Oracle Service Bus for financial services Solutions Guide

Version: 3.0
Document Date: October 2008

Overview

Oracle Service Bus for financial services provides integration with the Society for Worldwide Interbank Financial Telecommunication (SWIFT) network in a Service Oriented Architecture (SOA) environment. You can use Oracle Service Bus for financial services for all SWIFT integration, including FIN MT and MX messages, FileAct and InterAct protocols, and connectivity to SWIFTAlliance Gateway (SAG) and SWIFTAlliance Access (SAA). For additional details, refer to the product details page.

Oracle Service Bus for financial services has received SWIFTReady Financial EAI GOLD Certification for 2008.

This chapter includes the following topics:

What is Oracle Service Bus for financial services

Oracle Service Bus for financial services provides end-to-end integration product for financial institutions. For a complete list of all supported message types refer to the Supported Messages Formats section.

The following are the key features provided by Oracle Service Bus for financial services:

1. Message Processing Engine – The engine supports parsing, validation, and transformation of Financial Traffic (FIN) MT and MX messages. Users can customize message formats to enforce market practices and Straight Through Processing (STP) guidelines, and define additional message formats. You can also process and generate batches of mixed MT and MX messages.
2. Connectivity to SAG – Reliable, fast, and secure connectivity occurs through the MQ Host Adapter (MQHA) and SAA through the MQSeries Host Adapter on SAG (MQSA) and Automated File Transfer (AFT). You can use the ALSB MQ (ALSB MQ) transport, but other protocols like Web Services Host (WSHA) and File Transfer Agent (FTA) are not currently supported.
3. Transition of SWIFT Applications to an SOA Environment –Transformation capabilities for Message Type (MT) to XML and XML to MT provide an easy transition to SOA, in addition to ALSB's traditional SOA capabilities.

Oracle Service Bus for financial services comprises of two products that work together to provide the above described functionality:

Financial Message Designer 3.5

AquaLogic Service Bus 3.0 (ALSB)

Note: For more information on MQHA, MQSA, and AFT, see the SAG and SAA documentation.

Supported Message Formats

Oracle Service Bus for financial services supports the following message formats:

SWIFT MT1xx-MT9xx,

All 4 ISO20022/MX categories - Payments, Trade, Securities, Forex

SEPA (Core and AOS validations)

All FIX messages up to V4.4 and FIXML

FpML V4.2

ISO8583

EDI Payment messages – Paymul, Control, Bansta

BAI2

This section contains the following topics:

Financial Message Designer 3.5

Financial Message Designer is a standalone Swing application that lets you define data formats, validation rules, and mapping information in a cartridge.

After you define a cartridge, you can export it as a JAR file. The JAR file contains classes that are necessary for run-time execution; you can use the classes directly from an ALSB Java Callout.

Note: For more information, see the *Financial Message Designer User's Guide*.

AquaLogic Service Bus Run-Time

ALSB 3.0 is the run-time container for message parsing, validation, and transformation. Messages are received through a proxy service and processed inside the proxy pipeline. The message is typically identified, parsed, and validated before it is transformed and sent through a Business Service.

Before the message can be processed, the cartridge JAR generated by the Financial Message Designer component must be registered as a JAR resource in ALSB. ALSB also provides a facade JAR to simplify invoking common cartridge operations, and provide additional SWIFT-specific capabilities.

For example, SWIFT messages are often sent and received in batches. The facade JAR provides an API to split batched messages and inversely aggregate messages into a batch.

Note: For more details about the facade API, refer to the Javadoc.

Integration with SAG and SAA

ALSB exchanges messages with the SWIFTAlliance Access using File Transport for the AFT protocol and MQSeries Transport for MQSA protocol.

ALSB exchanges messages with the SWIFTAlliance Gateway using MQSeries Transport for MQHA protocol. Other options to connect to SAG are available through web services (WSHA and FTA), but are not supported.

Oracle Service Bus for financial services provides full support for parsing and identifying SWIFT status messages, ACK, and NAK**.** However, reconciling messages is better performed in a stateful process engine, such as Oracle Business Process Management (OBPM) or Oracle WebLogic Integration (WLI).

Note: For more information on the MQSeries Transport, MQHA, and MQSA, see the ALSB documentation and SWIFT documentation.

