webMethods FIX Module Installation and User's Guide

Version 7.2

December 2014

This document applies to webMethods FIX Module Version 7.2 and to all subsequent releases.

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

Document ID: ESTD-FIX-IUG-72-FCS-20141224

Table of Contents

About this Guide

This guide describes how to install, configure, and use FIX Module 7.2. It contains information for administrators to configure and manage a webMethods system and for application developers who want to create webMethods Integration Server services to exchange FIX messages with trading partners.

To use this guide effectively, you should:

- Have a basic knowledge of the Financial Information eXchange (FIX) message standard, FIX terminology, and the Appia server. For more information, visit http://www.fixtradingcommunity.org/.
- Have installed webMethods Integration Server, Software AG Designer, My webMethods Server, and webMethods Trading Networks. For more information about installing these components, see the webMethods installation guide for your release. See "About this Guide" for specific document titles.
- Be familiar with webMethods Integration Server and Integration Server Administrator.
- Be familiar with webMethods Integration Serverand understand the concepts and procedures described in the various webMethods Trading Networks guides.
- Be familiar with using Software AG Designer for creating processes and tasks and understand the concepts and procedures related to Designer. For more information, see the Designer Process Development online help for your release. See "About this Guide" for specific document titles.

Deprecation of webMethods Developer

webMethods Developer is deprecated and does not support all the features of webMethods Integration Server 8.2. SoftwareAG recommends the use of SoftwareAG Designer for service development.

Document Titles

Some webMethods document titles have changed during product releases. The following table will help you locate the correct document for a release on the Software AG Documentation Web site or the Empower Product Support Web site.

Documentation	7.x	8.0	8.0 SP1	8.2	9.0 and later
Designer Process Development online help					
webMethods BPM Process Development Help				x	x
webMethods Designer BPM Process Development Help		x	x		
webMethods Designer Process Development Help	x				
Designer Service Development online help					
webMethods Service Development Help				x	x
webMethods Designer Service Development Help	x	x	x		
Developer user's guide					
Developing Integration Solutions: webMethods Developer User's Guide			x	x	
webMethods Developer User's Guide	x	x			
Integration Server administration guide					
webMethods Integration Server Administrator's Guide	x	x			x
Administering webMethods Integration Server			x	x	
Integration Server built-in services reference guide					
webMethods Integration Server Built-In Services Reference	x	x	x	x	x
Integration Server clustering guide					

Documentation	7.x	8.0	8.0 SP1	8.2	9.0 and later
webMethods Integration Server Clustering Guide	x	x	x	x	x
Integration Server publish-subscribe developer's guide					
Publish-Subscribe Developer's Guide	x	x	x	x	x
My webMethods administration guide					
Administering My webMethods Server			x	x	x
My webMethods Server Administrator's Guide	x	x			
Optimize administration guide					
Administering webMethods Optimize			x	x	x
webMethods Optimize Administrator's Guide	x	x			
Optimize user's guide					
webMethods Optimize User's Guide	x	x			x
Optimizing BPM and System Resources with BAM: webMethods Optimize User's Guide			x	x	
Process Engine administration guide					
Administering webMethods Process Engine		x	x	x	x
webMethods Process Engine User's Guide	x				
Trading Networks administration guide					
webMethods Trading Networks Administrator's Guide	x	x			x

Documentation	7.x	8.0	8.0 SP1	8.2	9.0 and later
Building B2B Integrations: webMethods Trading Networks Administrator's Guide			x	x	
Trading Networks built-in services reference guide					
webMethods Trading Networks Built-In Services Reference	x	x	x	x	x
Trading Networks concepts guide					
webMethods Trading Networks Administrator's Guide and *webMethods Trading Networks User's Guide*					x
Understanding webMethods B2B: webMethods Trading Networks Concepts Guide			x	x	
webMethods Trading Networks Concepts Guide	x	x			
Trading Networks user's guide					
webMethods Trading Networks User's Guide	x	x			x
Managing B2B Integrations: webMethods Trading Networks User's Guide			x	x	
webMethods installation guide					
Installing webMethods Products and *Using the Software AG Installer*				x	x
Software AG Installation Guide			x		
webMethods Installation Guide	x	x			
webMethods logging guide					
webMethods Audit Logging Guide			x	x	x

Documentation	7.x	8.0	8.0 SP1	8.2	9.0 and later
webMethods Audit Guide	x	x			
webMethods upgrade guide					
Upgrading webMethods Products				x	x
webMethods Upgrade Guide	x	x	x		

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies storage locations for services on webMethods Integration Server, using the convention *folder.subfolder:service* .
UPPERCASE	Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+).
Italic	Identifies variables for which you must supply values specific to your own situation or environment. Identifies new terms the first time they occur in the text.
`Monospace font`	Identifies text you must type or messages displayed by the system.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.

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

Online Information

Software AG Documentation Website

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

Software AG Empower Product Support Website

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

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

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

Software AG TECHcommunity

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

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

1 Concepts

What is FIX?

FIX (Financial Information eXchange protocol) is a messaging standard used to exchange real-time electronic securities transactions between business partners. The protocol is used by stock exchanges, investment banks, mutual funds, and money managers to support a variety of business functions. FIX messages are exchanged between business partners using a FIX engine.

What is a FIX Engine?

A FIX engine is middleware that sits between your enterprise and that of your business partners to facilitate the exchange of messages. The FIX engine manages the network connection, sends and receives FIX messages, and validates that messages conform to the FIX format. There are many FIX engines available; however, FIX Module only supports NYSE Technologies Appia engine. Therefore, only the Appia FIX engine is discussed in this guide.

For more information about the FIX protocol, visit http://www.fixprotocol.org/.

What is webMethods FIX Module?

webMethods FIX Module allows you to exchange messages with your trading partners in standard FIX message format. FIX Module interfaces with the FIX engine, the gateway through which all FIX messages pass.

For incoming FIX messages sent from your partners, FIX Module receives the message from the FIX engine and, using Trading Networks, processes the message according to your configurations. You can customize how a message is processed by assigning a Trading Networks processing rule to perform a specific action on the message, such as routing the message to a system or sending an e-mail alert. Also, you can create a mapping service that defines data transformations to perform, such as allowing an internal system that uses a non-FIX format to use the message data.

FIX Module also processes outbound messages. When FIX Module receives messages from an internal system, it converts the messages into FIX standard message format, and then forwards the messages to the FIX engine, which sends the messages to your business partner.

FIX Module uses IS document types to determine how to process an inbound or outbound message, matching a message to a specific pre-defined format and then applying any configurations assigned to that IS document type. For more information about IS document types, see "IS Document Types" on page 20.

For a list of the FIX message versions that FIX Module supports, see *webMethods eStandards Modules System Requirements*, available in the webMethods area of the Software AG Documentation Web site.

FIX Module Packages

FIX Module contains the following packages that you install on Integration Server:

Package	Description
WmFIX	Contains the services to send and receive FIX messages, including converter services to transform an Integration Server document instance to a raw FIX message and vice versa. This package also contains the components and services necessary to communicate with the FIX engine.
WmFIXMessages	Contains the Integration Server document types corresponding to all message types for the FIX versions that FIX Module supports.
WmFIXSample	Contains sample services that show how to use different features of FIX Module. You can also use the sample services as examples for how to create your own services. To download this package, go to the ESB & Integration forum of the Software AG Developer Community for webMethods at http://communities.softwareag.com/ecosystem/communities/public/Developer/webmethods/products/esb/ and see the Code Samples. **Important:** FIX Module samples only demonstrate the features of the module and must not be used in the production environment. You must delete the WmFIXSample package before you go into production.

