

Apama Capital Markets Adapters

Version 10.3

July 2020

This document applies to Apama Capital Markets Adapters 10.3 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.

Copyright © 2017-2021 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <https://softwareag.com/licenses/>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <https://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <https://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

Document ID: PAMA-UG-103-20210512

Table of Contents

About this Guide	11
Online Information and Support.....	12
Data Protection.....	13
1 Release Notes	15
What's new in 10.3 release.....	16
What's new in 10.2 release.....	17
What's new in 10.1 release.....	18
What's new in 10.0 release.....	18
What's new in 9.12 release.....	19
What's new in 9.10 release.....	20
What's new in 9.9 release.....	20
What's new in 5.3 release.....	21
2 Apama Base FIX Adapter	23
Introduction.....	24
Prerequisites.....	28
FIX service monitors.....	28
IAF FIX adapter.....	71
Connectivity plug-in FIX adapter.....	81
Service monitor injection order.....	82
SSL enabled FIX sessions.....	84
Troubleshooting.....	85
3 Bank of America FXtransact FIX Adapter	89
Prerequisites.....	90
IAF configuration.....	90
Service monitor extensions.....	90
Service monitor injection order.....	90
Session configuration.....	90
Quote subscription or unsubscription.....	91
BOA InstinctFX Trading Platform.....	92
4 Barclays FX FIX Adapter	97
Prerequisites.....	98
IAF configuration.....	98
Service monitor injection order.....	99
Session configuration.....	99
MDA adapter configuration.....	99
Quote subscription and unsubscription.....	100
Example quote subscriptions and unsubscriptions.....	100

5 Bloomberg B-PIPE Adapter.....	103
Prerequisites.....	104
Transport configuration.....	104
Service monitor injection order.....	105
Symbol normalization.....	106
Market data service.....	106
Market depth service.....	107
Market depth data service.....	108
Configurability.....	108
Reference data service.....	110
Custom service.....	116
6 Bloomberg FXGO FIX Server Adapter.....	123
Prerequisites.....	124
IAF configuration.....	124
Service monitor injection order.....	124
Working with Bloomberg.....	124
Working of the Quote Provider.....	125
Quote Management.....	126
7 BM+F Bovespa UMDF Adapter.....	127
Prerequisites.....	128
Service monitor injection order.....	128
Transport properties configuration.....	129
Configuration for UMDF PUMA 2.0.....	135
Service monitor and session configuration.....	136
subscriptions management.....	143
Instrument static data management.....	144
Status reporting events.....	146
Unified news channel.....	147
Support for UMDF 2.0.....	148
Latency measurement.....	148
8 BM+F FIX Adapter.....	149
Prerequisites.....	150
IAF configuration.....	150
Service monitor extensions.....	150
Service monitor injection order.....	151
Session configuration for BMF Support monitors.....	151
Session configuration.....	152
Sample SymbolSet event.....	154
Marketdata subscription or unsubscriptions.....	154
Placing order.....	154
Sample events.....	156
New EntryPoint system.....	156

9 BNPP FX FIX Adapter.....	157
Prerequisites.....	158
IAF configuration.....	158
Service monitor extensions.....	158
Service monitor injection order.....	158
Service ID configuration.....	159
BNPP subscriptions and unsubscriptions (Quote).....	159
BNPP order management.....	160
10 Citigroup FIX Adapter.....	163
Citigroup SpotESP FIX Adapter.....	164
Citigroup Colo SpotESP FIX Adapter.....	170
11 CME iLink FIX Adapter.....	175
Prerequisites.....	176
IAF configuration.....	176
Service monitor extensions.....	177
Service monitor injection order.....	177
Session configuration.....	178
SenderSubID (header tag 50) Configuration.....	179
Audit logger configuration.....	179
Drop Copy session support.....	183
12 CME Simple Binary Encoding Adapter.....	185
Prerequisites.....	186
Configuring correlator.....	187
Configuring IAF.....	187
Using SecurityDefinition query interface.....	192
Subscribing and unsubscribing to an event.....	193
Troubleshooting.....	193
13 Credit Suisse SER FX FIX Adapter.....	195
Prerequisites.....	196
IAF configuration.....	196
Service monitor extensions.....	196
Service monitor injection order.....	196
Service configuration.....	197
CS subscriptions and unsubscriptions (Quote).....	198
CS order management.....	198
14 Currenex FIX Adapter.....	199
Prerequisites.....	200
IAF adapter configuration.....	200
Session configuration.....	200
Service monitor extensions.....	201
Service monitor injection order.....	201

Password management.....	201
Marketdata subscriptions.....	202
Order management.....	202
Currenex FIX MDA adapter.....	203
15 Deutsche Bank Autobahn FIX Adapter.....	205
Prerequisites.....	206
IAF configuration.....	206
Service monitor injection order.....	207
Platforms supported.....	207
16 EBS Spot Ai FIX Adapter.....	209
Prerequisites.....	210
Connecting to EBS AI FIX.....	210
Service monitor extensions.....	210
Service monitor injection order.....	210
Session configuration.....	211
UserLogonRequest.....	212
UserLogoutRequest.....	212
Password Management.....	212
Querying Login Response Fields.....	213
EBS Spot Ai Depth of MV.....	214
Market data subscription.....	214
Order management support.....	216
Connecting to EBS FIX 4.4.....	218
Sample Messages and events.....	221
17 FXall TCPI Server Adapter.....	233
Prerequisites.....	234
Transport properties configuration.....	234
Service monitor injection order.....	235
Service monitor and session configuration.....	235
Transport events.....	237
Working of the subscription provider.....	238
Order management.....	240
Info Management.....	240
Sample events.....	241
Reject Codes for Deals and QuoteRequests.....	242
Status reporting events.....	243
18 FXCMPPro FIX Adapter.....	245
Prerequisites.....	246
IAF configuration.....	246
Service monitor injection order.....	246
Session configuration.....	246
Sample events.....	247
Password Management.....	247

19 Goldman Sachs FX FIX Adapter	249
Prerequisites.....	250
IAF configuration.....	250
Service monitor extensions.....	250
Service monitor injection order.....	251
Service ID configuration.....	251
Session configuration.....	251
Session configuration parameters.....	251
GS Requirements or Recommendations.....	252
Quotes subscription extra parameters.....	252
Orders (Trade Requests) extra parameters.....	252
Goldman Sachs subscriptions and unsubscriptions (Quote).....	252
Sending orders using OMS interface NewOrder(NewOrderSingle).....	253
20 HotSpot FX FIX Adapter	255
Prerequisites.....	256
IAF configuration.....	256
HotSpot trading session configuration.....	256
Service monitor injection order.....	256
MarketData subscriptions.....	257
Order management.....	257
21 HSBC FX FIX Adapter	259
Prerequisites.....	260
IAF configuration.....	260
Service monitor injection order.....	260
Service ID configuration.....	260
Session configuration.....	260
Quote subscription or unsubscription.....	261
Order placing.....	261
22 JPMC FX FIX Adapter	263
Prerequisites.....	264
IAF configuration.....	264
Service monitor extensions.....	264
Service monitor injection order.....	264
Service ID configuration.....	265
Session configuration.....	265
JPMC requirements/recommendations.....	265
Market Data subscription EXTRA PARAMS.....	266
JPMorgan order management.....	267
23 kdb+ Adapter	269
Prerequisites.....	270
Features Supported.....	270
Configuration.....	270

Service monitors and injection order.....	272
Starting the Adapter.....	272
Update API call.....	274
24 Lava FX FIX Adapter.....	277
Prerequisites.....	278
IAF configuration.....	278
Service monitor injection order.....	279
Session configuration.....	279
Sample events.....	279
25 Morgan Stanley FX FIX Adapter.....	281
Prerequisites.....	282
IAF configuration.....	282
Service monitor injection order.....	282
Quote stream.....	282
MSFX Ladder Pricing.....	284
26 Reuters RFA Adapter.....	287
Prerequisites.....	288
Transport configuration parameters.....	290
Manually loading Reuters configurations.....	293
Overriding Default FIDs For TICK/DEPTH generation.....	294
Retrieving Level 2 Data From MARKET PRICE model.....	294
Publish empty Depths/Ticks.....	295
Dictionary path.....	295
RIC translator.....	295
Service monitor injection order.....	296
Event details.....	296
Working with OMM consumer.....	306
Working with OMM Non-Interactive provider.....	307
Working With OMM NEWS.....	308
Working with Machine Readable News.....	310
27 Reuters MAPI Adapter.....	317
Prerequisites.....	318
Transport configuration parameters.....	318
Service monitor injection order.....	321
RFA MAPI event details.....	321
Market Data subscriptions.....	324
Query symbol list.....	324
Custom OMM models.....	325
Working with RFA MAPI.....	326
Working with Fix-MAPI.....	327
Login management.....	329
Order management.....	329
Password management.....	336
Status reporting.....	339

RFA and MAPI-FIX.....	340
28 Trading Technologies Adapter.....	341
Trading Technologies Gateway platform.....	342
Trading Technologies FIX platform.....	351
29 UBS Fx2B FIX Adapter.....	355
Prerequisites.....	356
IAF configuration.....	356
Service monitor extensions.....	356
Service monitor injection order.....	357
Service ID configuration.....	357
Secure Tunnel.....	357
Session configuration.....	357
UBS subscriptions and unsubscriptions.....	358
UBS order management.....	359

About this Guide

- Online Information and Support 12
- Data Protection 13

This guide describes how to configure the Apama Capital Markets Adapters.

Online Information and Support

Software AG Documentation Website

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

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at <https://empower.softwareag.com/>.

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](#).

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at https://empower.softwareag.com/public_directory.asp and give us a call.

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.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

1 Release Notes

■ What's new in 10.3 release	16
■ What's new in 10.2 release	17
■ What's new in 10.1 release	18
■ What's new in 10.0 release	18
■ What's new in 9.12 release	19
■ What's new in 9.10 release	20
■ What's new in 9.9 release	20
■ What's new in 5.3 release	21

Release Notes describes the changes introduced in this release as well as earlier releases.

What's new in 10.3 release

Apama Base FIX Adapter

- `MessageIdentifier`. The transport property `MessageIdentifier` has been added to identify the context of an upstream message so that the message encoding can be performed in parallel.
- **Fix 1:**
 - `DecimalPrecision`. The transport property `DecimalPrecision` has been added to set the desired decimal precision for float fix values.
 - Added new FIX IAF transport configuration parameter `PublishMostRecentUpdate` to always publish the most recent market data updates. This parameter is applicable only for Marketdata provider sessions and also when `MessageIdentifier` IAF property is provided.
- **Fix 2:**
 - `SetNextSeqNumFromLogout`. The transport property `SetNextSeqNumFromLogout` has been added to start the logon with the next expected sequence number detailed in the tag 789.

Bloomberg B-PIPE Adapter

- A new custom service is now available in Bloomberg B-PIPE adapter. This new custom service is an EP based service where you can configure the adapter to connect to a new service offered by Bloomberg. As part of this custom service, the new EPL file `${APAMA_HOME}/adapters/monitors/BPIPE_ReqBasedService.mon` is now available in the service monitor injection order.
- The reference data service will be deprecated in a future release.

Citigroup FIX Adapter

- **Fix 1:**
 - Citi Adapter now supports Colo Spot API Specification v 7.0.16 version.

Note:

This Adapter is tested and certified for Tiered Pricing only.

The following order types are supported:

- Previously Quoted
- Market Order

CME iLink FIX Adapter

- CME iLink FIX adapter is now compatible with “CME Globex API Secure Logon”.

- **Fix 1**
 - CME iLink FIX adapter is now certified with “iLink Convenience Gateways Behavior Harmonizations”.
- **Fix 2**
 - CME iLink FIX Adapter now forwards “the consolidated iLink Fill details” to the user.

CME Simple Binary Encoding Adapter

- Apama CME Simple Binary Encoding Adapter is now certified for MDP3 Price Precision Extension.

Trading Technologies FIX platform

- Trading Technology (TT) FIX adapter is upgraded to version 7.17.x.
- Trading Technology (TT) FIX adapter is now compatible with optimized TT FIX new platform.