Compiling and Loading Cartridges

Cartridges contain flows that are required to process SWIFT messages. Use Financial Message Designer to create these flows, to process all SWIFT MT and MX messages, and return the XML representation of the message in the form to the caller.

The following general steps describe how to create a cartridge and load it into ALSB:

1. Create a cartridge with the message formats, validation rules, and transformations you want to use in ALSB. See Compiling Cartridges for instructions.
2. Compile the cartridge into a JAR file that can be consumed by ALSB.

The *Financial Message Designer User's Guide* provides instructions for creating new message formats, customizing existing formats, modifying validation rules, and defining mappings between message types.

This section contains the following topics:

Compiling Cartridges

To build a cartridge:

1. In the Financial Message Designer, choose **Build** > **Code Generation Settings.** For more information, see the *Financial Message Designer User's Guide* on the web site. The guide is also available at `<Designer Install Directory>\docs\index.pdf`.
2. Select the **Target Platform** tab and click **Generate Jar for Service Bus**.
3. Click **OK**.
4. Choose **Build** > **Generate Cartridge**.

The splitter window at the bottom shows the build progress. If the build is successful, note the location of the JAR file that you can import into ALSB.

Loading Cartridges

To load the cartridge into ALSB:

1. To use the classes generated by the Financial Message Designer component, you must import them into ALSB. Create a JAR resource and point to the cartridge JAR file.
2. Invoke the cartridge JAR file directly using a Java Callout. The cartridge public API documentation is located in the following directory: `<Designer Install directory>\docs\java\index.pdf`.

Invoking the cartridges directly in the API can be cumbersome and typically requires writing additional Java code. A better solution is to use the financial facade.

Importing the Financial Façade JAR File

Oracle Service Bus for financial services offers an API facade that simplifies how the cartridges are invoked and provides the following functionality for SWIFT messages:

Parses, validates, and transforms APIs

Invokes from a Java Callout with no need to write Java code

Identifies SWIFT message types

Provides batching and splitting capabilities

Provides optimized SWIFT processing

Before you can invoke the cartridge, you must load the `financial.jar` facade and register it as a JAR resource. The façade JAR file is located in the following directory:

```
<Bea Home>/weblogic92/servicebus/financial/financial.jar
```

Two schema files are also located in this folder. The files document the XML return type for some of the façade APIs:

```
<ALSB_HOME>/financial/financial.xsd
<ALSB_HOME>/financial/swift.xsd
```

For more information about the façade APIs, see the Javadoc.

To load the façade:

1. In the ALSB Console, create a new JAR resource.
2. Locate the `<ALSB_HOME>/financial/financial.jar` file.
3. Click **Save**.

After the resource is created, the following Informational Conflict message appears:

```
java.lang.NoClassDefFoundError:com/tplus/transform/runtime/DataObject
```

This conflict is harmless and is for informational purposes only. The message appears because the façade refers to the `DataObject` class that is present only in Financial Message Designer cartridge JAR files. Once you set a dependency between the façade and a cartridge JAR file, the conflict disappears.

Loading and Linking the Cartridges JAR File

To load a cartridge JAR file, follow the same steps you did to load the façade:

1. In the ALSB Console, create a new JAR resource.
2. Locate the `cartridge.jar` file.
3. Click **Save**.

After you load the cartridge, edit the façade resource and add a dependency to the imported cartridge JAR.

To edit the facade:

1. In the ALSB Console, click the façade resource.
2. Select the **Dependencies** tab and click **Update**.
3. Click **Add JARS**. A pop-up browser opens.
4. Select the cartridge JAR check box you want to use and click **Submit**.
5. Click **Save**.

The façade now points to the cartridge and can be used from a Proxy service.

Note: A façade can be associated with only one cartridge. If you have two cartridges, clone the façade and point the cloned façade to the second cartridge.

See the *Financial Message Designer Installation Guide* for more details.

Using Cartridges in ALSB

You can use cartridges from a proxy pipeline directly through a Java Callout action or through invoking the façade. Typically, operations available on the cartridge include Parse, Transform, Validate, and IdentifySwiftMessageType. The public classes are under the `com.bea.alsb.financial.api` package. See the Javadoc for more details about the API.

Some methods return a Java object. For example:

```
<con:java-content ref="jcid:-4af11bc6:1153ef98818:-7fef"xmlns:con=
"http://www.bea.com/wli/sb/context"/>
```

Those objects can be consumed only by other Java Callout actions. For more details about Java objects in the pipeline, see the *AquaLogic Service Bus User Guide*.