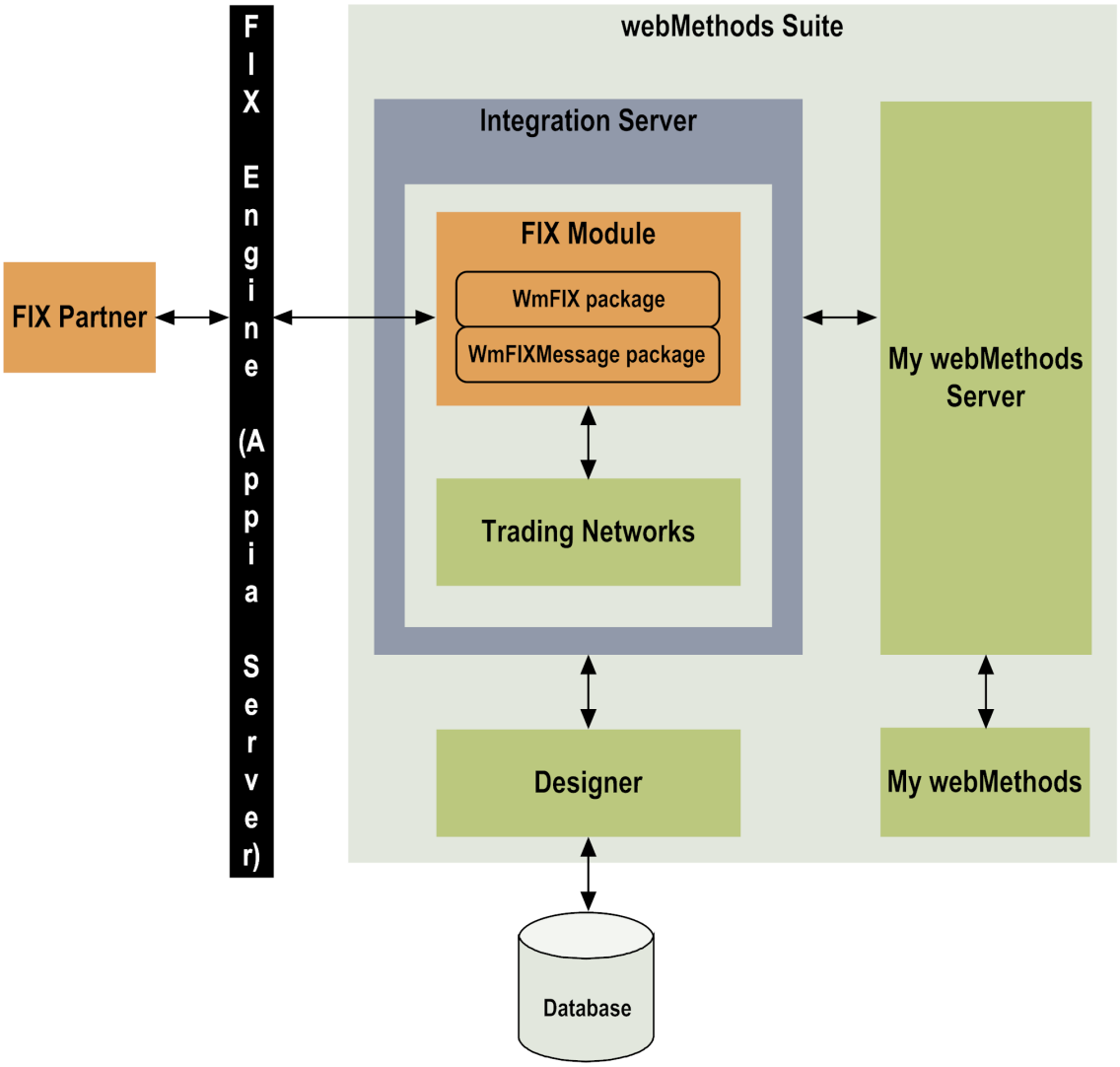
FIX Module Architecture

webMethods FIX Module requires the following components. For further information about these components, refer to their respective guides. See “About this Guide” for specific document titles.

Component	Description
webMethods FIX Module	The webMethods FIX Module and packages.

Component	Description
webMethods Integration Server	The server underlying webMethods components. Use Integration Server Administrator to manage, configure, and administer all aspects of Integration Server, such as users, security, packages, and services. For details, see the Integration Server administration guide for your release. See “About this Guide” for specific document titles.
webMethods Trading Networks	Trading Networks stores inbound and outbound FIX messages. Trading Networks tracks the delivery status for outbound FIX messages and uses Trading Networks document types and processing rules to identify and route incoming messages to other services or systems.
My webMethods Server and My webMethods	My webMethods Server is a run-time container for the functions available in webMethods components, such as, Integration Server, Trading Networks, and FIX Module. The user interface in which you perform these functions is called My webMethods.
FIX engine	Third-party FIX engine, such as the Appia engine, provides the infrastructure so that you can exchange messages with your trading partners.

The following diagram illustrates the high-level architecture of webMethods FIX Module and its components.



FIX Module Features

FIX Module provides the following features:

Feature	Description
FIX Engine Configuration and Administration Using My webMethods	Using My webMethods, you can define the connection details to communicate with the FIX engine and send operator commands.
Integration with Trading Networks	Trading Networks stores all messages that FIX Module sends and receives. Trading Networks also

Feature	Description
	provides message monitoring features for tracking the status of message transmission.
Message Formatting	Creates messages in the format required for the receiving system. FIX Module uses IS document types to transform data generated by your internal (back-end) system into FIX format before sending the message to your partner. FIX Module also transforms messages received from your partners from FIX format into an IS document.
Message Routing and Processing	Messaging features allow you to exchange FIX messages with your partners through the FIX engine. (To send and receive messages, you must have Trading Networks installed.)
Reliable Messaging	FIX Module provides reliable message delivery when sending and receiving messages. FIX Module automatically detects if the connection to the FIX engine is lost and attempts to reestablish the connection until it is restored or the retry limit is met. If there are any unsent messages, the module adds them to the queue and re-sends them as soon as the connection is reestablished.
Integration Server Clustering	FIX Module operates in Integration Server cluster mode to provide high system availability. In cluster mode only one node of the cluster is connected to the FIX engine, but any node in the cluster can send a FIX message (routed through the connected node). The node that is connected to the FIX engine also receives incoming FIX messages and stores them in the Trading Networks database.
Communication with Appia Server	Allows operators to send commands to the Appia server from the FIX Module Administration page.

Message Processing

FIX Module enables you to seamlessly integrate FIX messaging into your system architecture to achieve straight-through processing.

Message Parsing and Formatting

FIX Module processes incoming and outgoing FIX messages. When the module receives incoming FIX messages, it converts them into FIX Integration Server records using the wm.fix.format:convertFIXToIData service. FIX Module creates outbound FIX messages by converting data from Integration Server records to FIX format using the wm.fix.format:convertIDataToFix service.

Message Exchange and Routing

FIX Module integrates with Trading Networks to provide simple and reliable message processing for all messages exchanged with your partners.

- **Inbound Message Routing**. FIX Module receives a new message by registering a listener object with the Appia server. When a new message arrives, FIX Module translates the message into IData format, and then forwards the message to Trading Networks for processing. Trading Networks stores the raw FIX message and the translated IData FIX message in the database (as one record) and uses the profile, document type, and assigned processing rule to process the message accordingly.
- **Outbound Message Processing**. FIX Module uses the wm.fix.tn.trp:send service to process outbound messages. This service executes the following steps:
  - Formats the raw FIX message using the wm.fix.format:convertIDataToFix service.
  - Stores the message in the Trading Networks database.
  - Sends the message to Appia.

Message Storage and Queuing

FIX Module stores all messages exchanged between FIX Module and the FIX engine in the Trading Networks database. Each message is assigned a status that indicates its transmission state:

- **FIX_RECEIVED**—FIX Module received the message from the Appia server.
- **FIX_SENT**—FIX Module successfully sent the message to the Appia server.
- **FIX_QUEUED**—The message could not be sent because the connection to Appia was lost. FIX Module guarantees delivery of messages. If a problem occurs during message transmission, for example, when the connection to Appia is lost, the message is queued in Trading Networks. When the connection is restored, messages are automatically re-sent and updated with a status of `FIX_SENT`.

FIX Module also captures the following Appia events for sent messages:

- **APPIA_MESSAGE_VALIDATED**—Appia validated the message.
- **APPIA_MESSAGE_SENT**—Appia sent the message to your partner.

- **APPIA_MESSAGE_COMMITED**—Appia committed the message to its persistence store.

You can view all messages sent, received, and queued in My webMethods. For more information about viewing messages in Trading Networks, see the Trading Networks user's guide for your release. See "About this Guide" for specific document titles.

IS Document Types

An IS document type defines the structure and type of data in a document (also known as an IData object). In FIX Module, IS document types correspond to each of the FIX message types that FIX Module supports.

You can use an IS document type to build a document or a document list field and define it as the blueprint for pipeline and IData object validation. FIX Module also uses the data stored in IS document types to transform outgoing FIX IS documents into standard raw FIX messages (using the wm.fix.format:convertIDataToFix service). FIX Module provides you with the flexibility to define custom fields, as needed, in the IS document type. For more information about defining custom fields, see "Creating Custom Fields" on page 41. FIX Module also provides an IS document type for processing unidentified fields.

IS Document Type Package Location

You can use Designer to view the IS document types for each FIX message type. IS document types are located in the wm.fix.rec.*FIX** package as part of the WmFIXMessages, where *FIX** is the FIX message version. For example, the folder FIX40 contains the IS document types that correspond to the FIX message version 4.0.

Each IS document adheres to the naming convention, *FIX <FIX version><message type>*, where *FIX version* is the version of FIX message and *message type* is the FIX message type. For example, "FIX40Advertisement" is the IS document for FIX version 4.0 messages of message type "advertisement."

IS Document Structure

The FIX message document structure varies according to the FIX message version and the message type. For example, the structure of the message header for version 4.0 is different than for version 5.0. Similarly, the structure of the header and trailer is the same for all messages of the same version, but the application message structure varies according to the message type.

Each IS document contains the following fields as indicated by the corresponding IS document type:

IS Document Field	Description
Header	Contains the FIX message header fields.

IS Document Field	Description
ApplicationMessage	Contains the FIX application fields.
Trailer	Contains the FIX message trailer fields.
unIdentifiedFields	Contains fields that cannot be identified in the incoming message because they are not part of the FIX specification. This field is only populated for incoming messages. It is not used when sending a FIX message.