What's new in 10.2 release

Apama Base FIX Adapter

- Apama Base FIX adapter supports mapping group elements in `extraParams` of upstream events to repeating group fields in outgoing FIX messages.

Bloomberg FXGO FIX Server Adapter

- Bloomberg FXGO FIX Server adapter supports Bloomberg MiFID-II regulator requirements.

BM+F Bovespa UMDF Adapter

- BM+F Bovespa UMDF adapter now includes solution identification Apama BM+F Bovespa UMDF Adapter 10.x in the logon message.

BM+F FIX Adapter

- BM+F FIX adapter now includes solution identification Apama BM+F FIX Adapter 10.x in the logon message.

CME Simple Binary Encoding Adapter

- CME Simple Binary Encoding adapter now supports configuring `MulticastInterface` for different feeds (primary and secondary).
- CME Simple Binary Encoding adapter now supports extracting requested fields/tag of SBE market updates.

- Added the following transport properties:
 - `PublisherPoolSize`. This property is used to configure the size of the publisher thread pool. The default size is 2.
 - `DecoderPoolSize`. This property is used to configure the size of the decoder thread pool. The default size is 1.

FXall TCPI Server Adapter

- FXall TCPI Server adapter supports FXallTCPI MiFID-II regulator requirements.

Reuters RFA Adapter

- Reuters RFA adapter is now compatible with TREP 3.0.

What's new in 10.1 release

Apama Base FIX Adapter

- Added support for handling order management sessions using connectivity plug-in transport.

BM+F Bovespa UMDF Adapter

- Upgraded adapter to support PUMA 2.0 Derivatives.

EBS Spot Ai FIX Adapter

- Upgraded to EBS Ai FIX 2.0.
 - Support for Continuous Matching orders is removed starting EBS Ai FIX 2.0.
 - eFix (Fixing) Orders are not supported.
 - Support for NDF Fix Outrights and NDF Swaps is added.

Reuters RFA Adapter

- Added support for Reuters Machine Readable News.

What's new in 10.0 release

Apama Base FIX Adapter

- Apama Base FIX adapter supports handling Drop Copy sessions.

- Added the parameter `SubscriptionManager.RemoveSubscriptionOnReject` to handle `MarketDataReject` messages that are received when gateway is unavailable (required for TT-FIX).

Bank of America FXtransact FIX Adapter

- The Bank of America FXtransact FIX adapter has been upgraded to version 9.7.6.7.

CME iLink FIX Adapter

- CME iLink FIX adapter supports for handling Drop Copy sessions.

Currenex FIX Adapter

- Upgraded to Currenex v11.3.

Trading Technologies FIX Adapter

- Trading Technologies FIX adapter supports for handling removing subscription on `MarketDataRequest Reject` due to `GatewayStatus` down.

What's new in 9.12 release

Apama Base FIX Adapter

- Added support for filtering transaction log messages by FIX message type when "FIXLog.ExcludeMsgTypes" is defined.
- Upgraded QuickFIX library to 1.14.3
- Added message helpers to support an audit logging.
- Added Transport configuration parameter `EnableMessageSendAcknowledgements` to forward the acknowledgements of upstream messages.

BV+F Bovespa UMDF Adapter

- Upgraded QuickFIX library to 1.14.3

EBS Spot Ai FIX Adapter

- Added Query interface to retrieve login fields.
- Upgraded to support EBS FIX 4.4

Reuters MAPI Adapter

- Upgraded to MAPI v1.6. Removed Support for Jobbing Orders

- Reuters MAPI adapter uses Trade Capture Report to publish order updates when session configuration parameter "MAPI.OrderUpdateUsesTCR" is set to true.
- Enhancements for handling of MAPI User notification.

What's new in 9.10 release

CME iLink FIX Adapter

- Updated CME iLink adapter to support Front-End Audit Trail Requirements.
- Added transport parameter `ForwardRefMessageonSessionReject` to forward the reference message of Session Reject.

Reuters MAPI Adapter

- Upgraded to MAPI v1.5.5

Reuters RFA Adapter

- Upgraded to RFA Java v8.0.0.L2

Trading Technologies FIX Adapter

- Trading Technologies FIX adapter supports for delaying subscriptions before sending to TT gateway.
- Trading Technologies FIX adapter supports of GatewayStatus request to TT_SecurityStatusSupport monitor.
- Added the parameters `OrderManager.GetSecurityParams` and `SubscriptionManager.GetSecurityParams` to disable lookup of SecurityParams for orders and market data subscriptions.

What's new in 9.9 release

kdb+ Adapter

- Use correlator 'log' instead of Logging Manager/Logging Plugin.

Reuters MAPI Adapter

- Use correlator 'log' instead of Logging Manager/Logging Plugin.

Reuters RFA Adapter

- Use correlator 'log' instead of Logging Manager/Logging Plugin.

What's new in 5.3 release

Apama Base FIX Adapter

- Added support for Mass Quote stream.

Bloomberg B-PIPE Adapter

- Implemented options for configuring MarketDataService.
- Added support for Reference Data Service.

Citigroup SportESP FIX Adapter

- Adapter has been upgraded to Citi version 4.5.

Currenex FIX Adapter

- Upgraded to Currenex v11.0.9.

EBS Spot Ai FIX Adapter

- Adapter is successfully certified with EBS Hedge platform.
- Upgraded to EBS AI 6.7 FIX 1.7.

2 Apama Base FIX Adapter

■ Introduction	24
■ Prerequisites	28
■ FIX service monitors	28
■ IAF FIX adapter	71
■ Connectivity plug-in FIX adapter	81
■ Service monitor injection order	82
■ SSL enabled FIX sessions	84
■ Troubleshooting	85

Introduction

This section provides an introduction to the Apama Base FIX Adapter and its various components.

FIX

FIX (<http://www.fixtradingcommunity.org/>) is the industry standard protocol for the real-time exchange of financial related information and electronic trading. The protocol consists of the following two layers:

- The Session Layer which is concerned with establishing and managing a connection as well as ensuring data integrity, sequencing and addressing
- The Application Layer which is concerned with providing business related services such as market data streaming and order execution

QuickFIX

QuickFIX (<http://www.quickfixengine.org>) is a fully featured, open source FIX engine. It is currently available for Windows, Linux, Solaris, FreeBSD and Mac OS X.

In essence, QuickFIX takes care of the session layer of FIX and provides a convenient abstraction of the application layer over which host applications provide and use business services.

Apama FIX adapter

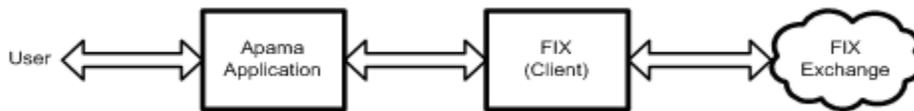
The Apama FIX adapter is a set of components based upon the open source QuickFIX library that allows Apama applications to connect to and communicate with FIX compliant systems. The Apama FIX adapter is compatible with the FIX 4.2-4.4, FIX 5.0, FIX 5.0 SP1, FIX 5.0 SP2 specifications.

The adapter includes the following components:

- The FIX Transport which links with the QuickFIX library to provide a means of establishing a connections to and exchanging messages with FIX systems
- The FIX Service monitors which provide a set of business services to Apama applications wishing to use FIX

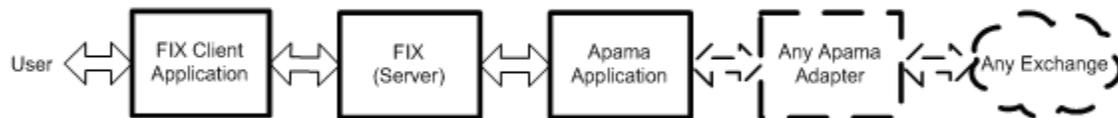
These components can be composed and configured to act as a FIX client, a FIX order server, or a FIX market data server or a FIX Provider. When used as a FIX client, the FIX adapter provides a means for an Apama application to connect to a FIX compliant exchange and request market data and/or execute trades against its available markets. This allows the creation of event based trading applications using the Apama platform. “Figure ” on page 24 shows a high level diagram of this process:

FIX adapter as a client



When used as a FIX order server, the FIX adapter allows FIX compliant systems to connect to and execute trades against Apama applications. This allows, for example an existing FIX client application to send orders to a scenario which could in turn break those orders up into smaller pieces based on some algorithm and submit them to another completely different exchange. When used as a FIX market data server, the FIX adapter allows FIX compliant systems to connect to and issue requests for market data such as Tick and Depth requests against Apama applications, When used as a FIX provider, the FIX adapter provides a means for a liquidity provider to connect to a FIX compliant exchange.

FIX adapter as a server



Supported FIX Features

The following is a list of the FIX features which are currently supported by the FIX adapter:

Feature	Supported	Notes
Heartbeat	Y	Supported in transport/QuickFIX
Logon	Y	Supported in transport/QuickFIX
Test Request	Y	Supported in transport/QuickFIX
Resend Request	Y	Supported in transport/QuickFIX
Reject (session-level)	Y	Supported in transport/QuickFIX
Sequence Reset (Gap Fill)	Y	Supported in transport/QuickFIX
Logout	Y	Supported in transport/QuickFIX
Advertisements	N	
Indications of Interest	N	
News	N	
Email	N	

Feature	Supported	Notes
Quote Request	Y	Restricted (see “FIX_SubscriptionManager” on page 30)
Quote	Y	Restricted (see “FIX_SubscriptionManager” on page 30)
Mass Quote	N	
Quote Cancel	Y	Restricted (see “FIX_SubscriptionManager” on page 30)
Quote Status Request	N	
Quote Acknowledgement	N	
Market Data Request	Y	
Market Data Snapshot / Full Refresh	Y	
Market Data Incremental Refresh	Y	
Market Data Request Reject	Y	
Security Definition Request	Y	Restricted (see “FIX_DataManager” on page 53)
Security Definition	Y	Restricted (see “FIX_DataManager” on page 53)
Security Status Request	N	
Security Status	N	
Trading Session Status Request	Y	
Trading Session Status	Y	
New Order Single	Y	
New Order List	N	
New Order Cross	N	
New Order Multileg	N	
Execution Reports	Y	
Don't Know Trade (DK)	N	
Order Cancel/Replace Request	Y	
Order Cancel Request	Y	

Feature	Supported	Notes
Order Cancel Reject	Y	
Order Status Request	N	
Allocation	N	
Allocation ACK	N	
Settlement Instructions	N	
Bid Request	N	
Bid Response	N	
List Strike Price	N	
List Status	N	
List Execute	N	
List Cancel Request	N	
List Status Request	N	
Business Message Reject	Y	

Preparation Checklist

Before attempting to establish a session with a FIX server/client using the FIX adapter, it is essential that at least the following parameters are configured. Depending on the target exchange and application requirements, further configuration may or may not be required in addition.

Configure transports

When acting as client, a separate transport must be configured for each target FIX server to be connected to. The following parameters must be specified for each transport:

Name	Description
Transport Name	A name for the transport/session.
SocketConnectHost (client only)	The target IP/hostname. (obtained from exchange)
SocketConnectPort	The target port. (obtained from the exchange)
TargetCompID	Identifies the receiving system. (obtained from the exchange)
SenderCompID	Identifies the sending system. (obtained from the exchange)

Name	Description
DataDictionary	The path to the data dictionary file.

Configure sessions

Each transport must be identified and configured for the service monitors. Clients transports are configured using a `com.apama.fix.SessionConfiguration` event and server transports are configured using a `com.apama.fix.ServerConfiguration` or `com.apama.fix.MDServerConfiguration` event. The configuration parameters used will depend upon the target system and application requirements.

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3

FIX service monitors

Client session

When configured as a client, the FIX adapter enables Apama applications to connect to and interact with external FIX servers. Currently, the FIX adapter provides service monitors that provide support in the areas of market data subscription management, order execution as well as session monitoring and management. For a full list of the areas of FIX that the adapter supports see [“Supported FIX Features” on page 25](#).

