Configuring Secure Inter-Domain and Intra-Domain Transaction Communication in *Developing JTA Applications for Oracle WebLogic Server*.