For more information about IS documents, see the Designer Service Development online help for your release. See “About this Guide” for specific document titles. For more information about FIX message types, see the FIX Protocol Organization Web site.

2 Installing webMethods FIX Module

Overview

This chapter explains how to install, upgrade, and uninstall webMethods FIX Module 7.2. The instructions use the Software AG Installer and the Software AG Uninstaller wizards. For complete information about the wizards or other installation methods, or to install other webMethods products, see the webMethods installation guide for your release. See “About this Guide” for specific document titles.

Requirements

For a list of the operating systems and webMethods products that FIX Module 7.2 supports, see *webMethods eStandards Modules System Requirements*, available in the webMethods area of the Software AG Documentation Web site.

Installing FIX Module

Note: If you are installing FIX Module 7.2 in a clustered environment, you must install it on each Integration Server in the cluster, and each installation must be identical. For more information about working with FIX Module in a clustered environment, see "Administering webMethods FIX Module in a Cluster" on page 64.

To install webMethods FIX Module 7.2

1. Download Software AG Installer from the Empower Product Support Website at https://empower.softwareag.com.
2. If you are installing FIX Module 7.2 on an existing Integration Server installation, shut down Integration Server.
3. Start the Software AG Installer wizard.
   a. In the **Release** list, choose the webMethods release that includes the Integration Server version on which to install the module. For example, if you are installing FIX Module on Integration Server 7.1, select the 7.1 release.
   b. Provide your Software AG Empower user name and password. Installer uses these to connect to the installer server and download the products for which you have licenses.
   c. Specify the installation directory to use (the default is SoftwareAG). If you are installing on an existing Integration Server, specify the webMethods installation directory that contains the host Integration Server. If you are installing both Integration Server and the module, specify the installation directory. Installer installs the module in the *Integration Server_directory* \packages directory.

When installing the module on Integration Server 9.6 and above, you can choose to install the package in the default instance. The package will be installed both in the package repository and the default instance packages directory located in *Integration Server_directory*\instances\default\packages.

You can move the package to an IS Instance of your choice by using the is_instance Script. For more information about using is_instance Script, see *webMethods Integration Server Administrator's Guide*.

d. In the product selection list, choose **eStandards > webMethods FIX Module 7.2** and **My webMethods User Interfaces > webMethods FIX Module 7.2**. You can also choose to install any required products indicated in the *webMethods eStandards Modules System Requirements* .

4. After installation completes, close Installer.
5. Complete the installation as described in Completing the Installation below.

Completing the Installation

This section provides instructions for completing the installation of FIX Module.

To complete the installation

1. Install the FIX engine according to the vendor's instructions.
2. Create the WM_FIX_CONFIGURATION_PARAMS table to store the FIX engine connection information, as follows:

   a. In the *Integration Server_directory* \packages\WmFIX\config folder, use the appropriate database client to run the SQL script script_<*database*>.sql (where *database* is your database vendor, either MS SQL, DB2, or Oracle).

   b. When prompted, connect to the same database instance that you configured when installing Trading Networks.

3. Copy the appia.jar file from the *AppiaServer_homeDirectory* \lib folder to the *Integration Server_directory* \packages\WmFIX\code\jars directory.
4. If you are installing FIX Module on an already installed My webMethods Server, execute the script file located in *Software AG_directory* \MWS\bin by running the following command:

   ```
   mws -s servername update
   ```

   The FIX Module-My webMethods Server user interface is now deployed to your existing My webMethods Server server instance.

5. Start the following servers:

   - FIX engine
   - My webMethods Server
   - The Integration Server on which you installed FIX Module 7.2

Upgrading from FIX ModuleFIX Module 6.5 to FIX ModuleFIX Module 7.2

This section explains how to upgrade FIX Module and migrate packages from FIX Module 6.5 to FIX Module 7.2 for users of an implementation of FIX Module 6.5 using Trading Networks.

Users with an implementation of FIX Module 6.5 without Trading Networks and customized settings must uninstall FIX Module and reinstall it as a new installation by following the procedure described in "Installing webMethods FIX Module " on page 23.

FIX Module 7.2 introduces several changes to the module's design. Therefore, as part of the upgrade procedure, you must complete the configuration sub-steps described in this section to use FIX Module 7.2 successfully.

Before you begin the upgrade process, read the entire procedure to understand changes in the module's design. When you install FIX Module 7.2, all existing IS document type documents will be overwritten. If you have modified any of the IS document types, for example, by adding new fields, in your FIX Module 6.5 WmFIXMessages package, and you want to keep these custom fields, after installing FIX Module 7.2, you must modify the corresponding IS document types installed with the WmFIXMessages 7.2 package. For more information about using custom fields, see "Creating Custom Fields" on page 41.

Step 1: Uninstall FIX Module 6.5 and Install FIX Module 7.2

To upgrade and migrate from FIX Module 6.5 to FIX Module 7.2

1. Back up your existing WmFIX, WmFIXMessages, and WmFIXForTN 6.5 package installation and back up your webMethods databases according to your RDBMS vendor's instructions.
2. Uninstall FIX Module 6.5. For instructions, see *webMethods FIX Module Installation and User's Guide* , version 6.5.
3. Install FIX Module 7.2 as described in "Installing FIX Module " on page 24.

Step 2: Reconfigure Appia Connections

In FIX Module 6.5, connections to the Appia engine were manually configured in the Appia configuration file (appia.cnf). In FIX Module 7.2, connections to the Appia engine are configured instead through My webMethods. Therefore, you must reconfigure your Appia connections. For more information, see "Configuring webMethods FIX Module " on page 31.

Step 3: Reconfigure Outbound Mapping Services

In FIX Module 6.5, to send a message to a trading partner, it was necessary to complete a multi-step configuration process:

- Create an outbound mapping service to map an internally-generated message (from a back-end system) to a *bizdoc* and invoke the fix.tn.doc:addContentPart service.
- Create a processing rule for the IS document type and configure the processing action:
  - Invoking the outbound mapping service in the **Execute Service** option.
  - Using `FIX Transport` as the **Deliver Document By** option.

In FIX Module 7.2, sending a message to a trading partner is much simpler. The fix.tn.doc:addContentPart service has been replaced by the wm.fix.tn.trp:send service, which transforms messages into FIX format and sends them to the FIX engine. Also, FIX Module 7.2 does not require you to define a processing rule. Therefore, you must update your outbound mapping services that invoked fix.tn.doc:addContentPart in FIX Module 6.5 to now invoke wm.fix.tn.trp:send instead.

To update the mapping of outbound services

1. In My webMethods, delete the processing rule that is configured in the processing rule **Action**, *Deliver Document By* option.

   For information about using processing rules, see the Trading Networks administration guide for your release. See "About this Guide" for specific document titles.

2. In Designer, change the mapping of any services that call fix.tn.doc:addContentPart, instead, mapping the IS document directly to the *fixIData* input field of the wm.fix.tn.trp:send service.

   For complete information about mapping services, see the Designer Service Development online help for your release. See "About this Guide" for specific document titles.

Step 4: Change IS Document Field Names

In FIX Module 7.2, some fields in the *ApplicationMessage* part of the IS document type have new names and structures. These changes affect IS document types that correspond to FIX message versions 4.2 and 4.4. Follow the instructions in this section to update the specified field names for version 4.2 IS document types. (IS document types for version 4.4 are updated when you install FIX Module 7.2.)

Updating FIX Message Version 4.2

In FIX Module 7.2, field names have changed in the *ApplicationMessage* part of the FIX42SecurityStatus and the FIX42BidRequest IS document types for version 4.2 FIX messages.

To update field names in IS document types for version 4.2 FIX messages

In Designer, run the migration utility, wm.fix.util:modify42FieldNames.

The utility updates the following fields:

IS Document Type Name	Application Message Field Name Change	
	6.5 Field Name	7.2 Field Name
FIX42SecurityStatus	CorporationCorporateAction	CorporateAction
FIX42BidRequest	OutSideMainCountryIndex	OutMainCntryUIndex

Updating FIX Message Version 4.4

In FIX Module 7.2, field names and the field structure have changed in the *ApplicationMessage* part of some version 4.4 FIX message IS document types. When you install FIX Module 7.2, the current versions of the IS document types are also installed. However, if you have modified any of the affected IS document types in FIX Module 6.5 and want to keep your custom fields, you must manually update the corresponding IS document types installed with FIX Module 7.2.

The following changes have been made to the version 4.4 IS document types:

- In the **FIX44OrderMassCancelReport** IS document type:
  - The document list *NoAffectedOrders* has been renamed as *AffectedOrders* .
  - The new, optional, String type field, *NoAffectedOrders* , has been added above the *AffectedOrders* document list.
- In version 4.4 IS document types, the following changes have been made in the document list *Legs* :
  - The document list *NoLegSecurityAltID* has been renamed as *LegSecurityAltID* .
  - The new, optional String type field, *NoLegSecurityAltID* , has been added above the document list *LegSecurityAltID* .