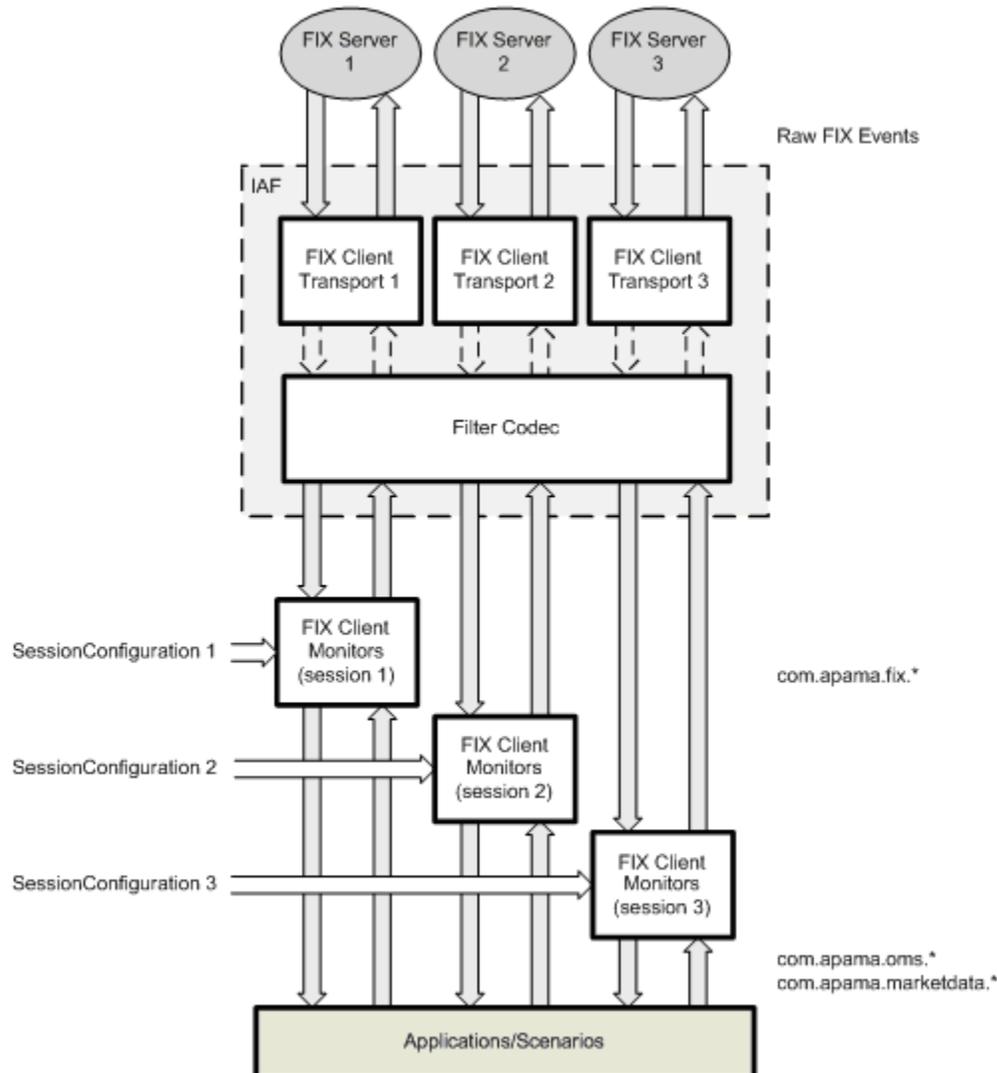
Session configuration

Each service monitor is designed in a way that allows it to be configured differently for the session (that is, transport) it is going to interact with. To configure the FIX service monitors for a particular session, the following event must be sent to the correlator:

```
com.apama.fix.SessionConfiguration {
    string connection;                // FIX transport name
    dictionary<string, string> configuration; //set of configuration parameters
}
```

Each monitor defines its own configuration parameters and sending in this event can be thought of as simultaneously creating a new instance of each service monitor and tailoring its behavior to a particular session. This concept is illustrated below:

FIX adapter as a client



Note, it is not possible to use the FIX service monitors without having first configured them for at least one session.

FIX_SessionManager

The `FIX_SessionManager` provides support for the other monitors by monitoring the state of the FIX sessions which have been configured and sending out notifications when change occur. The following are monitored:

- **Connection Status.** The session manager sends a periodic heartbeat request to the transport and expects a reply within 15 seconds. If no such reply is received then all service monitors are notified of a loss of connection for the session in question.
- **Session Status.** Notifies all service monitors when the transport is logged in or out of the target server.

- **Trading Session Status.** Some exchanges support the notion of trading session status and report this through the corresponding FIX message. The session manager notifies all the service monitors of a change in Trading Session status.

Configuration parameters

Name	Type	Description
FixChannel	String Default: "FIX"	The channel to emit upstream events to. See "Session configuration" on page 28 .
SendSessionStatusRequest	Boolean Default: False	Whether to send a FIX <code>SessionStatusRequest</code> once logged on.
NMDA_FIX_SESSION_NAME	String	Provide the <code>SessionName</code> for the MDA session. This parameter is used to provide the MDA session name value to the FIX adapter to support session management using the CMF Session Manager library.

FIX_SubscriptionManager

The `FIX_SubscriptionManager` provides market data subscription services via the `com.apama.marketdata` interface.

The monitor maintains a set of subscriptions, each of which represents the market for a particular instrument or product on an exchange.

Subscribing/unsubscribing

A subscription is created by issuing a `com.apama.marketdata.SubscribeDepth/Tick` event specifying the following inputs:

Input	Description
Symbol	The instrument (as defined by the exchange)
ServiceId	"FIX"
Market	The session (that is, transport) to use
extraParams	Any extra parameters to be sent with the request (Note, genuine FIX field names will be automatically translated to their integer tag equivalent)

The subscription manager supports market data through the following FIX mechanisms:

- `MarketDataRequest` - A `com.apama.marketdata.SubscribeDepth/Tick` event will be converted into this type of request by default.
- `RequestForQuote` - Can be issued instead by setting the extra parameter `Quote=Y`. (Note, only a continuous stream of replacement quotes and `MassQuotes` is supported currently).

The resultant market data updates (or quote stream events) that are generated as a result of the subscription request are converted into `com.apama.marketdata.Depth` and `com.apama.marketdata.Tick` events and routed. Note, each subscription maintains a reference count which is simply incremented for duplicate requests (same symbol + extra parameters) rather than issuing another request to the exchange.

To stop (and completely remove) a subscription, a `com.apama.marketdata.UnsubscribeDepth/Tick` event must be routed. This will be translated into the equivalent FIX request and sent to the exchange providing the reference count for the subscription is at zero otherwise the count is reduced.

Extra parameters

As discussed in [“Event mappings” on page 78](#), any FIX tags which are received from an exchange but are not mapped as top level fields in the event mapping are automatically copied to extra parameters. Since FIX market data messages are composed of a series of repeating entries, the subscription manager associates each parameter with its side and depth level using the following format:

```
(<SIDE><LEVEL>_<NAME>=<VALUE>)
```

For example:

```
"ASK1_Currency=EUR;ASK1_MDEntryOriginator=Bank3;ASK2_Currency=EUR;
ASK2_MDEntryOriginator=Bank1;BID1_Currency=EUR;
BID1_MDEntryOriginator=Bank3"
```

Note, standard FIX tag numbers will be translated to their field names. Where this is not possible (such as for custom tags), the tag will be output as is. For example:

```
"ASK1_9001=12.0;ASK2_9002=foo"
```

Subscription errors

The FIX adapter does not use the `com.apama.marketdata.DepthSubscriptionError` or `com.apama.marketdata.TickSubscriptionError` events to notify clients of a problem with the data stream. Instead, a `com.apama.marketdata.Depth` and/or `com.apama.marketdata.Tick` is issued with `_ERROR` and `_FAULT` set in the extra parameters.

You can use `"_ERROR"` to get the error message (corresponds to `DepthSubscriptionError.status` or `TickSubscriptionError.status`).

`"_FAULT"` indicates the status of the subscription (corresponds to `DepthSubscriptionError.fault` or `TickSubscriptionError.fault`).

Using `Depth` and `Tick` events instead of `DepthSubscriptionError/TickSubscriptionError` allows multiple subscriptions to the same symbol (with differing extra parameters) to be maintained.

Configuration parameters

Name	Type	Description
FixChannel	String Default: "FIX"	The channel to emit upstream events to. See “Session configuration” on page 28 .
SubscriptionManager. RequireSessionStatusOpen	Boolean Default: False	Whether a FIX session status open message must have been received for the session to be active.
SubscriptionManager. RequireSessionStatusOpen SubscriptionManager. ResubscribeOnConnect	Boolean Default: False	Whether to re-send all subscription requests to the market in the event of a re-connect
SubscriptionManager. ResubscribeOnLogon	Boolean Default: True	Whether to re-send all subscription requests to the market in the event of a re-logon
SubscriptionManager. DistinctDepthTickRequest	Boolean Default: True	Whether to send separate depth and tick subscription requests
SubscriptionManager. MarketDataFullRefresh	Boolean Default: False	Whether to request full refreshes (snapshots) rather than snapshot + updates
SubscriptionManager. MarketDataDepthLevel	Integer Default: 0 (Full Book)	Subscription request depth level
SubscriptionManager. FieldOmissions	String Default: ""	Whitespace delimited set of market data entry fields to omit from the output
SubscriptionManager. ForwardMassQuoteRejectReason	boolean Default: False	Enable this to forward the QuoteRejectReason from the MassQuoteAcknowledgement message as the reject reason.
SubscriptionManager. ClearPricesOnSubscribe	Boolean Default: True	Whether to clear all price data prior to issuing a re-subscription.
SubscriptionManager. MaxDepthLevels	Integer Default: 0 (Full Book)	Output depth level

Name	Type	Description
SubscriptionManager. SecDefRequestTags	String Default: ""	Whitespace delimited set of tag numbers to request from the data manager (see “FIX_DataManager” on page 53 for more details)
SubscriptionManager. Aggregate	Boolean Default: False	Whether to aggregate depth entries with the same side and price.
SubscriptionManager. ReuseRequestID	Boolean Default: True	Whether to reuse the request id when re-subscribing or generate a new one each time.
SubscriptionManager. ReturnZeroPricesOnError	Boolean Default: True	Whether to return a Depth event with all prices set to zero instead of a DepthError.Note: The default value of this parameter will change to False in future releases.
SubscriptionManager. IncludeTimeInRequestID	Boolean Default: False	Specifies that the session startup time should be pre-pended to market data request identifiers.
SubscriptionManager. PriceDivisor	Float Default: 1.0	Specifies a value that prices in the depth events send to applications will be divided by.
SubscriptionManager. RecordLatency	Boolean Default: False	Adds support for latency measurement on marketdata messages, stringified timestamp set dictionary is added to <code>com.apama.marketdata</code> events.
SubscriptionManager. LogLatency	Boolean Default: False	Print marketdata latency. Also means that RecordLatency will be enabled.
SubscriptionManager. IgnoreSnapshotOn Unsubscription	Boolean Default: True	Ignores any trade message received after unsubscription.
SubscriptionManager. SupportZeroQuantities	Boolean Default: False	Overwrites old quantity if the new quantity is zero
SubscriptionManager. MDupdateInPlace	Boolean Default: False	Enables the in-place updating of Market data processing (position based updates are applied in-place).