Other APIs return *typed XML*. Typed XML is treated the same as XML in the pipeline. The schemas are located in the `<Bea Home>/weblogic92/servicebus/financial/` directory.

Note: The façade does not support message types created in Financial Message Designer with the **Mode of Operation:** set to **Batch Mode**. You should use the split/aggregate methods provided by the façade or use the cartridge Runtime APIs directly if you still want to define your message in Batch Mode.

This section contains the following topics:

Handling Errors and Exceptions

The APIs give you the flexibility to choose the appropriate behavior for your application. When parsing and validating a message, two situations can occur:

1. Errors can accumulate – When you allow errors to accumulate, you should verify if there were errors after the parsing is completed. Generally, letting errors accumulate is the best method to use.
2. When the first error is encountered, an exception is raised in the pipeline – When you raise an exception, the biggest disadvantage is you see only the first error detected.

Errors are returned as an XML document and you can wrap them into a Service Oriented Architecture Protocol (SOAP) fault.

Modifying Financial Messages

Messages in XML format can be modified using regular ALSB XQuery features. For example, you can modify a value or add a new element.

MT messages can be more challenging. You can use any of the following approaches:

1. Define a static mapping in Financial Message Designer.
2. Map the MT message to XML, modify it, and map it back to MT – This is the recommended approach, because it provides the most flexibility and the impact on performance is negligible.
3. Use the cartridges API directly to modify the message – Write additional Java code to modify message fields. Retrieve the parsed `DataObject` from the `MessageWrapper` using the generic `getMessageDataObject()` API. You can then retrieve the field and set the value. The expected Java class to set the value can be retrieved from the message data format in the Financial Message Designer.

Performing a BIC Lookup

Every SWIFT partner receives a unique Bank Identifier Code (BIC). SWIFT makes the BIC directories available to all partners. Oracle Service Bus for financial services can look up a BIC table in a resource file or database at run time to perform validation or message mapping. For more information, see the *SWIFT Resources Guide*.

Troubleshooting

Errors can occur during SWIFT message parsing and validating because of validation problems or any of the following reasons:

The wrong data format is used to process the message – Ensure that the message headers specified in the Financial Message Designer (when you create the external message) are correct. For example, specify incoming SWIFT Headers for messages coming from SWIFT. This is the most common reason for errors.

Parsing MT messages where the Carriage Return (CR) characters have been removed – SWIFT requires that a Line Feed (LF) character must be preceded by a CR character. However, some editors (especially on UNIX or Apple Macintosh systems) remove the CR character. Therefore, you need to verify that your MT messages have not been corrupted. You can use a File Transport Business Service and a hex editor to check the content of a pipeline

variable. The same issue occurs if you try to embed SWIFT messages within an XML document. The XML specification requires the processor to strip out any CR characters when parsing the XML. To avoid this problem, carry SWIFT messages within XML as base64-encoded binary data.

When you use the split method, special XML characters declared within the ALSB pipeline must be escaped, as shown in the following table.

Special XML Characters

Special XML Characters	Escaped XML Characters
quote (")	"
apostrophe (')	'
ampersand (&)	&
less than (<)	<
greater than (>)	>
Carriage Return (\r)
Line Feed (\n)
Horizontal Tab (\t)	

When you integrate with SAA (notably through MQSA), some special trailer blocks might be appended to the response MT message. The blocks might include the S block, MAC block, and PK block. This integration is configurable from the SAA Administration Console. Those blocks are not currently supported by the Oracle Service Bus for financial services message libraries and errors appear if you try to parse the blocks. You can safely ignore those errors or simply deactivate those options from SAA.

MQSeries Transport uses a local transaction when retrieving messages from the response queue. If pipeline processing fails and the message is put back into the queue, SAA sends another message with similar content. Therefore, you might receive multiple responses to your query, but it is actually the same response.

If a `ClassCastException` error appears after you manually set a message field to a `DataObject`, you are probably not using the right Java type. The expected Java type is defined in the data format of the message, and is visible in the Financial Message Designer. When you do a Java Callout to a method with an Object parameter, the variable passed from the ALSB pipeline to the Java method will usually be converted to String. The Java Callout action does not keep the parity between XQuery types and Java types.