  The changes to the document list *Legs* apply to all of the following 4.4 version IS document types:

  FIX44Advertisement FIX44News

FIX44AllocationInstruction

FIX44AllocationReport

FIX44AssignmentReport

FIX44CollateralAssignment

FIX44CollateralInquiry

FIX44CollateralInquiryAck

FIX44CollateralReport

FIX44CollateralRequest

FIX44CollateralResponse

FIX44Confirmation

FIX44CrossOrderCancelReplaceRequest

FIX44CrossOrderCancelRequest

FIX44DerivativeSecurityList

FIX44DontKnowTradeDK

FIX44Email

FIX44ExecutionReport

FIX44IOI

FIX44MarketDataIncrementalRefresh

FIX44MarketDataRequest

FIX44MarketDataSnapshotFullRefresh

FIX44MassQuoteAcknowledgement

FIX44MultilegOrderCancelReplace

FIX44PositionMaintenanceReport

FIX44PositionMaintenanceRequest

FIX44PositionReport

FIX44Quote

FIX44QuoteCancel

FIX44QuoteRequest

FIX44QuoteRequestReject

FIX44QuoteResponse

FIX44QuoteStatusReport

FIX44QuoteStatusRequest

FIX44RequestForPositions

FIX44RequestForPositionsAck

FIX44RFQRequest

FIX44SecurityDefinition

FIX44SecurityDefinitionRequest

FIX44SecurityList

FIX44SecurityListRequest

FIX44SecurityStatus

FIX44SecurityStatusRequest

FIX44TradeCaptureReport

FIX44TradeCaptureReportAck

FIX44TradeCaptureReportRequest

FIX44NewOrderCross

FIX44TradeCaptureReportRequestAck

FIX44NewOrderMultileg

Uninstalling FIX Module

This section provides instructions that are specific to uninstalling FIX Module 7.2. For instructions on uninstalling FIX Module version 6.5, see *webMethods FIX Module Installation and User's Guide* version 6.5.

Note: Before you begin, back up any custom packages that you have created for FIX Module, in case you need them in the future.

To uninstall FIX Module 7.2

1. Shut down the Integration Server that hosts FIX Module 7.2.
2. Start Software AG Uninstaller, as follows:

System	Instructions
Windows	In the Add or Remove Programs window, select the installation directory of the Integration Server on which FIX Module 7.2 is installed.
UNIX	Navigate to the *installation_directory* \bin directory of the installation that includes the Integration Server on which FIX Module 7.2 is installed and enter `uninstall` (wizard) or `uninstall -console` (console mode).

3. In the product selection list, select **eStandards > webMethods FIX Module 7.2**. Choose to uninstall Program Files.
4. Software AG Uninstaller does not delete files that you have created or configuration files associated with FIX Module, nor does it delete the directory structure that contains those files. If you do not want to save those files, in the *Integration Server_directory* \packages directory, delete the WmFIX package directory.
5. Restart the Integration Server on which you uninstalled FIX Module 7.2.

3 Configuring webMethods FIX Module

Overview

This chapter describes how to use My webMethods to configure FIX Module. Once configuration is complete, FIX Module is ready to send and receive FIX messages.

Before you can configure FIX Module, you must have already installed and configured the FIX engine. For more information about configuring the FIX engine, see your FIX engine documentation.

Note: When configuring the Appia engine in the appia.ini configuration file, set the **middleware_outbound_reliable** property to ON to prevent the loss of FIX messages that are being sent during exceptional conditions, such as unexpected shut down of Integration Server or the Appia engine.

Configuring FIX Module

Configuring FIX Module comprises setting the FIX engine sessions to which you are connecting and defining the retry thread parameters:

- **Session connections.** These are the FIX engine sessions to which FIX Module connects (to exchange messages between the FIX engine and FIX Module).
- **Connection retry thread parameters.**FIX Module retry thread parameters specify the number of times to retry a connection and the interval between the retries when the connection to the FIX engine is lost. There are two parameters that control the behavior of the thread:
  - **Retry Limit** is the number of times the thread tries to connect to the FIX engine when the connection between FIX Module and the FIX engine is lost.
  - **Retry Sleep Interval** is the number of milliseconds the connection retry thread waits between connection attempts.

  At startup, FIX Module determines if there is an active connection with the FIX engine using a "version" command to create a connection retry thread. If there is no active connection, the thread continues attempting to connect to the FIX engine, waiting the length of time specified in the **Retry Sleep Interval** between each connection attempt. If the thread cannot connect after the number of times specified in the **Retry Limit**, a log message is generated indicating that the thread has been killed and the WmFIX package must be reloaded. Once the package is reloaded, FIX Module starts a new connection retry thread and attempts to connect to the FIX engine again.

After you have successfully defined your session connections and retry thread parameters, you can perform administration tasks and issue operator commands to the FIX engine. For more information about working with the FIX engine, see "Working with FIX Engine Connections" on page 35.

The remainder of the chapter provides the procedures for configuring FIX Module.

Specifying Session Connections in FIX Module

This section describes how to use My webMethods to configure FIX engine sessions. These instructions assume that the FIX engine is Appia.

To specify session connections in FIX Module

1. In My webMethods, select **Administration > Integration > B2B > FIX Configuration**.
2. In the Integration Server Configuration panel, click **Configure**.
3. In the Server Details panel, enter the host name or IP address and the port number of the Integration Server to which you want to connect and click **Save**.
4. In the Appia Configuration panel, click **Configure**.
5. In the Appia Connection Details panel, enter the host name or IP address and the port number of the Appia server to which you want to connect and click **Retrieve Sessions**.
6. Specify the sessions that you want to use to exchange FIX messages by selecting the session IDs from the **Available** list and clicking the single right arrow to move them to the **Selected** list. To move all of the session IDs to the **Selected** list, click the double right arrow.
7. Click **Save**.

FIX Module reinitializes itself, cleans up existing middleware connections, and creates new middleware connections for each of the newly selected sessions.

Before continuing with the next steps, check the Integration Server logs to confirm that the middleware connection for each session connects without error.

Note: If you do not want to use My webMethods to configure the Appia connection, you can execute the wm.fix.appia.connection:setConnectionDetails service to set the Appia connection details.

Configuring FIX Module Connection Retry Thread Parameters

This section describes how to configure the FIX Module connection retry thread parameters using My webMethods. For more information about these parameters, see "Configuring FIX Module " on page 32.

To configure FIX Module connection retry thread parameters

1. In My webMethods, select **Administration > Integration > B2B > FIX Configuration**.
2. In the Connection Retry Details panel, click **Configure**.
3. Specify parameters as follows:

- For **Retry Limit**, type an integer to specify the number of times the thread should try to connect to the FIX engine when the connection between FIX Module and the FIX engine is lost. The default is 10.
- For **Retry Sleep Interval**, type the number of milliseconds the connection retry thread should wait between connection attempts. The default is 30000 milliseconds.

4. Click **Save**.

FIX Module discards the number of connection attempts already tried and starts a new connection retry thread with the newly configured parameters.

Note: If you do not want to use My webMethods to configure the retry thread parameters, you can execute the service wm.fix.appia.connection:setRetryThreadDetails to set the retry thread parameters.

4 Working with FIX Engine Connections

Overview

This chapter describes how to use My webMethods to administer your FIX engine connection, including how to establish, view, and disconnect a FIX engine session. It also provides information for working with Appia as your FIX engine.

Working with Appia

FIX Module only works with NYSE Technologies Appia; therefore, you must use Appia as the FIX engine. This section describes Appia middleware event handling by FIX Module and how to send operator commands to Appia.

Appia Middleware Events

Appia sends inbound and outbound application message events, as well as details about session and server events, by publishing related middleware events to FIX Module. After these middleware events are delivered to FIX Module, FIX Module submits them to Trading Networks for recognition and processing. Trading Networks matches the event to a TN document type and extracts the information specified by the document type attributes. Trading Networks then executes the processing rule assigned to the document type.

Appia Document Type Attributes

Each of the middleware document types has one or more of the following attributes:

- `EventType`—Numeric value indicating the type of middleware event contained in the message.
- `SessionID`—The session responsible for sending the message.
- `Protocol`—The protocol associated with the message, such as FIX 42.
- `ClientMsgID`—A unique identifier used to match acknowledgement events with the messages that triggered them.
- `MsgType`—The protocol-specific type of the message. This attribute is only populated for events related to incoming or outbound messages.
- `MsgSeqNum`—The sequence number associated to the event. This attribute is only populated when the associated message relates to an incoming or outbound message.

For more information on Appia event types, see the Appia documentation. For more information about defining and managing document type attributes, see the Trading Networks user's guide for your release. See "About this Guide" for specific document titles.

TN Document Types for Appia Middleware Events

FIX Module creates the following TN document types for handling Appia middleware events during startup:

- `Appia Internal Middleware Event`—Used to provide information about the middleware. These events are always delivered to the application, and include `MIDDLEWARE_OPENED` and `MIDDLEWARE_CLOSED`, indicating the state of the Appia middleware connection.
- `Appia Inbound Message Event`—Used to deliver inbound application messages sent from your partner to FIX Module, as well as inbound session-level messages and error notifications relating to inbound messages.
- `Appia Outbound Message Event`—Provides information about the progress of an application message sent from FIX Module to the Appia engine, for example, the progress of the message delivery, acknowledgments, or error details.
- `Appia Session Event`—Indicates a change in the session state, such as a connect or a disconnect event. By default, the session event attribute is set to deliver session events to FIX Module.
- `Appia Global Event`—Issued when a condition occurs in the Appia server that can impact multiple sessions. For example, Appia will issue a global event when the Appia engine is shut down.
- `Appia Miscellaneous Event`—Includes notifications for when a restore is finished and when an outbound message has a serious problem.