Name	Type	Description
SubscriptionManager. SuppressZeroQuantities	Boolean Default: False	Suppress zero quantity entries from com.apama.marketdata. Depth event.
SubscriptionManager. MDUpdateActionOrder	Sequence Default: "1 2 0" (that is, Change, Delete, New)	Sequence of MDUpdateAction values separated by spaces. SubscriptionManager specified UpdateAction order will be used to update marketdata from incremental refresh.
SubscriptionManager. PublishNonPositivePrices	Boolean Default: False	Publish zero and negative along with positive prices in Depth.
SubscriptionManager. SubscriptionKeyTags	String Default: ""	Specifies that which parameters or tags should be considered for creating the subscription key for Depth/Tick subscriptions. The value to be given as string with elements separated by spaces, for example "22 207" or "". For more about the tags used with this parameter, see "SubscriptionKeyTag" on page 37.
SubscriptionManager. UseAltSecurityId	Integer Default: ""	Allows users to put alternative security IDs in the "symbol" field of a market data request and have this mapped to the correct tag on the FIX message (and vice-versa for downstream messages). The value of this should be the number of the FIX tag that the alternative security ID will go in. For more information about this parameter, see "UseAltSecurityId note" on page 38.
SubscriptionManager. RepeatingGroupTags	String Default: ""	Repeating groups that are part of outgoing request
SubscriptionManager. UseDefaultTradeEntryTypes	Boolean Default: True	Use the default MDEntry type for Trade subscription
SubscriptionManager. UseDefaultDepthEntryTypes	Boolean Default: True	Use the default MDEntry type for Depth subscription

Name	Type	Description
SubscriptionManager. AdditionalMDEntryTypes	String Default: ""	Additional MDEntryTypes to be sent with Subscribe request
SubscriptionManager. RequireSessionStatusOpen	Boolean Default: False	Requires Session State to be open to make subscriptions
SubscriptionManager. SupportZeroQuantities	Boolean Default: False	Overwrites old quantity, if the new quantity is Zero
SubscriptionManager. SupportNonUniqueMDEntryIDs	Boolean Default: False	Required for exchanges which send same MDEntryID for both BID and Offer Sides
SubscriptionManager. QuoteExpiryTime	Float Default: 60	Provides the Quote Expiry Time
SubscriptionManager. SubscriptionRequestType	Integer Default: "1"	Provides the Update Mechanism for the subscription ("1" is SNAPSHOT_PLUS_UPDATES)
SubscriptionManager. SendQuoteAck	Boolean Default: False	Send Quote Acknowledgment
SubscriptionManager. delayBeforeResubscription	Float Default: 0	Time to wait before sending a subscription request
SubscriptionManager. UpdateDataWithoutEntryId	Boolean Default: False	Custom handling for updating Order book when EntryID is not present
SubscriptionManager. AcceptNonPositivePrices	Boolean Default: False	Required to support Zero or Negative prices
SubscriptionManager. ForwardMDReqID	Boolean Default: False	Forward MDReqID used in subscription along with Depth updates
SubscriptionManager. UseCurrencyFromSymbol	String Default: "BASE"	Extract the Currency information from the Symbol. Accepted Values are "BASE" and "TERM". For more information, see "UseCurrencyFromSymbol note" on page 42.

Name	Type	Description
SubscriptionManager. ValidateDepthSubError OnQuoteCancel	Boolean Default: False	SubscriptionManager sends DepthSubscriptionError with 'fault' as "true" in case of QuoteCancel. This parameter is used in QuoteCancel to send DepthSubscription Error.fault as 'false' which indicates that the subscription may still exist and unsubscribe is necessary.
SubscriptionManager. TagsToForwardFromExchange	String Default: ""	List of tags received from exchange to be forward to the user
SubscriptionManager. ReturnZeroPricesOnError	Boolean Default: True	Return empty depth/tick on subscription error
SubscriptionManager. EnableMassQuote Acknowledgement	Boolean Default: False	Enable Mass Quote Acknowledgment
SubscriptionManager. UnsubscribeBeforeSubscribe	Boolean Default: False	Send an unsubscribe request to the exchange before making a subscribe request
SubscriptionManager. GenerateInstrumentID	Boolean Default: False	Lets the adapter generate an instrumentID
SubscriptionManager. IncludeTimeInRequestID	Boolean Default: False	Include Time in Request ID
SubscriptionManager. IgnoreUnexpectedUpdate	Boolean Default: False	Ignore Unexpected Market Data Update
SubscriptionManager. UnsubscribeOnQuoteCancel	Boolean Default: False	Remove subscription when a QuoteCancel is received form exchange
SubscriptionManager. IgnoreQuoteCancel OnUnsubscription	Boolean Default: False	This parameter will handle the QuoteCancel messages received as the response to the Quote Unsubscription.
SubscriptionManager. ClearAttributesOnSnapshot	sequence	This parameter describes what type of prices/quantities need to be cleared on receiving a snapshot. It may be either 0 or 1 or 2 .

Name	Type	Description
		If you assign value * then BID/OFFER/TRADE/OTHER data is cleared. Defaults to *.
		If you assign value 0 then BID data is cleared.
		If you assign value 1 then OFFER data is cleared.
		If you assign value 2 then TRADE data is cleared.
		Any value other than */0/1/2 then no data will be cleared

When the session configuration parameter `SubscriptionManager.ReturnZeroPricesOnError` is set to true, the adapter will send depth/tick events in place of `DepthSubscriptionError/TickSubscriptionError`. The extraparam of this depth/tick event will have the following values:

- **_ERROR**. The error message (corresponds to `DepthSubscriptionError.status`)
- **_FAULT**. Flag to indicate whether an unsubscribe is necessary or not (corresponds to `DepthSubscriptionError.fault`)

SubscriptionKeyTag

The tags specified in the `SubscriptionKeyTags` configuration parameter are searched in the same order in the `extraParams` field of `com.apama.marketdata.SubscribeDepth()` and `com.apama.marketdata.SubscribeTick()` to create the subscription key. The reference counting and uniqueness of the subscription is based on presence of these tags and values corresponding to these tags. For example, consider the following:

```
SubscriptionManager.SubscriptionKeyTags = "22 207"
com.apama.marketdata.SubscribeDepth("FIX", "Connection", "SYMBOL",
    {"22":"8", "207":"ABCD", "123":"EFG", "234":"IJK"})
```

The key prepared is:

SYMBOL:8:ABCD

For the following,

```
com.apama.marketdata.SubscribeDepth("FIX", "Connection", "SYMBOL",
    {"22":"8", "123":"EFG", "234":"IJK"})
```

The key prepared is:

SYMBOL:8:

The empty value of the configuration parameter indicates that the key is prepared just from the symbol, so the uniqueness is based on just the symbol.

Note:

The `SubscriptionManager.SubscriptionKeyTags` configuration parameter is not supported in session reconfiguration, only the initial provided values are used in succeeding re-configurations. Also this parameter has a dependency on the `SubscriptionManager.DistinctDepthTickRequest` configuration parameter. If both parameters are used in a session configuration event, then for succeeding re-configurations the value of `SubscriptionManager.DistinctDepthTickRequest` should not be altered.

UseAltSecurityId note

You can set the `SubscriptionManager.UseAltSecurityId` and `OrderManager.UseAltSecurityId` configuration parameters to alternate FIX tags; for example:

```
SubscriptionManager.UseAltSecurityId:"10455"
```

In this case a market data request can be sent as:

```
com.apama.marketdata.SubscribeDepth("FIXService", "Transport",
  "AlternateName", {"55":"SYMBOL"})
```

A `NewOrder` can be sent as:

```
com.apama.oms.NewOrder("1040", "AlternateName", 5.031, "SELL",
  "LIMIT", 10, "FIXService", "", "", "Transport", "", "", {"55":"SYMBOL"})
```

After these are processed by the respective monitors, the `SYMBOL` and `AlternateName` get exchanged and `AlternateName` will be assigned to tag 10455.

FIX_OrderManager

The `FIX_OrderManager` provides order execution services via the `com.apama.oms` interface.

Placing an order

A new order is placed by issuing a `com.apama.oms.NewOrder` event. The monitor generates a `FIX clOrdId` and maps this event to a `FIX NewOrderSingle` message and sends it to the specified session. The following inputs are required:

Input	Description
OrderId	The block order id seed
ServiceId	"FIX"
Broker	Unused
Book	Unused
Market	The session (that is, transport) to use

Input	Description
Exchange	Unused
Symbol	The instrument (as defined by the exchange)
Price	The price (should be zero for market orders)
Qty	The quantity to trade
Side	The side 1=BUY, 2=SELL
Type	The order type (as defined by the exchange)
extraParams	Any extra parameters to be sent with the order (Note, genuine FIX field names will be automatically translated to their integer tag equivalent)

Once in market, the monitor listens for execution reports and translates these to corresponding `com.apama.oms.OrderUpdate` events that describe the current state of the order.

Amending or cancelling an order

To amend or cancel an existing order a `com.apama.oms.AmendOrder` or `com.apama.oms.CancelOrder` event must be routed specifying the order id of the original order. The Order Manager will generate a new FIX client order id and issue a `OrderCancel[Replace]Request` referencing the original client order id.

During a amend or cancel request, the order will be in a non-modifiable state and failures will be notified by setting the `OrderChangeRejected` flag of the `com.apama.oms.OrderUpdate` event.

Trade bust

It is possible in FIX for a previous order execution report to be “undone” by issuing a trade bust. The `FIX_OrderManager` caters for this by providing the `bustTime` configuration parameter which indicates that the execution listeners must be kept “alive” for a certain period of time once the order has been completed.

Configuration parameters

Name	Type	Description
<code>FixChannel</code>	String Default: "FIX"	The channel to emit upstream events to. See “Session configuration” on page 28 .
<code>OrderManager. RequireSessionStatusOpen</code>	Boolean Default: False	Whether a FIX session status open message must have been received for the session to be active.

Name	Type	Description
OrderManager. OrderIDServiceName	String Default: "FIX"	The service to request order ids from
bustTime	String Default: "" (0 days)	Either a decimal number days (relative) or a cron format (absolute) time to continue listening for busted trade execution reports after an order has become final.
OrderManager. SecDefRequestTags	String Default: ""	Whitespace delimited set of tag numbers to request from the data manager (see "FIX_DataManager" on page 53 for more details)
OrderManager. CopyExecTypeToOrdStatus	Boolean Default: False	Whether to simply assign the value of ExecType to OrdStatus thereby ensuring they are always identical.
OrderManager. useMillisecond TransactTimes	Boolean Default: False	Whether to use millisecond accuracy on order transact times.
OrderManager. IgnoreStatusOnOrder CancelReject	Boolean Default: False	Specifies that the status of an order will be unchanged by a OrderCancelReject message. The succeeding Amends/Cancel after the rejected message will have their origClorderId set to clorderId of the last successful order.
OrderManager. KillOrdersOnSessionDown	Boolean Default: True	Cancels all order listeners when the session goes down.
OrderManager. RecordLatency	Boolean Default: False	Enables logging of order latency. If this parameter is set to true, the Order Manager will 1) Add a microsecond-accurate timestamp to all outgoing order submissions, amendments and cancellations, and 2) Calculate and log the end-to-end latency of all incoming order executions, assuming that timestamp recording has also been enabled in the transport configuration.
OrderManager. LogLatency	Boolean Default: False	Print Order management latency, also enables OrderManager.RecordLatency

Name	Type	Description
OrderManager. CancelRejectUsesOrigId	Boolean Default: False	Handles Order Cancel Reject with swapped ClOrdID(11) and OrigClOrdID(41) fields.
OrderManager. amendUsesNonRejected ClOrderID	Boolean Default: False	Specifies that order amend/cancel uses last ClOrderID if it is not rejected.
OrderManager. UseCurrencyFromSymbol	String	Populates the currency (Tag 15) from supplied symbol. Allowed values are BASE and TERM. For more information, see "UseCurrencyFromSymbol note" on page 42.
OrderManager. UseAltSecurityId	Integer	Allows users to put alternative security IDs in the "symbol" field of a market data request and have this mapped to the correct tag on the FIX message (and vice-versa for downstream messages). The value of this should be the number of the FIX tag that the alternative security ID will go in. For more information about this parameter, see "UseAltSecurityId note" on page 38.
FIX.TagsToSupress	Sequence Default: "35 52 34 8"	Provide a list of tags that needs to be removed from the payload of events emitted from the correlator in the upstream direction.
OrderManager. GenerateNewStyle OrderIds	Boolean Default: False	Generates the id with the complete current (full length) time for uniqueness.
OrderManager. HandleSuspended ExecType	Boolean Default: False	Required to handle Suspended order ExecType
OrderManager. CustomOrdTypes	String Default: ""	Supports custom Order Types
OrderManager. updateKeyParams OnStateChange	Boolean Default: False	Price/Qty fields of order update will update only on an amend order is accepted

Name	Type	Description
OrderManager. SendExecutionAck	Boolean Default: False	Sends back acknowledgment to Execution Report
OrderManager. CopyClOrdIDTo OrderUpdate	Boolean Default: False	Forwards ClOrderID sent to the exchange in Order Update
OrderManager. waitForMarketOrderId	boolean	By enabling this parameter, adapter will not process Cancel/Amend order request until NewOrder acknowledged by Exchange.
OrderManager. SuppressZeroPriceAndQuantity	boolean Default: False	Set this to "true", for exchanges which doesn't send price/quantity for order updates.
OrderManager. MassCancelUsesMarketId	boolean	Set this to "true", to handle MassOrderCancel based on MarketId.
OrderManager. SetOrderChangeRejected	boolean Default: False	By enabling this parameter ,the OrderUpdate.orderChangeRejected flag is set to true when order is rejected.
OrderManager. UseTransactTimeOfOrder	boolean	Use TransactTime forwarded from order.

UseCurrencyFromSymbol note

Use the configuration parameter "OrderManager.UseCurrencyFromSymbol" (For Orders) and "SubscriptionManager.UseCurrencyFromSymbol" (For MarketData Subscriptions) to specify currency pair symbols, for example:

Symbol => USD/GBP or USDGBP (Symbol expected xxx/yyy or xxxyyy)

BASE currency => USD

TERM currency => GBP

Scenario 1:

In the default behavior (not setting the configuration parameter) you need to give the tag 15 in order requests, without which the order will not be accepted.

Scenario 2:

If you set the configuration parameter and do not specify a value for tag 15 in order requests, the tag will be populated from the symbol as per the configuration.

Scenario 3:

If you set the configuration *and* specify a value for tag 15 in order requests, the supplied tag value will be used.

UseAltSecurityId note

You can set the `SubscriptionManager.UseAltSecurityId` and `OrderManager.UseAltSecurityId` configuration parameters to alternate FIX tags; for example:

```
SubscriptionManager.UseAltSecurityId:"10455"
```

In this case a market data request can be sent as:

```
com.apama.marketdata.SubscribeDepth("FIXService", "Transport",
  "AlternateName",{ "55":"SYMBOL"})
```

A `NewOrder` can be sent as:

```
com.apama.oms.NewOrder("1040","AlternateName",5.031,"SELL",
  "LIMIT",10,"FIXService","", "", "Transport","", "", {"55":"SYMBOL"})
```

After these are processed by the respective monitors, the `SYMBOL` and `AlternateName` get exchanged and `AlternateName` will be assigned to tag 10455.

MultiContext in Order Manager

`OrderManager` now supports `MultiContext` behaviour. To enable the `MultiContext` behaviour you must set the `SessionConfig` parameter `OrderManager.MultiContext` to `true`. You can add/remove contexts dynamically. The user can set up the `MultiContext` behaviour by either using the `MultiContext Helper Utility` or it can send the `SetupContext` request. The creation of the context is the responsibility of the user.

MultiContextAPI

The following events are available:

```
event SetupContext {
  string marketId;
  string requestId;
  context ctx;
}
```

Used to register/initialize a context. This event should be sent to the Main Context. Parameters are as follows:

- **requestId**. A unique identifier for this request. A `SetupContextResponse` will be received with the same id.
- **ctx**. The context user wants to register.

Once the context is registered, the API will respond with a `SetupContextResponse`.

```
event SetupContextResponse {
```

```

string marketId;
string requestId;
context mainCtx;
}

```

A response event that will be received in the newly created context once the context is successfully registered. The `requestId` will be the same as the one passed in the `SetupContext` event. A reference to the main context is sent in this response.

```

event ClearContext {
    string marketId;
    string requestId;
    context ctx;
}

```

Used to unregister/uninitialize a context. This event should be sent to the Main Context. The API will respond with a clear context response.

```

event ClearContextResponse {
    string marketId;
    string requestId;
    string success;
}

```

A response event that will be received in the context that was cleared, once the context has been removed. The parameter `success` will be true only when there are no pending orders in this context.

Example usage:

```

context mainContext;
action onload(){
    mainContext := context.current();
    //create the context
    context c := context("dummy", false);
    spawn setup to c;
}
action setup(){
    string id := "request1";
    //register this context with the API.
    enqueue com.apama.fix.SetupContext(marketId, id, context.current()) to mainContext;
    on com.apama.fix.SetupContextResponse(marketId=marketId, requestId=id){
        //context registered succesfully
        //send orders
    }
}

```

MultiContextHelper

A set of actions user that can use to add/remove contexts. The following actions are available:

```

action init(string marketId, context refContext)

```

Used to register a context with the MutliContext API. You must call this action from the newly created context. The parameters are as follows:

- **marketId**. The transport name.

- **refContext.** Should be the Main Context.

Note:

refContext should be an already existing context, otherwise this context will not be registered. When the context is getting registered, you can send Orders to this context. These orders will be queued and will be sent once the initialization is successful. Upon successful creation of the context, a SetupContextResponse event will be sent to the newly created context.

```
action deInit(string marketId, context refContext)
```

Used to unregister a context. Should be called only after `init`. `refContext` should be the Main Context.

Example usage:

```
context mainContext;
action onload() {
    mainContext := context.current();
    //create a new context
    context c := context("dummy", false);
    spawn setup to c;
}
action setup(){
    //register this context
    (new com.apama.fix.MultiContextHelper).init(marketId, mainContext);
    //send orders
}
```

You can set up the MultiContext behaviour by either sending the SetupContext event or by using the helper utility. The helper utility also enables queuing of orders, that is you do not have to wait for a SetupContextResponse. It can send orders as soon as it calls the `init`.

As part of the multi-context implementation, the order manager design is moved to use the modular interfaces and callbacks so that the multi-context implementation for the support monitors can be easily extended.

Advantages:

- Easily provide multi-context support for the venue specific monitors.
- Performance improvement by replacing “route event” (if applicable) with callbacks.
- User can provide own implementation by overriding the interface actions.
- Performance improvements due to replacement of nested listener by global listeners.
- Can be easily plugged to other module (multi context, support monitor).

Event Interfaces APIs

Package `com.apama.fix`

Order Manager(OM) provides the following interfaces and factory interfaces:

OrderManagerInterface

Provides the callback actions to handle the `com.apama.oms` New/Amend/Cancel events and session reconfiguration.

```
event OrderManagerInterface
{
  // action call back for handle Neworder
  action<com.apama.oms.NewOrder /*newOrder*/, dictionary<integer, float>
    /*timestamps*/ > onNewOrder;
  // action call back for handle AmendOrder
  action<com.apama.oms.AmendOrder /*amendOrder*/ , dictionary<integer, float>
    /*timestamps*/ > onAmendOrder;
  // action call back for handle CancelOrder
  action<com.apama.oms.CancelOrder /cancelOrder*/ , dictionary<integer, float>
    /*timestamps*/ > onCancelOrder;
  // action call back for handle Mass CancelOrder
  action<com.apama.oms.CancelOrder /*massCancel*/ > CancelAllOrders;
  // action call back for Session configuration
  action<SessionConfiguration /*config*/> handleSessionReconfigurations;
  // action to set a callback for order update
  action < action<com.apama.oms.OrderUpdate /*update*/, boolean /*isFinal*/,
    dictionary<integer, float> /*timeStamp*/ > > setOnOrderUpdateCallback;
  //following actions are for internal use only
  action<ContextDictInterface, MultiContextAPI> contextInit;
  action<> reset;
}
```

Description:

```
action<com.apama.oms.NewOrder /*order*/, dictionary<integer, float>
  /*timestamps*/ > onNewOrder : This action will be called when Neworder
  event listener is triggered.
```

Argument :

```
@1 - com.apama.oms.NewOrder event object
@2 - timestamps dictionary
```

```
action<com.apama.oms.AmendOrder /*amendOrder*/, dictionary<integer, float>
  /*timestamps*/ > onAmendOrder : This action will be called when AmendOrder
  event listener is triggered.
```

Argument :

```
@1 - com.apama.oms.AmendOrder event object
@2 - timestamps dictionary
```

```
action<com.apama.oms.CancelOrder /cancelOrder*/, dictionary<integer, float>
  /*timestamps*/ > onCancelOrder : This action will be called when
  CancelOrder event listener is triggered.
```

Argument :

```
@1 - com.apama.oms.CancelOrder event object
@2 - timestamps dictionary
```

```
action<com.apama.oms.CancelOrder /*massCancel*/> CancelAllOrders :
  This action will be called when CancelOrder(for MassCancel)
  event listener is triggered.
```

Argument :

```
@1 - com.apama.oms.CancelOrder event object
```

```
action<SessionConfiguration /*config*/> handleSessionReconfigurations :
```

```
This action is used to handle the session re-configuration.
Argument :
  @1 - SessionConfiguration event object
```

The response for all placed Orders must be updated by setOnOrderUpdateCallback callback.

```
action < action<com.apama.oms.OrderUpdate /*update*/, boolean /*isFinal*/,
dictionary<integer, float> /*timeStamp*/ > > setOnOrderUpdateCallback :
Argument :
  @1 - An update action that will be called whenever an order update is received.
  This action should have the following parameters -
  @1 - com.apama.oms.OrderUpdate event object
  @2 - the Order is finished or not
  @3 - timestamps dictionary
```

See "BaseFIXOrderManager" module in "FIX_OrderManager.mon" file for the default implementation of above call backs. You can provide custom implementation by overriding these actions.

Messagehandleriface

This interface provides a set of callbacks for handling the FIX based (both downstream and upstream) messages.

```
event Messagehandleriface
{
  //send NewOrderSingle message
  action <string /*orderId*/,string /*fixSymbol*/,string /*side*/,
    string /*ordType*/,float /*OrderQty*/,
    dictionary <string, string> /*extraParams*/,
    com.apama.oms.NewOrder/*origOrder*/,
    dictionary<integer,float> /*timeStamp*/> sendNewOrderSingleMessage;
  //send OrderCancelReplaceRequest message
  action <string /*origClOrdId*/,string /*ClOrdId*/,string /*marketOrderId*/ ,
    string /*fixSymbol*/,string /*side*/,string /*ordType*/,float /*OrderQty*/,
    dictionary <string, string> /*extraParams*/,
    com.apama.oms.AmendOrder/*amendOrder*/,
    dictionary<integer,float> /*timeStamp*/> sendOrderCancelReplaceRequest;
  //send OrderCancelReplaceRequest message
  action <string /*origClOrdId*/,string /*clOrdId*/,string /*marketOrderId*/,
    string /*fixSymbol*/, string /*side*/,integer /*quantity*/,
    dictionary <string, string> /*extraParams*/,
    dictionary<integer,float> /*timeStamp*/> sendOrderCancelRequest;
  //send sendQuoteResponse message
  action <string /*orderId*/,string /*fixSymbol*/,string /*side*/,
    string /*ordType*/,float /*OrderQty*/,
    dictionary <string, string> /*extraParams*/,
    com.apama.oms.NewOrder/*origOrder*/,
    dictionary <integer,float> /*timeStamp*/> sendQuoteResponse;
  action <string /*clOrderId*/ >sendOrderMassCancelRequest;
  action <string /*fixId*/,string /*preFixId*/ ,OrderCancelReject
    /*orderCancelReject*/, OrderContainer /*orderContainer*/,
    float /*bustTime*/,dictionary <string, string>
    /*orderRejectedIDs*/ > gotOrderChangeReject;
  action <ExecutionReport /*executionReport*/,OrderContainer
    /*orderContainer*/,float /*bustTime*/,
    dictionary <string, string> /*orderRejectedIDs */ ,
    dictionary<string,OrderContainer>
```

```

/*clOrdIdToOrderContainer*/ > gotExecutionReport;
action < action<com.apama.oms.OrderUpdate, boolean,
    dictionary<integer, float> >,
    action<string /*fixId*/,OrderContainer /*ordContainer*/
    /*finishedOrder*/ > setUpdateCallbacks;
// action call back for Session configuration
action<SessionConfiguration> handleSessionReconfiguration;
// generate FIX Order ID
action <> returns string getFIXOrderID;
action<> reset;
}

```

Description:

```

action <string /*orderId*/,string /*fixSymbol*/,string
/*side*/,string /*ordType*/,float /*OrderQty*/,
dictionary <string, string> /*extraParams*/,
com.apama.oms.NewOrder/*origOrder*/,
dictionary<integer,float> /*timeStamp*/> sendNewOrderSingleMessage;
It is used to create the NewOrderSingle message based on parameters
and send it to exchange. When OM receives the NewOrder request,
it will validate the request then call this interface action for
order handling.