Issuing Operator Commands to the Appia Server

You can issue the following operator commands from My webMethods to the Appia server.

- `Connect`—Establishes a connection between two sessions.
- `Disconnect`—Disconnects the specified session.
- `EOD`—Runs “End of Day” processing for the specified session.
- `Hold`—Holds an active session, maintaining the connection and allows the exchange of session-level messages, but rejects application messages from the counterparty. In this state, the client application can still send application messages. A session can be removed from the HOLD state by issuing the release command or by restarting Appia.
- `Release`—Takes the specified session out of `HOLD` mode.
- `Resume`—Resumes a session from the SUSPEND state.

- `Suspend`—Suspends an active session, allowing Appia to send a logout message to the counterparty and disconnect the socket connection. Subsequent logon attempts will fail and the session will remain suspended until the `Resume` command is issued.

Note: In Integration Server cluster mode, operator commands can be issued from any Integration Server node of the cluster.

To issue operator commands to the Appia server

1. In My webMethods, select **Administration > Integration > B2B > FIX Administration**.
2. In the Sessions panel, select the session ID of the Appia server to which you want to send the command.
3. In the Session Commands panel, select the operator command you want to issue.

   For more information about using operator commands, see the Appia documentation.
4. Click **Send**.

Viewing FIX Engine Session Information

To view information about configured FIX engine sessions

1. In My webMethods, select **Administration > Integration > B2B > FIX Administration**.
2. In the Sessions panel, select the session name for which you want to view details.

   The Session Info panel displays the details for the session you selected.

Subscribing to Appia Events

To subscribe to an Appia event

1. In My webMethods, select **Administration > Integration > B2B > FIX Administration**.
2. In the Sessions panel, click the session name to which you want to subscribe.
3. In the Middleware Configuration panel, check or uncheck **Global Messages**, **Session Messages**, and **Session Events**, based on your requirements.

For more information about Appia events, see the Appia documentation.

5 Sending and Receiving FIX Messages

Overview

FIX Module enables you to exchange FIX messages with your partners. The module translates all incoming and outgoing messages into the format appropriate for the receiving system. Integration with Trading Networks provides the flexibility to define rules to specify exactly how to process inbound and outbound documents.

This chapter explains how to use FIX Module to send and receive messages.

> **Note:** To use FIX Module, you must have Trading Networks already installed and running.

FIX Module exchanges both application and administrative messages with the FIX engine:

- **Application messages.** Messages exchanged with your trading partner through the FIX engine, for example, New Order Single and Execution Report.
- **Administrative messages.** Messages exchanged only with the FIX engine.

FIX Module stores all messages that it sends and receives in the Trading Networks database.

Configuring Trading Networks for Message Processing

To exchange documents with your partners, complete the following tasks in Trading Networks:

- **Create Partner Profiles.** Profiles identify the partners with whom you exchange business documents. They provide a summary of information about your partner and information about how to exchange documents. For complete information about defining and managing partner profiles, see the Trading Networks administration guide for your release. See "About this Guide" for specific document titles.
- **Define FIX_IDs.** At startup, FIX Module adds the `FIX_ID` ID type to Trading Networks. The FIX_ID (external ID type) field is defined in the profile and identifies the sender or receiver of documents. You must define the FIX_ID for every profile. Define the FIX_ID to match the values defined in the appia.ini configuration file as follows:
  - In your profile, define the value of the FIX_ID the same as the `local_firm_id`.
  - In your partner's profile, define the value of the FIX_ID the same as the `remote_firm_id`.

    > **Note:** You must define the FIX_ID in the profile of every partner with whom you exchange documents.

For complete information about defining external IDs in partner profiles, see the Trading Networks administration guide for your release. See “About this Guide” for specific document titles.

- **Create Document Types.** Document types define the properties of incoming documents and how to process them. For complete information about defining and managing document types, see the Trading Networks administration guide for your release. See “About this Guide” for specific document titles.
- **Create Processing Rules.** Processing rules specify the actions that you want Trading Networks to perform on incoming documents. These rules can execute one or more of the following actions: execute a service (synchronously or asynchronously), send an e-mail alert, change the user status for the document, deliver the document to a specified receiver using one of four delivery methods, or respond with a specified message. Trading Networks only executes a processing rule if the incoming document matches one of the document types that you defined.

  Note: There are no default processing rules. Therefore, you must define at least one rule for processing and routing incoming documents.

  For complete information about defining and managing processing rules, see the Trading Networks administration guide for your release. See “About this Guide” for specific document titles.

Creating Custom Fields

FIX Module allows you to create custom fields and include them in the messages that you exchange with your trading partners. A custom field is a user-defined field that is not part of the standard FIX specification, but can be created and added to an IS document type so that two trading partners can tailor messages to their specific needs.

FIX Module has an IS document type that corresponds to each FIX message type in the FIX specification. To create a custom field, you must edit the corresponding IS document type as indicated in the following instructions.

Important: Both partners must define the same custom fields and use the same message structure to successfully exchange messages.

To create a custom field for a FIX message

1. In Designer, navigate to the WmFIXMessages package and open the FIX IS document in which you want to add a field.

   For example, to add a new field to the FIX50AllocationReport, open the IS document wm.fix.rec.FIX50:FIX50AllocationReport in the WmFIXMessages package.
2. Add a new field to the IS document, and type the name of the new field.
3. Edit the Comments section and add the tag number of the field.

4. Repeat these steps for every new field that you want to add.
5. For the changes to take effect, reload FIX Module from either Integration Server Administrator (by clicking the Reload icon for the WmFIX package on the Management window) or from Designer (by right-clicking the WmFIX package and selecting the **Reload Package** option from the menu.

Inbound Message Processing

FIX Module handles end-to-end message processing for messages received from the FIX engine. FIX Module transforms the incoming message from a raw FIX message into an IData message and stores it (with both formats) in the Trading Networks database. Trading Networks then proceeds with processing the message according to the profiles and processing rules defined for the document type that matches the incoming message.

When using Integration Server clustering, as long as one Integration Server node of the cluster is running, FIX Module will continue to receive FIX messages and store them in Trading Networks for processing.

Outbound Message Processing

FIX Module performs all the processing steps required to send outgoing messages. Using the wm.fix.tn.trp:send service, FIX Module converts messages from the original IS document format into the FIX standard format (known as a raw FIX message). It stores each outgoing message in the Trading Networks database and sends the message to the FIX engine (Appia server) for transmission to the business partner.

Every message is stored in Trading Networks with a status that indicates its transmission state to the FIX engine: either `FIX_SENT`, `FIX_QUEUED` or `FIX_SEND_ERROR`. For more information about these statuses or message storage and queuing, see "Message Storage and Queuing" on page 42.

Note: FIX Module always verifies that there is a connection to the FIX engine before sending messages. In the event that a session is not connected, FIX Module automatically issues a `connect` operator command before sending the message.

When using Integration Server clustering, FIX Module can send a message from any node in the cluster.

Message Storage and Queuing

FIX Module ensures reliable message delivery, storing every message sent and received in the Trading Networks database. In the event that there is no connection to the FIX engine during message transmission, FIX Module's queuing functionality guarantees

that no message will ever be lost. When the connection to the FIX engine is lost, FIX Module stores unsent messages with a status of `FIX_QUEUED`. After the connection is restored, FIX Module automatically sends all queued messages in the order in which they were stored and updates the status to `FIX_SENT`.

FIX Module stores every message with a status indicating its transmission state. The following statuses are possible:

- `FIX_RECEIVED`—The message was received from the FIX engine.
- `FIX_SENT`—The message was successfully sent to the FIX engine.
- `FIX_QUEUED`—The message is queued for transmission because there is no active connection to the FIX engine. Once the connection is restored, FIX Module will automatically send messages in the order in which they were stored and update the status to `FIX_SENT`.
- `FIX_SEND_ERROR`—There was an error during message transmission.

In addition to the aforementioned statuses, the message status may also correspond to its processing state within Appia. For every message that FIX Module sends to Appia, Appia returns an event that reflects its processing state. When FIX Module receives these events, it retrieves the relevant Integration Server document in Trading Networks and updates the FIX status to reflect the Appia event status. These events may arrive in any order.

The following Appia related user statuses are possible:

- `APPIA_MESSAGE_VALIDATED`—Appia validated the message and returned a `MESSAGE_VALIDATED` event status.
- `APPIA_MESSAGE_SENT`—Appia sent the message to your partner and returned a `MESSAGE_SENT` event status.
- `APPIA_MESSAGE_COMMITED`—Appia committed the message to its persistence store and returned a `MESSAGE_COMMITED` event status.
- `APPIA_MESSAGE_VALIDATION_ERROR`—An error occurred during the transaction due to invalid message syntax.