```

Argument :

```

@1 - string /*orderId*/ -> clOrdId
@2 - string /*fixSymbol*/ -> symbol
@3 - string /*side*/ -> order side
@4 - string /*ordType*/ -> Order type
@5 - float /*OrderQty*/ -> Order quantity,
@6 - dictionary <string, string> /*extraParams*/ -> dictionaries
of extraParams of NewOrders
@7 - com.apama.oms.NewOrder/*origOrder*/ -> NewOrder object
@8 - dictionary<integer,float> /*timeStamp*/ -> timestamps

```

```

action <string /*origClOrdId*/,string /*clOrdId*/,string /*marketOrderId*/ ,
string /*fixSymbol*/,string /*side*/,string /*ordType*/,float /*OrderQty*/,
dictionary <string, string> /*extraParams*/,
com.apama.oms.AmendOrder/*amendOrder*/,
dictionary<integer,float> /*timeStamp*/> sendOrderCancelReplaceRequest;
This is used to create the OrderCancelReplaceRequest message based on
parameters and send it to exchange.

```

When OM receives the Amend order request, it will validate the request then call this interface action for handle the AmendOrder.

Argument :

```

@1 - string /*origClOrdId*/ -> original clOrdId
@2 - string /*clOrdId*/ -> clOrdId
@3 - string /*marketOrderId*/ -> market Order Id
@4 - string /*fixSymbol*/ -> symbol
@5 - string /*side*/ -> order side
@6 - string /*ordType*/ -> Order type
@7 - float /*OrderQty*/ -> Order quantity,
@8 - dictionary <string, string> /*extraParams*/ ->
dictionaries of extraParams of Amend order
@9 - com.apama.oms.AmendOrder/*amendOrder*/ -> AmendOrder object
@10 - dictionary<integer,float> /*timeStamp*/ -> timestamps

```

```

action <string /*origClOrdId*/,string /*clOrdId*/,string
/*marketOrderId*/,string /*fixSymbol*/,
string /*side*/,integer /*quantity*/,dictionary <string, string>
/*extraParams*/,dictionary<integer,float> /*timeStamp*/>

```

```
sendOrderCancelRequest;
```

This is used to create the OrderCancelRequest message based on parameters and send it to exchange.

On receiving the Cancel order request, OM validates the request then call this interface action for handling the Cancel order.

Argument :

```
@1 - string /*origClOrdId*/ -> original clOrdId
@2 - string /*ClOrdId*/ -> clOrdId
@3 - string /*marketOrderId*/ -> market Order Id
@4 - string /*fixSymbol*/ -> symbol
@5 - string /*side*/ -> order side
@6 - float /*OrderQty*/ -> Order quantity,
@7 - dictionary <string, string> /*extraParams*/ ->
    dictionaries of extraParams of order
@8 - dictionary<integer,float> /*timeStamp*/ -> timestamps
```

```
action <string /*orderId*/,string /*fixSymbol*/,string
/*side*/,string /*ordType*/,float /*OrderQty*/,
dictionary <string, string> /*extraParams*/,
com.apama.oms.NewOrder/*origOrder*/,
dictionary<integer,float> /*timeStamp*/> sendQuoteResponse;
```

This is used to create the QuoteResponse message based on parameters and send it to exchange.

Argument :

```
@1 - string /*orderId*/ -> clOrdId
@2 - string /*fixSymbol*/ -> symbol
@3 - string /*side*/ -> order side
@4 - string /*ordType*/ -> Order type
@5 - float /*OrderQty*/ -> Order quantity,
@6 - dictionary <string, string> /*extraParams*/ ->
    dictionaries of extraParams of order
@7 - com.apama.oms.NewOrder/*origOrder*/ -> NewOrder object,
@7 - dictionary<integer,float> /*timeStamp*/ -> timestamps.
```

```
action <string /*origClOrdId*/,string /*clOrdId*/,string
/*marketOrderId*/,string /*fixSymbol*/,
string /*side*/,integer /*quantity*/,dictionary <string, string>
/*extraParams*/,dictionary<integer,float> /*timeStamp*/>
sendOrderCancelRequest;
```

This is used to create the OrderCancelRequest message based on parameters and send it to exchange.

On received the Cancel order request,OM validate the request then call this interface action for handling the Cancel order.

Argument :

```
@1 - string /*origClOrdId*/ -> original clOrdId
@2 - string /*ClOrdId*/ -> clOrdId
@3 - string /*marketOrderId*/ -> market Order Id
@4 - string /*fixSymbol*/ -> symbol
@5 - string /*side*/ -> order side
@6 - float /*OrderQty*/ -> Order quantity,
@7 - dictionary <string, string> /*extraParams*/ ->
    dictionaries of extraParams of order
@8 - dictionary<integer,float> /*timeStamp*/ -> timestamps.
```

```
action <string /*clOrderId*/ >sendOrderMassCancelRequest;
```

This is used to create the OrderMassCancelRequest message based on parameters and send it to exchange.

On receiving the Mass Cancel request, OM validates the request then call this interface action for handling the Mass cancel.

Argument :

```
@1 - string /*clOrderId*/ -> clOrdId .
```

```
action<SessionConfiguration /*config*/> handleSessionReconfiguration;
It is used to handled the session re-configuration.
```

Argument :

```
@1 - SessionConfiguration /*config*/ -> SessionConfiguration object
```

```
action <> returns string getFIXOrderID;
```

It is used to generate the FIX order id. By overriding this call back user can provide the custom implementation.

```
action < action<com.apama.oms.OrderUpdate, boolean,
dictionary<integer, float> >,
action<string /*fixId*/,OrderContainer /*ordContainer*/>
/*finishedOrder*/ > setUpdateCallbacks;
Using above callbacks , user should update the OrderUpdate and
finishedOrder details to "BaseFIXOrderManager" module which is
responsible for memory cleaner and/or update the order response
to user.
```

```
action<com.apama.oms.OrderUpdate, boolean, dictionary<integer,
float> /*onOrderUpdateCallback*/ - > Order Update call back.
```

Argument :

```
@1 - com.apama.oms.OrderUpdate /*update*/ -> Order Update object
```

```
@2 - boolean /*isFinal*/ -> is Order finished ?
```

```
@3 - dictionary<integer, float> -> timeStamp.
```

```
action<string /*fixId*/,OrderContainer /*ordContainer*/>
```

```
/*finishedOrder*/ -> finishedOrder
```

Argument :

```
@1 - string /*fixId*/ -> clOrdId
```

```
@2 - OrderContainer /*container*/ -> Order Container list
```

Downstream Message handle callbacks

```
action <string /*fixId*/,string /*preFixId*/, OrderCancelReject
/*orderCancelReject*/, OrderContainer /*orderContainer*/,float
/*bustTime*/,dictionary <string, string> /*orderRejectedIDs*/ >
gotOrderChangeReject;
```

On receiving the OrderChangeReject request, OM calls this interface action to process the Order amend/cancel reject message and update the result by calling onOrderUpdateCallback.

Argument :

```
@1 - string /*fixId*/ -> clOrdId
```

```
@2 - string /*preFixId*/ -> original clOrdId
```

```
@3 - OrderCancelReject /*orderCancelReject*/ -> orderCancelReject object
```

```
@4 - OrderContainer /*orderContainer*/ ->
```

```
OrderContainer (it contains corresponding dictionaries)
```

```
@5 - float /*bustTime*/ -> bustTime
```

```
@6 - dictionary <string, string> /*orderRejectedIDs*/ ->
```

```
previous orderRejectedIDs list
```

```
action <ExecutionReport /*executionReport*/,OrderContainer
/*orderContainer*/,float /*bustTime*/,
dictionary <string, string> /*orderRejectedIDs */ ,
dictionary<string,OrderContainer> /*clOrdIdToOrderContainer*/ >
gotExecutionReport;
```

On receiving the ExecutionReport request, OM calls this interface action to process the execution report message and update the result by calling onOrderUpdateCallback.

Argument :

```

@1 - ExecutionReport /*executionReport*/ -> ExecutionReport object
@2 - OrderContainer /*orderContainer*/ -> OrderContainer
    (it contains corresponding dictionaries)
@3 - float /*bustTime*/ -> bustTime
@4 - dictionary <string, string> /*orderRejectedIDs*/ -> previous
    orderRejectedIDs list
@5 - dictionary<string,OrderContainer> /*clOrdIdToOrderContainer*/ ->
    dictionary for clOrdId to OrderContainer.

```

Factory API's:(package com.apama.fix)

The factory object performs the following steps and returns corresponding interface to the user.

- Create an interface object
- Create the default handler object
- Maps each callbacks to default handler actions.

DefaultMessagehandlerfactory

This factory object takes Messagehandleriface interface as input and maps it to the DefaultMessagehandler.

It initializes the interface, maps each actions to the default implementation and returns an interface object (Messagehandleriface).

```

action create(SessionConfiguration config ,string idNum,
    OrderManagerUtils orderManagerUtils) returns Messagehandleriface.
Argument:
@1 - SessionConfiguration /*config*/ -> session config parameter
@2 - string /*idNum*/ -> which is used during FIX order id generation.
@3 - OrderManagerUtils -> OrderManagerUtils object.

```

BaseFIXOrderManagerFactory

This factory object takes OrderManagerInterface interface as input and maps it to the BaseFIXOrderManager.

It initializes the interface, maps each actions to the default implementation and returns an interface object (OrderManagerInterface).

```

action create(string serviceId,string CONNECTION, SessionConfiguration config,
    OrderManagerUtils orderManagerUtils, Messagehandleriface msgIface )
    returns OrderManagerInterface
Argument:
@1 - string /*serviceId*/ -> serviceId
@2 - string /*CONNECTION*/ -> TransportName
@3 - SessionConfiguration /*config*/ -> session config parameter
@4 - OrderManagerUtils -> Order Manager Utils object.
@5 - Messagehandleriface -> Messagehandleriface interface object.

```

Example(custom implementation for Amend order)

```

monitor OrderManager
{

```

```

constant string MONITOR_NAME := "FIX Order Manager";
string SERVICE_NAME := "FIX";
SessionConfiguration config;
// the connection to use
string CONNECTION;
//the current id
string idNum := "0";
OrderManagerUtils orderManagerUtils;
action onload()
{
  orderManagerUtils.init();
  on all unmatched SessionConfiguration():config {
    integer val := integer.parse(idNum)+1;
    idNum:=val.toString();
    spawn newSession;
  }
}
action newSession {
  CONNECTION:=config.connection;;
  SERVICE_NAME := config.getConfigString(FIXConfigParams.SERVICEID,SERVICE_NAME);
  MessageHandlerInterface mhiface := (new DefaultMessageHandlerFactory).
    create(config, idNum, orderManagerUtils);
  OrderManagerInterface iface := (new BaseFIXOrderManagerFactory).
    create(SERVICE_NAME, CONNECTION, config, orderManagerUtils, mhiface);
  iface.onAmendOrder := customAmendOrderHandler;
  GenericOrderManager genOMImp := new GenericOrderManager;
  genOMImp.init(SERVICE_NAME,CONNECTION,iface, config) ;
}
action customAmendOrderHandler()
{
}
}

```

Floating point quantities

In order to submit a new order or amend an order with floating point quantity values, the following session configuration parameter must be specified:

```
"AddQuantityToExtraParams":"true"
```

NewOrder and AmendOrder Examples :

```

com.apama.oms.NewOrder("order1", "gbp/usd", 1.9857, "BUY",
  "LIMIT", 6000000, "TRANSPORT", "", "", "Connection", "", "",
  {"Quantity":"6000000.45"})
com.apama.oms.AmendOrder("order1","TRANSPORT","gbp/usd",1.9857,"BUY",
  "LIMIT",3000000,{"Quantity":"3000000.34"})

```

For trade report service, additional quantity information can also be found in the `com.apama.oms.OrderUpdate` extraParams.

The keys are as follows:

- Quantity
- QuantityExecuted
- QuantityRemaining

■ LastQuantityExecuted

FIX_DataManager

The `FIX_DataManager` provides a means to retrieve and store standing data about the products that can be traded on an exchange through the `FIX SecurityDefinitionRequest/Response` mechanism.

When used in conjunction with the `Subscription Manager` and `Order Manger`, this can be used to automatically retrieve extra information about a particular instrument prior to submitting a request to the market.

For example, some exchanges require a numerical instrument id to be specified when placing an order rather than a descriptive name of the product. In this case, the tag number of the instrument id would be added to the `OrderManager.SecDefRequestTags` configuration parameter which would cause the `Order Manager` to issue a request for this information to the `Data Manager`. The `Data Manager` would then send a security definition request to the exchange containing all the order parameters and providing it is possible to disambiguate the required product, the data manager will return the instrument id back to the `Order Manager` ready to be inserted into the order itself. Once the data is retrieved, it is cached for future requests.

It will depend on the target exchange whether or not this feature is necessary and possible.

Configuration parameters

Name	Type	Description
<code>FixChannel</code>	String Default: "FIX"	The channel to emit upstream events to. See "Session configuration" on page 28 .
<code>DataManager.SecDefRequestType</code>	String Default: 3 (Request List Securities)	The security request type to be sent to the market (see the FIX specification for more details)
<code>DataManager.SymbolFormationTags</code>	String	Specifies which tags are to be used for symbol normalization. For more information, see "Symbol Normalization" on page 54 .
<code>DataManager.MappedSecDefListenerKey</code>	String	This parameter may be used in conjunction with the <code>DataManager.SymbolFormationTags</code> parameter. For example: "". Empty strings also could be a possible case where in just symbol will be used for creating subscription key. The unique ness is based on just the symbol.

Symbol Normalization

The Symbol Normalization feature enables you to use unique symbol (service monitor created symbol) for multi-leg securities. In the absence of this feature, you must give several tag values for each user request (Subscription, NewOrder, AmendOrder, CancelOrder) and must write several "if" conditions to uniquely identify the response event. This new feature reduces the task by providing a `mapped_symbol` so that all the interaction of the application with the service monitor can then be through this `mapped_symbol`. The service monitors will take care of to and fro translation of security definition and mapped symbol.

Session configuration parameters for symbol normalization

Parameter	Description
DataManager. SymbolFormationTags": " 167 207 [146 308 309 310 313]"	The configuration to create the mapped symbol. This parameter specifies which tag values must be used for constructing the mapped symbol. The mapped symbol will have the tag values in the same sequence as mentioned in the configuration including the position of occurrence of the repeating group.
DataManager. MappedSecDefListenerKey": " "MARKET_DATA"	<p>This parameter specifies which MarketId/Transport name to be used for fetching the security information of the mapped symbol. By default each configured session in the service monitors uses its own MarketId/Connection/TransportName.</p> <p>In dual session MARKET_DATA and TRADE_DATA, MARKET_DATA populates the intermediate cache of the mapped symbol to the security definition and TRADE_DATA session can use cache created by setting the configuration parameter to corresponding MARKET_DATA transport name.</p> <p>In the absence of this parameter setting, user has to send two <code>InstrumentDataRequest()</code>, one with MARKET_DATA transport and other with TRADE_DATA transport to populate the <code>mapped_symbol</code> cache for each session.</p> <p>Note: You must send the <code>InstrumentDataRequest</code> event with MARKET_DATA transport/connection to populate the cache of mapped symbol to security definition before using the cache in other sessions.</p>

Symbol formation

The symbol is formed with the normalized values of tags mentioned in the configuration parameter `DataManager.SymbolFormationTags`. The tags should to be delimited by space. The repeating group

tags can be mentioned in "[]" square brackets. The first tag in the brackets needs to be the repeating group tag. The tags in the "[]" are also space delimited.

From the configuration `DataManager.SymbolFormationTags:"167 207 [146 309 310 313]"`

The base tags : 167 207

Repeating group tags : 309 310 313

For example:

```
(tag string)
167:207:309:310:313-309:310:313-309:310:313
```

If you replace the tag numbers with their corresponding values picked from the security definition, that will become the mapped symbol.

```
MLEG:ICE_IPE:217922:FUT:201103-217951:FUT:201108-217923:FUT:201110
(This security has 3 legs)
```

The mapped symbol contains all the legs of the multi-leg security with configured tag values populated. If you find minimal tags to distinguish each security, then the mapped symbol is recommended.

For example:

```
DataManager.SymbolFormationTags:"207 [146 10456]"
207:10456-10456
The mapped symbol will be CME:CLF1-CLG1 (This security has only two legs)
```

Multi-leg symbol formation

All the above examples are the multi-leg scenarios.

Single leg symbol formation

In single leg securities, though the session configuration mentions the repeating group tags, the security definition received from exchange will not have the repeating group. So that part is removed while constructing the mapped symbol.

For example:

```
167:207:309:310:313-309:310:313-309:310:313
```

If you replace the tag numbers with their corresponding values, that will become the mapped symbol.

```
MLEG:ICE_IPE:217922 (repeating group absent in the security defintion
for single legged security)
```

You must choose minimum base tags such that symbol formed will be unique and simple.

The created mapped symbol will appear in the dictionary field "extraParams" of `InstrumentDataResponse()` as a key value pair.

```
"MAPPED_SYMBOL" : "MLEG:ICE_IPE:217922"
```

Requesting the security definition for multi-leg securities

Following are the request/response events for getting the security definition:

```
package com.apama.fix
event InstrumentDataRequest {
  string marketId;
  string key;
  string symbol;
  dictionary <string,string> extraParams;
  sequence<string> requiredParams;
}
event InstrumentDataResponse {
  string marketId;
  string key;
  boolean success;
  string errorMessage;
  dictionary <string,string> parameters;
}
```

Requesting required information

In the extraParams field of the InstrumentDataRequest, you can give the filtering information so that the matching security definitions are received as InstrumentDataResponse() for each matched security.

Example: To fetch the security definition for the multi-leg instrument whose leg information is as follows

```
309 : 217922 (tag 309 is UnderlyingSecurityID)
309 : 217951
com.apama.fix.InstrumentDataRequest
("MARKET_DATA","40","",{"167":"MLEG","207":"ICE_IPE","146":"
[{"309:\\"217922\\"}, {"309:\\"217951\\"}]}",["55","48","207"])
```

This request will result in unique/multiple InstrumentDataResponse() events. If there are more than one security available with this legs information, then user gets multiple responses. In case of multiple responses the boolean flag "success" of response event is set to false.

Receiving multi-leg information as part of InstrumentDataResponse

The 'parameters' field of the InstrumentDataResponse() is populated based on the tag list mentioned in requiredParams field of the InstrumentDataRequest. The requiredParams sequence field now accepts repeating group as a string.

Example: To receive the repeating field tags 310 311 313, the request has to be

```
com.apama.fix.InstrumentDataRequest
("MARKET_DATA","40","",{"167":"MLEG","207":"ICE_IPE"},
["55","48","207","[146 310 311 313]"])
```

The response would be

```
com.apama.fix.InstrumentDataResponse
("MARKET_DATA", "40", true, "", {"146": "[\\"310\\":\\"FUT\\",\\"311\\":
  \\"IPE e-Brent\\",\\"313\\":\\"201103\\"],
 {"310\\":\\"FUT\\",\\"311\\":\\"IPE e-Brent\\",\\"313\\":\\"201108\\"}],
  "207":\\"ICE_IPE\\", "48":\\"219924\\", "55":\\"
IPE e-Brent\\", "MAPPED_SYMBOL":\\"MLEG:ICE_IPE:_:ICE_IPE:217922:
FUT:201103-ICE_IPE:217951:FUT:201108\\"})
```

Note:

Always mention the tags in `requiredParams` field of the `InstrumentDataRequest()`. The tags mentioned in this parameter are the out going tags in each request. The mapped symbol cache is formed against these tags as Mapped symbol.

Using mapped symbol

Subscription management

```
com.apama.marketdata.SubscribeDepth("FIX_SERVICE",
  "MARKET_DATA", "CME:CFL1-CFG1", {})
com.apama.marketdata.UnsubscribeDepth("FIX_SERVICE",
  "MARKET_DATA", "CME:CFL1-CFG1", {})
com.apama.marketdata.Depth("CME:CFL1-CFG1", [], [], [], [], [], {})
or
com.apama.marketdata.SubscribeTick("FIX_SERVICE", "MARKET_DATA",
  "CME:CFL1-CFG1", {})
com.apama.marketdata.UnsubscribeTick("FIX_SERVICE", "MARKET_DATA",
  "CME:CFL1-CFG1", {})
com.apama.marketdata.Tick("CME:CFL1-CFG1", 123.23, 100000, {})
```

Order management

```
com.apama.oms.NewOrder("19", "CME:CFL1-CFG1", 35350, "BUY",
  "LIMIT", 1, "FIX_SERVICE", "", "", "TT_TRADING", "", "", {})
com.apama.oms.AmendOrder("19", "FIX_SERVICE", "CME:CFL1-CFG1",
  120450, "BUY", "LIMIT", 2000, {})
com.apama.oms.CancelOrder("19", "FIX_SERVICE", {})
com.apama.oms.OrderUpdate("19", "CME:CFL1-CFG1", 35350, "BUY",
  "LIMIT", 1000, true, true, true, false, false, false, "", 1000,
  1000, 1000, 35350, 35350, "Fill:Partial Fill", {})
```

FIX_StatusManager

The `FIX_StatusManager` provides session status notifications to other applications via the `com.apama.statusreport` interface.

For each session the following are monitored:

- Connection Status - Notifies applications of a loss of connection to the IAF as detected by the `FIX_SessionManager`.
- Session Status - Notifies applications when the transport is logged in or out of the target server.
- Trading Session Status - Some exchanges support the notion of trading session status and report this through the corresponding FIX message. The session manager will notify applications of a change in status.

- **Market Data** - Warns applications that no market data has been received for a specified length of time.

Any application wishing to receive the reports must route a `com.apama.statusreport.SubscribeStatus` event specifying the following parameters (blank strings are taken as wildcards):

Input	Description
ServiceId	"FIX"
Object	"Adapter"
Sub_serviceId	The connection (transport) prefix
connection	The connection (transport) name

The sub-service id can be used to subscribe to groups of related connections. For example, if you had the following transports:

```
EXCHANGE1_MARKET_DATA
EXCHANGE1_TRADING
EXCHANGE2_MARKET_DATA
EXCHANGE2_TRADING
```

Subscribe to all status relating to exchange 1 by routing the following subscribe event:

```
SubscribeStaus("FIX", "Adapter", "EXCHANGE1", "")
```

Subscribe to status for all the connections by routing the following:

```
SubscribeStaus("FIX", "Adapter", "", "")
```

Note, the sub-service separator token defaults to “_” but can be overridden at session configuration time (see below).

To stop receiving status events a corresponding `com.apama.statusreport.UnsubscribeStatus` event must be routed.

Configuration parameters

Name	Type	Description	Default
StatusManager. MarketDataTimeout	Float	The period of time in seconds that no market data must have be received before a warning is sent. A value of 0.0 switches the warnings off.	0.0
SubscriptionManager. RequireSession StatusOpen	Boolean	Whether the session requires a FIX session status message to have been received before being available.	false

Name	Type	Description	Default
NMDA_FIX_SESSION_NAME	String	Provide the SessionName for the MDA "" session. This parameter is used to provide the MDA session name value to the FIX adapter to support session management using the CMF Session Manager library.	