Viewing Message Transaction History

FIX Module stores all messages sent, received, queued, and failed in the Trading Networks database. You can view both the content and state of these messages in the Transaction Analysis screen in My webMethods. For more information about viewing messages, see the Trading Networks administration guide for your release. See “About this Guide” for specific document titles.

6 Error Handling

Overview

This chapter describes message logging and exception handling in FIX Module. See "FIX Module Error Codes" on page 46 for a list of error codes and supporting information.

FIX Module Message Logging

FIX Module uses Integration Server's logging mechanism to log informational, warning, and error messages to the server log of Integration Server. To view the server log, use the Integration Server Administrator. For information about viewing and configuring the server log, see the Integration Server administration guide for your release. See "About this Guide" for specific document titles.

FIX Module adds errors, warnings, and informational messages to the server log, using the format, FIX.000*n* .*nnnn* , where:

- FIX is the product code that indicates the message is issued by FIX Module.
- 000*n* is the major error code, where *n* can be any of the following values:
  - 0—FIX Module encountered a general error.
  - 1—Generated during the initialization of FIX Module (which includes the initialization of FIX engine sessions).
  - 2—Generated while receiving a FIX message.
  - 3—Generated while sending a FIX message.
- *nnnn* is the minor error code.

FIX Module Error Codes

0001.1004 The sessionID [{0}] is invalid. Ignoring this session

Explanation: Warning. The specified sessionID does not exist in the Appia engine. FIX Module ignores this session and continues initializing the other valid sessions.

Action: None.

0001.1006 Failed to connect to Appia engine {0}: {1}

Explanation: An error occurred when connecting to the specified Appia engine.

Action: Ensure that the host and port specified are valid or that the Appia engine is running at the specified host and port, and then reload FIX Module.

0001.1008 FIX Module is not yet configured. See FIX Module Installation and User's Guide to complete the configuration

Explanation: FIX Module needs the host, port, and session information of the Appia engine.

Action: Use the FIX Module-My webMethods Server user interface to set the host, port, and session of the Appia engine.

0001.1012 FIX Module is not yet initialized. Check connection to Appia engine and try again.

Explanation: A request was sent to FIX Module before FIX Module was initialized.

Action: Ensure FIX Module is configured successfully before sending any requests.

0001.1023 FIX Module connection retry limit was reached

Explanation: The connection with the Appia engine could not be established even after retrying for the specified retry limit.

Action: Ensure connectivity with the Appia engine is restored, and then reload FIX Module.

0001.1025 FIX Module cannot connect to the Appia engine. Will retry up to {0} more time(s)

Explanation: Warning. FIX Module lost connectivity with the Appia engine, either because the Appia engine went down or because network connectivity was lost.

Action: Ensure that the connectivity to the Appia engine is established. FIX Module automatically recognizes when connectivity is restored and reloads itself to reinitiate the connection with the Appia engine.

0001.1026 FIX Module connection retry thread is being killed. Reload the FIX Module

Explanation: When FIX Module loses connectivity with the FIX engine, it contacts the engine at fixed intervals to attempt to reestablish the connection. When the configured retry limit is reached, FIX Module stops attempting to reconnect.

Action: Ensure connectivity to the Appia engine, and then manually reload FIX Module.

0001.1027 FIX Module is not configured. See the FIX Module installation and users guide for details on how to configure the FIX Module.

Explanation: During FIX Module startup, FIX engine configuration details were unavailable.

Action: Configure the FIX engine connection details using the FIX Module-My webMethods user interface.

0001.1033 Unrecoverable error in connection retry thread. Reload FIX Module. Error Message {0}

Explanation: FIX Module could not handle an error that occurred.

Action: Reload FIX Module.

0000.1015 Failed to retrieve session list from FIX engine. Supply a valid host and port and try again

Explanation: When attempting to retrieve the session list from FIX engine, the specified host and port for the FIX engine are invalid.

Action: Specify a valid FIX engine host and port and try again.

0000.1021 Error occured while retrieving session list from Appia engine. Reason: {0}

Explanation: FIX Module could not retrieve the FIX engine session list because the specified FIX engine is not running.

Action: Ensure that the FIX engine at the specified host and port is running before attempting to retrieve the session list.

0000.1022 Error occurred while storing FIX engine connection details in repository. Detailed message: {0}

Explanation: The FIX engine connection details were not stored in the repository because the Trading Networks database where the connection details reside was unavailable.

Action: Ensure that the Trading Networks database where the FIX engine connection details are stored is available.

0000.1023 Error occurred while retrieving FIX engine connection details from repository. Detailed message: {0}

Explanation: FIX Module could not retrieve the FIX engine connection information from the Trading Networks database.

Action: Ensure that the Trading Networks database is available.

0000.1025 Error occurred while storing connection retry thread details. Detailed message: {0}

Explanation: FIX Module connection retry thread information was not stored successfully.

Action: Ensure that the Trading Networks database is available.

0000.1026 Error occurred while retrieving connection retry thread details from repository. Detailed message: {0}

Explanation: FIX Module could not retreive the FIX engine connection retry thread information from the Trading Networks database.

Action: Ensure that the Trading Networks database is available.

0000.1032 Error occured while executing the operator command. Detailed message: {0}

Explanation: An error occured while executing the specified FIX engine operator command.

Action: Ensure the syntax of the operator command being executed is correct and the FIX engine is running.

0000.1044 Error initializing FIX Module. Check if Appia jar file is placed in the WmFIX package.

Explanation: During FIX Module startup, the Module could not find the appia.jar file.

Action: Copy the appia.jar file from the *AppiaServer_homeDirectory*\lib folder to the *Integration Server_directory*\packages\WmFIX\code\jars directory. Then restart Integration Server.

0000.1045 FIX Module cannot find the WM_FIX_CONFIGRATION_PARAMS table in the Trading Networks database. Create the table and restart FIX Module.

Explanation: FIX engine connection information is stored in the WM_FIX_CONFIGURATION_PARAMS table in the Trading Networks database. During FIX Module startup, FIX Module could not find the table in the Trading Networks database.

Action: Ensure that the WM_FIX_CONFIGURATION_PARAMS table exists. If it does not exist, create it by executing the .sql script provided in the WmFIX\config folder on the same database instance where Trading Networks is connected. Then reload FIX Module.

0002.1007 Error persisting inbound event "{0}". Cause: {1}

Explanation: An inbound Appia event cannot be persisted in Trading Networks.

Action: Examine the cause stated in the error message and contact Software AG Global Support.

0002.1009 Message Type dictionary for FIX version {0} is not loaded. Either the version is wrong or FIX module is not initialized.

Explanation: FIX Module received a FIX message for a version that is not supported.

Action: If the indicated FIX message version is not one that FIX Module supports, contact Software AG Global Support.

0003.1000 Operator command {0} to session {1} failed. Ensure that the Appia engine has network connectivity and the Appia engine is running, and then try again

Explanation: The operator command failed to execute, probably because connection with the Appia engine was lost.

Action: FIX Module automatically reconnects with the Appia engine after losing its connection. Try running the operator command again.

0003.1003 Could not retrieve the application message version for the session {0}.

Explanation: FIX Module is attempting to send a FIX message of FIXT type and cannot obtain the application message version because there is a mismatch in FIX version numbers.

Action: This error might occur if the session receiving the message is not configured for messages of type FIXT. Ensure that the configuration of the FIX engine session version and the FIX message version are the same.

0003.1004 The value {0} for the ApplVerID is invalid. Specify a valid value and try again. Refer to FIX specification for a list of valid values

Explanation: The FIX message being sent has an invalid value for the ApplVerID field.

Action: Ensure a valid value based on FIX specification is set for the ApplIVerID field of the FIX message, and then try to send the FIX message again.

0003.1005 The field {0}, of the header is mandatory.

Explanation: FIX Module is sending a message that is missing values for required header fields.

Action: Specify the values for the mandatory fields, and then resend the message.

Summary of Services

The WmFIX package contains services for configuring the Appia engine connections and for retrieving and managing Appia sessions. My webMethods Server uses these services to retrieve and display Appia connection and session parameters in My webMethods.

The following table summarizes the services available.

Service	Description
wm.fix.appia.connection:getAppiaSessions	Retrieves the list of configured Appia sessions for the specified Appia server host and port.
wm.fix.appia.connection:getConnectionDetails	Retrieves the connection details of the Appia server that is configured in FIX Module.
wm.fix.appia.connection:getRetryThreadDetails	Retrieves connection retry thread details.
wm.fix.appia.connection:setConnectionDetails	Sets the Appia server connection and session details.
wm.fix.appia.connection:setRetryThreadDetails	Sets the connection retry thread parameters.
wm.fix.appia.session:configureEvents	Sets a filter for Appia events. Events that can be enabled or disabled are global messages, session messages, session events, and outbound message events. If enabled, FIX Module receives these events and persists them in Trading Networks.

Service	Description
wm.fix.appia.session:getEventConfig	Retrieves Appia event filter information (set by the wm.fix.appia.session:configureEvents service) for configured Appia events in the given session.
wm.fix.appia.session:getSessionInfo	Retrieves information about a configured Appia session.
wm.fix.appia.session:getSessions	Retrieves a list of Appia session IDs configured for FIX Module.
wm.fix.appia.session:sendCmd	Issues operator commands to the configured Appia server for a specific session.
wm.fix.appia.session:sendRawFIXMessage	Sends the given raw FIX message directly to the Appia engine over a given session.
wm.fix.appia.session:addFIXMTsToIgnoreList	Adds FIX messages to the Ignore List, these messages will not be stored in the Trading Networks database.
wm.fix.appia.session:getFIXMTsFromIgnoreList	Retrieves a list of message types from an Ignore List for a specified sessionID.
wm.fix.appia.session:removeFIXMTsFromIgnoreList	Removes a list of message types from an Ignore List for a specified sessionID.

wm.fix.appia.connection:getAppiaSessions

Retrieves the list of configured Appia sessions for the specified Appia server host and port.

Input Parameters

host **String** Host name or IP address of the Appia server for which you want to retrieve session IDs.

port **String** Port number that corresponds to the Appia server.

Output Parameters

sessionIDs **String List** Optional. List of the session IDs configured on the specified Appia server.

error **String** Optional. Errors generated while retrieving the session IDs.

wm.fix.appia.connection:getConnectionDetails

Retrieves the connection details of the Appia server that is configured in FIX Module.

Input Parameters

None.

Output Parameters

connectionDetails **Document** Information about the host, port, and session IDs of the Appia engine configured in FIX Module.

- *host* —**String** Host name or IP address of the Appia server.
- *port* —**String** Port number of the Appia server.
- *sessionIDs* —**String List** Comma-separated list of the session IDs configured for the specified Appia server.

wm.fix.appia.connection:getRetryThreadDetails

Retrieves connection retry thread details.

Input Parameters

None.

Output Parameters

retryThreadDetails **Document** The configuration details of the retry thread.

- *connectionRetryLimit* —**String** Number of times to attempt to reconnect to the Appia server when the connection is lost.

- *connectionRetrySleepInterval* —**String** Number of milliseconds to wait in between connection attempts.

Usage Notes

For more information about the connection retry thread, see "Configuring FIX Module Connection Retry Thread Parameters" on page 33.

wm.fix.appia.connection:setConnectionDetails

Sets the Appia server connection and session details.

Input Parameters

connectionDetails **Document** Information about the host, port, and session IDs of an Appia engine configured in FIX Module.

- *host* —**String** Host name corresponding to the IP address of the Appia server.
- *port* —**String** Port number of the Appia server.
- *sessionIDs* —**String List** Comma-separated list of the session IDs configured for the specified Appia server.

Output Parameters

error **String** Optional. Error generated while defining the Appia server connections.

Usage Notes

If this service executes successfully, the existing host, port, and sessions are overwritten with the new values. FIX Module then reloads and connects to the new host, port, and sessions.

If the input host and port is invalid or the Appia engine is not running when the service executes, the existing settings are not changed and the service generates an error message.

wm.fix.appia.connection:setRetryThreadDetails

Sets the connection retry thread parameters.

Input Parameters

retryThreadDetails **Document** The configuration details of the retry thread.

- *connectionRetryLimit* —**String** Number of times to attempt to reconnect when the connection to the Appia server is lost. The default is 10.
- *connectionRetrySleepInterval* —**String** Number of milliseconds the connection retry thread should wait in between connection attempts. The default is 30000 milliseconds.

Output Parameters

error **String** Optional. Error message generated while setting the connection retry thread parameters.

Usage Notes

After this service finishes executing, the connection retry thread restarts automatically with the new values. For more information about the connection retry thread, see "Configuring FIX Module Connection Retry Thread Parameters" on page 33.

wm.fix.appia.session:configureEvents

Sets a filter for Appia events. Events that can be enabled or disabled are global messages, session messages, session events, and outbound message events. If enabled, FIX Module receives these events and persists them in Trading Networks.

Input Parameters

sessionID **String** The session that corresponds to the filter being defined. The filter is set on this session.

middlewareConfig **Document** The middleware configuration parameters for which this filter is set to enable or disable the following messages or events. The value `True` enables the message or event:

- *GlobalMessages* —**String** Enable or disable Appia global messages.
- *SessionMessages* —**String** Enable or disable Appia session messages.
- *SessionEvents* —**String** Enable or disable Appia session events.

- *OutboundMessageEvents* —**String** Enable or disable handling of outbound message events.

Output Parameters

error **String** Optional. Error message generated while setting the event configuration filter.

wm.fix.appia.session:getEventConfig

Retrieves Appia event filter information (set by the wm.fix.appia.session:configureEvents service) for configured Appia events in the given session.

Input Parameters

sessionID **String** The session ID of Appia engine.

Output Parameters

middlewareConfig **Document** The middleware configuration parameters filter currently set for the specified session. The value `True` indicates that the Appia message or event filter is enabled.

- *GlobalMessages* —**String** Status of Appia global messages filter.
- *SessionMessages* —**String** Status of Appia session messages filter.
- *SessionEvents* —**String** Status of Appia session events filter.
- *OutboundMessageEvents* —**String** Status of the outbound message events filter.

error **String** Optional. Error messages generated while retrieving the event configuration filter for the given session.

wm.fix.appia.session:getSessionInfo

Retrieves information about a configured Appia session.

Input Parameters

sessionID	**String** Session ID that corresponds to the Appia session details being retrieved.

Output Parameters

sessionInfo	**String** Optional. Session information for the specified session, formatted as a sequence of comma-separated key=value pairs.
error	**String** Optional. Error messages generated while retrieving Appia session details.

wm.fix.appia.session:getSessions

Retrieves a list of Appia session IDs configured for FIX Module.

Input Parameters

None.

Output Parameters

sessionList	**String list** Optional. List of Appia session IDs configured for FIX Module.
error	**String** Optional. Error messages generated while retrieving Appia session IDs.

wm.fix.appia.session:sendCmd

Issues operator commands to the configured Appia server for a specific session.

Input Parameters

sessionID	**String** Session ID of the Appia server to which you want to issue a command.
command	**String** A valid Appia operator command.

Output Parameters

output	**String** Optional. Appia engine response to the operator command.
error	**String** Optional. Error generated while sending Appia commands.

Usage Notes

For additional information about Appia operator commands, see the Appia documentation.

wm.fix.appia.session:sendRawFIXMessage

Sends the given raw FIX message to the Appia engine over a given session. This service sends the message directly to the Appia engine without storing the message in Trading Networks. The message is not queued if the send fails.

Input Parameters

rawMessage	**String** Raw FIX message to be sent to the Appia engine.
targetSessionId	**String** Session ID of the Appia engine over which the message will be sent.
messageId	**String** A unique ID for the message.

Output Parameters

status	**String** Status of the send operation: ■ `True`—Indicates that the message was sent successfully. ■ `False`—Indicates that the send operation failed.

wm.fix.appia.session:addFIXMTsToIgnoreList

Adds FIX messages to the Ignore List, these messages are not persistent and will not be stored in the Trading Networks database. Provide only the Message Tag as input to this list. For example, for Heartbeat – '0' should be provided as input.

Input Parameters

sessionID **String** Session ID of the Appia engine.

msgTypeList **String List** List of FIX message types.

override **String** Optional. Indicate if existing message type Ignore List is to be replaced by a new list.

- `True`— Indicates that if a list exists, it should be replaced by the new list.
- `False`— Indicates that if a list exists, new message types should be added to the existing list.

Output Parameters

error **String** Optional. Error generated when adding a message type to the list.

status **String** Status of the add operation:

- `True`— Indicates that the service executed without any errors and message types have been added tot the list.
- `False`— Indicates that the add operation has failed.

wm.fix.appia.session:getFIXMTsFromIgnoreList

Retrieves a list of message types from an Ignore List for a specified sessionID. An error is displayed if the list is not available for a specific message type.

Input Parameters

sessionID **String** The session ID of Appia engine.

Output Parameters

fixMTs **String List** List of FIX message types fetched for a specified sessionID.

error **String** Optional. Error generated while fetching the list.

status **String** Status of the fetch operation.

- `True`— Service executed without any error.
- `False`— Indicates that the fetch operation failed.

wm.fix.appia.session:removeFIXMTsFromIgnoreList

Removes the list of message types from the Ignore List for the given sessionID. Returns an error if IgnoreList is empty or input items cannot be found in the list.

Input Parameters

sessionID **String** The sessionID of Appia engine.

msgTypeList **String List** Optional only if removeAll flag is set to `True`. List of FIX message types to be removed from the Ignore List.

removeAll **String** Optional. Flag to indicate if the Ignore List is to be deleted.

- `True`— Indicates that the existing list (if exists) should be deleted.
- `False`— Indicates to remove only selected message types from the Ignore List.

Output Parameters

error **String** Optional. Error generated while removing the list.

status **String** Status of the fetch operation.

- `True`— Service executed without any error.
- `False`— Indicates that the delete operation failed.

Summary of Services

The WmFIX package contains services to convert messages exchanged with the FIX engine and to assist with upgrading to the latest version of FIX Module. Transport, utility, and cluster-related services are also provided.

FIX Module services are divided into the following folders:

Folder	Contains services you use to...
wm.fix.cluster	Manage a clustered Integration Server environment
wm.fix.format	Format messages in either raw FIX or IData format
wm.fix.tn.trp	Send and receive messages
wm.fix.util	Manage utilities

The following table summarizes the FIX Module services.

Service	Description
wm.fix.cluster:getClusterMembers	When using Coherence clustering, retrieves the list of all the nodes of the cluster if Integration Server is cluster enabled. If Integration Server is not cluster-enabled, this service returns an error.
wm.fix.format:convertFIXToIData	Converts an incoming raw FIX message into an IData object.
wm.fix.format:convertIDataToFix	Converts a FIX IData object into a raw FIX message to be sent to a FIX engine.
wm.fix.tn.trp:send	Processes outbound messages.
wm.fix.util:flushQueuedMessages	Clears the queue of messages that no longer require processing by FIX Module.
wm.fix.util:modify42FieldNames	Updates field names in your custom package for FIX Message version 4.2.

wm.fix.cluster:getClusterMembers

When using Coherence clustering, retrieves the list of all the nodes of the cluster if Integration Server is cluster enabled. If Integration Server is not cluster-enabled, this service returns an error. This service is not available with Terracotta clustering.

Input Parameters

None.

Output Parameters

clusterNode **Document** An array of all nodes of the cluster.

- *memberAddress* —**String** IP address of the cluster node.
- *memberPort* —**String** Port of the cluster node.
- *memberUID* —**String** Unique ID of the cluster member (unique among all the members of the cluster). The ID is assigned to the node by Integration Server.
- *isConnected* —**String** Indicates if the node is connected to the Appia engine. In a cluster only one node is connected to the Appia engine.
  - `True`—The node is connected to Appia engine.
  - `False`—The node is not connected to Appia engine.

error **String** Optional. Error message generated while retrieving the cluster member list.

wm.fix.format:convertFIXToIData

Converts an incoming raw FIX message into an IData object.

Input Parameters

fixMsg **String** Raw FIX message.

sessionID **String** Optional. SessionID of the FIX engine from which FIX Module received the raw FIX message. This parameter applies for FIX messages of version 5.0 or later.

Output Parameters

fixIData **IData** IData object conforming to the IS document type in the WmFIXMessages package.

error **String** Optional. Errors generated during message conversion.

wm.fix.format:convertIDataToFix

Converts a FIX Data object into a raw FIX message to be sent to a FIX engine.

This service works with FIX Module tag dictionaries to correlate tag numbers with tag names. Tag dictionaries are located in the /config/dictionaries folder of the WmFIXMessages package.

Input Parameters

fixIData	**IData** IData object conforming to the IS document type in the WmFIXMessages package.
sessionID	**String** Optional. SessionID of the FIX engine to which FIX Module will send the created raw FIX message. this parameter applies for FIX messages of version 5.0 or later.

Output Parameters

None.

wm.fix.tn.trp:send

Processes outbound messages by executing the following steps:

- Converts the FIX message from IData format into raw FIX message format.
- Stores the message (with both formats) in the Trading Networks database with the **User Status** `FIX_QUEUED`.
- Sends the message to the FIX engine. If the message sends successfully, the service updates the **User Status** to `FIX_SENT`.

If the connection to the FIX engine is lost, FIX Module queues the FIX messages. As soon as the connection is restored, FIX Module resubmits messages to the FIX engine in the order in which they were stored.

Input Parameters

fixIData	**Document** FIX message IS document (IData). The input should be one of the Integration Server documents included with the WmFIXMessages package.
sessionID	**String** SessionID of the FIX engine to which the FIX message is sent.

Output Parameters

sentStatus	**String** The transmission status of the message to the FIX engine. Valid values are as follows: ■ **FIX_SENT**—Message sent successfully. ■ **FIX_QUEUED**—Message is queued to send, and will be resubmitted after the connection to the FIX engine is restored. ■ **FIX_SEND_ERROR**—An error occurred while sending the message.
TNBizDocId	**String** The Doc ID of the message.
error	**String** The error message and error reason. This field is only populated if the *sentStatus* is `FIX_SEND_ERROR`.

wm.fix.util:flushQueuedMessages

Clears the queue of messages that no longer require processing by FIX Module. The service changes the status of queued FIX messages in Trading Networks from the **User Status** of `FIX_QUEUED` to `FIX_OBSOLETE`. Changing the **User Status** eliminates the messages from the queue. Use this service when you do not want FIX Module to process the messages that are still in the queue.

Input Parameters

sessionID	**String Array** The list of sessionIDs that correspond to the FIX message (BizDocEnvelope) with a **User Status** of `FIX_QUEUED`. When this service is used, the **User Status** is updated to `FIX_OBSOLETE`.
docID	**String Array** A list of the docIDs that correspond to the messages with the **User Status** of `FIX_QUEUED`. The corresponding messages are cleared from the queue when this service is executed.

Output Parameters

resultCounts	**String** The number of messages that were removed from the queue after executing this service.

wm.fix.util:modify42FieldNames

Updates field names in your custom package for FIX Message version 4.2. For more information, see "Step 4: Change IS Document Field Names" on page 27.

Input Parameters

package	**String** Name of the package that you want to migrate from an earlier version of FIX Module.

Output Parameters

None.

Overview

webMethods FIX Module provides functionality to allow clustering to ensure high system availability. This chapter describes what clustering is and how to implement it in FIX Module.

Clustering is an advanced feature of the webMethods product suite that substantially extends the scalability, availability, and reliability of Integration Server. Clustering accomplishes this by providing the infrastructure and tools to deploy multiple Integration Servers as if they were a single virtual server, thereby allowing applications to leverage that architecture.

Clustering provides the following benefits:

- **Scalability**. Without clustering, only *vertical scalability* is possible. Requirements for increased capacity can only be met by deploying larger, more powerful machines, typically housing multiple CPUs. Clustering provides *horizontal scalability*, virtually limitless expansion of capacity by simply adding more machines of the same or similar capacity.
- **Availability**. Without clustering, even with expensive fault-tolerant systems, a system failure (hardware, java runtime, or software) may result in unacceptable downtime. Clustering provides virtually uninterrupted availability by deploying applications on multiple Integration Servers; in the worst case, a server failure produces degraded but not disrupted service.
- **Reliability**. Unlike a server farm (an independent set of servers), clustering provides the reliability required for mission-critical applications. Distributed applications must address network, hardware, and software errors that might produce duplicate (or failed) transactions. Clustering makes it possible to deliver “exactly once” execution as well as checkpoint/restart functionality for critical operations.

For complete information about Integration Server clustering, see the Integration Server clustering guide for your release. See “About this Guide” for specific document titles.

When FIX Module is used in a clustered environment, it operates as follows:

- In an Integration Server cluster, only one FIX Module node in the cluster is connected to the Appia engine at one time.
- If the FIX Module node that is connected to the Appia engine goes down, one of the other FIX Module nodes in the Integration Server cluster automatically connects to the Appia engine.
- The client can send a FIX message by connecting to any of the FIX Module nodes in the Integration Server cluster.
- The FIX Module node that is directly connected to the Appia engine handles the actual exchange of FIX messages with the Appia engine. (The connected node serves as the intermediary between the Appia engine and FIX Module instances.)
- Information about all the nodes in the cluster can be retrieved using the service, wm.fix.cluster:getClusterMembers.

Clustering Requirements for Each Integration Server

The following table describes the requirements of each Integration Server in a given cluster:

All Integration Servers in a given cluster must have identical...	For example...
Integration Server versions	All Integration Servers within the cluster must be identical. Every server must be the same version and have the same service packs and fixes (updates) applied. For example, one Integration Server cannot be version 8.0 and another Integration Server version 8.2. Additionally, all Integration Servers should use the same clustering implementation, either Coherence or Terracotta.
FIX Module packages	All FIX Module packages on one Integration Server should be replicated to all other Integration Servers in the cluster.
FIX Module versions	All FIX Module servers within the cluster must be identical. They must all be the same version, with the same fixes (updates) applied.

All Integration Servers in a given cluster must have identical...	For example...
FIX Module services	If you configure a specific FIX Module service, this same service must be installed on all servers in the cluster so that any Integration Server in the cluster can handle requests identically.

Replicating FIX Module Packages in a Clustered Environment

When you configure FIX Module, you must ensure that each Integration Server in the cluster contains an identical set of packages. You can create custom packages of generated services of FIX Module on one host and use package replication features to publish custom packages to each of the other hosts. For information on package replication, see the Integration Server clustering guide for your release. See “About this Guide” for specific document titles.

Managing FIX Module in a Cluster

When using FIX Module in a clustered Integration Server environment, you can use the wm.fix.cluster:getClusterMembers service to retrieve a list of all the nodes of the cluster and each node's IP address, port, UID, and Appia engine connection status. For complete information about this service, see wm.fix.cluster:getClusterMembers on page 60.

Index