FIX_EventViewer

The FIX_EventViewer provides a means to display certain key event types in a readable format in the service monitor log file for debugging purposes. Each field will occupy its own line and where possible enumerated types such as OrdType will be displayed as a string value rather than an integer. Currently, the following events are supported:

- ExecutionReport
- MarketDataSnapshotFullRefresh
- MarketDataIncrementalRefresh
- SecurityStatus
- News

Configuration parameters

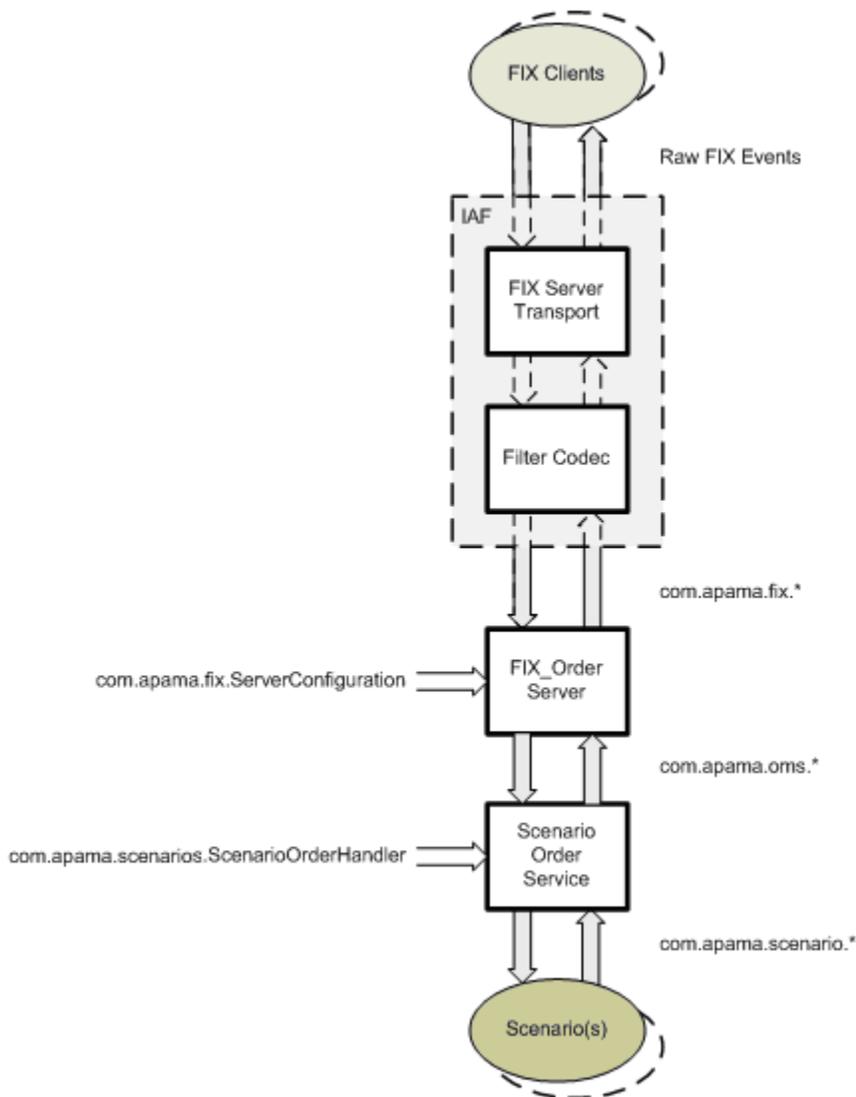
Name	Type	Description	Default
EventViewer.Enable	Boolean	Whether to log the events or not.	False

Server session

When configured as a server, the FIX adapter enables FIX clients to connect to and interact with Apama applications. When used in conjunction with the scenario order service, It is possible for a FIX client to submit an order to Apama which causes a scenario instance to be created and for that scenario instance to do something with the order and report its status back to the original client via FIX execution reports.

Order server

FIX adapter as an order server



In this configuration, the FIX transport is designated as a server (acceptor) and `targetCompIds` are set up for each FIX client.

The `FIX_OrderServer` listens for `FIX NewOrderSingle` messages and forwards these onto the `ScenarioOrderService` which instantiates the specified scenario with the values from the order and listens for updates.

Once an order has been created and its scenario has been instantiated, it is possible for the client to modify or delete the order by sending the appropriate FIX messages.

Configuration properties

Name	Type	Description	Default
CancelFlag	String	The scenario input to set to true upon receiving a CancelOrder message (must be a boolean input)	""

FIX_OrderServer

The `FIX_OrderServer` translates incoming fix messages (new, amend, cancel) into their apama equivalents and reports on order status using FIX execution reports. An order server is configured and instantiated by sending in the following event:

```
com.apama.fix.ServerConfiguration {
  string connection;
  dictionary<string,string> configuration;
}
```

This event ties the order server to a particular server transport (connection) and sets up its runtime behavior (configuration). It is possible to configure multiple servers if there is more than one server transport configured in the IAF.

On receiving a `com.apama.fix.NewOrderSingle` event from its transport, the server creates a `com.apama.fix.ExecutionReport` with a `OrdStatus` of `PENDING_NEW`. It then generates an order id and creates a corresponding `com.apama.oms.NewOrder` event before sending it to the target service. The target service defaults to the scenario order service but can be configured to be something else (for example, a market simulator).

If the original FIX order specifies a target strategy (default FIX tag 847), its identity will be copied into the broker field of the resultant Apama order. This is commonly used to specify a Scenario Order Handler (see [“ScenarioOrderService” on page 64](#)). Strategy parameters (default FIX tag 848) will be merged into the extra parameters of the order.

For example, `NewOrderSingle` should contain the details of the `TargetStrategy` (Tag 847) and `TargetStrategyParameters` (Tag 848). `TargetStrategyParameters` can be provided using the following convention `"848"="Param1:Value1;Param2:Value2; ... ;ParamN:ValueN"`, as shown in the following event sample:

```
com.apama.fix.NewOrderSingle("Server","", "1:0:1e+09","", "1", "ORCL", "1",
  "20010909-02:46:40", 10000, "1", 0, [], {}),
  {"109": "Client_1_ID", "847": "TestBrokerID", "848": "strgy1:100;strgy2:200;strgy3:300"})
```

The above `NewOrderSingle` will be translated to the following:

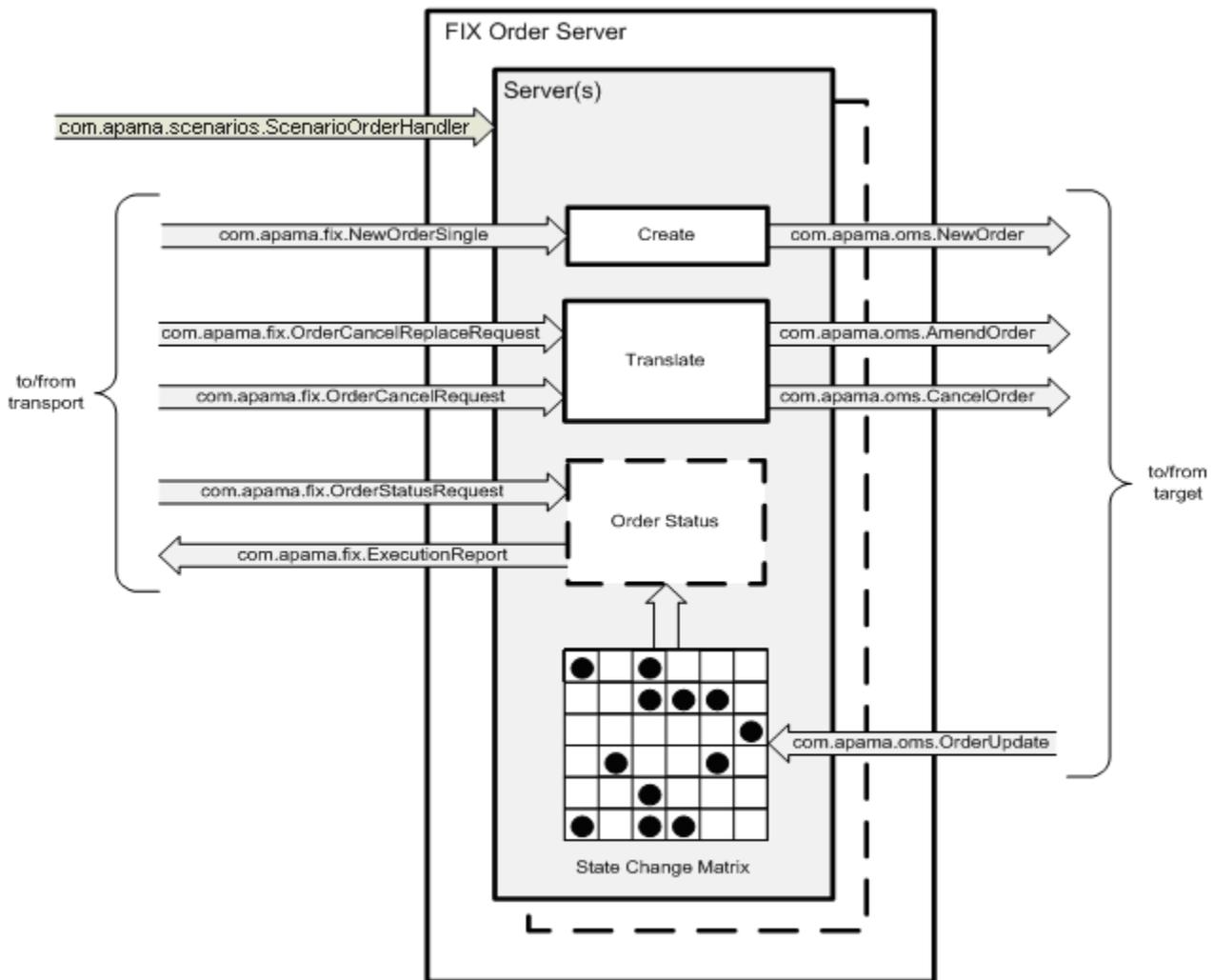
```
com.apama.oms.NewOrder("Server:1e+09:1", "ORCL", 0, "BUY", "MARKET", 10000,
  "MarketSimulator", "TestBrokerID", "", "Server", "", "Client_1_ID",
  {"109": "Client_1_ID", "Account": ""},
  {"__timestamps": "{}", "strgy1": "100", "strgy2": "200", "strgy3": "300"})
```

For each `com.apama.oms.OrderUpdate` event that is received with generated order id, the order state is determined using an implementation of the state change matrix as defined by the FIX specification (see <http://www.fixprotocol.org/specifications>).

Any change in state is reported back to the client as a `com.apama.fix.ExecutionReport`.

Order amendments and cancels are possible and are translated into their Apama equivalents (that is, `com.apama.oms.AmendOrder` or `com.apama.oms.CancelOrder`) and it is possible to request order status by sending in a `com.apama.fix.OrderStatusRequest` message for the order in question.

FIX Order Server



Configuration properties

Name	Type	Description
FixChannel	String Default: FIX	The channel to emit upstream events to. See “Session configuration” on page 28.
TargetService	String Default: MarketSimulator	The identifier of the target service.
ScenarioParameter	string Default: TargetStrategy (847)	The FIX field to extract the scenario identifier from.
OrderServer. UseFirewallServiceID	boolean	When set to true, this property adds Firewall service identifier in the extraparams while routing NewOrder, AmendOrder and CancelOrder.
OrderServer. ForwardRequestID	boolean	Enable this to forward a unique requestID in all orders generated by the fix order server. A uniqueID with key as FIX_SERVER_REQUEST_ID will be forwarded in oms NewOrder, AmendOrder and CancelOrder. The fix order server expects the tag to be returned in the OrderUpdate with key as FIX_SERVER_REQUEST_ID_ACK. This parameter is used by the fix order server to identify if the update was generated for the original order or an amend/cancel request.
OrderServer. RecordLatency	boolean	Adds support for latency measurement on OrderServer.
OrderServer. LogLatency	boolean	Print OrderServer latency, also means that OrderServer.RecordLatency will be enabled.
OrderServer. ValidateSessionParams	boolean Default: False	Set this to "true" to validate incoming OrderUpdates based on SERVICE_NAME and Market.
OrderServer. AckPendingNewState	boolean Default: True	When set to false, OrderServer does not acknowledge Pending New state.

ScenarioOrderService

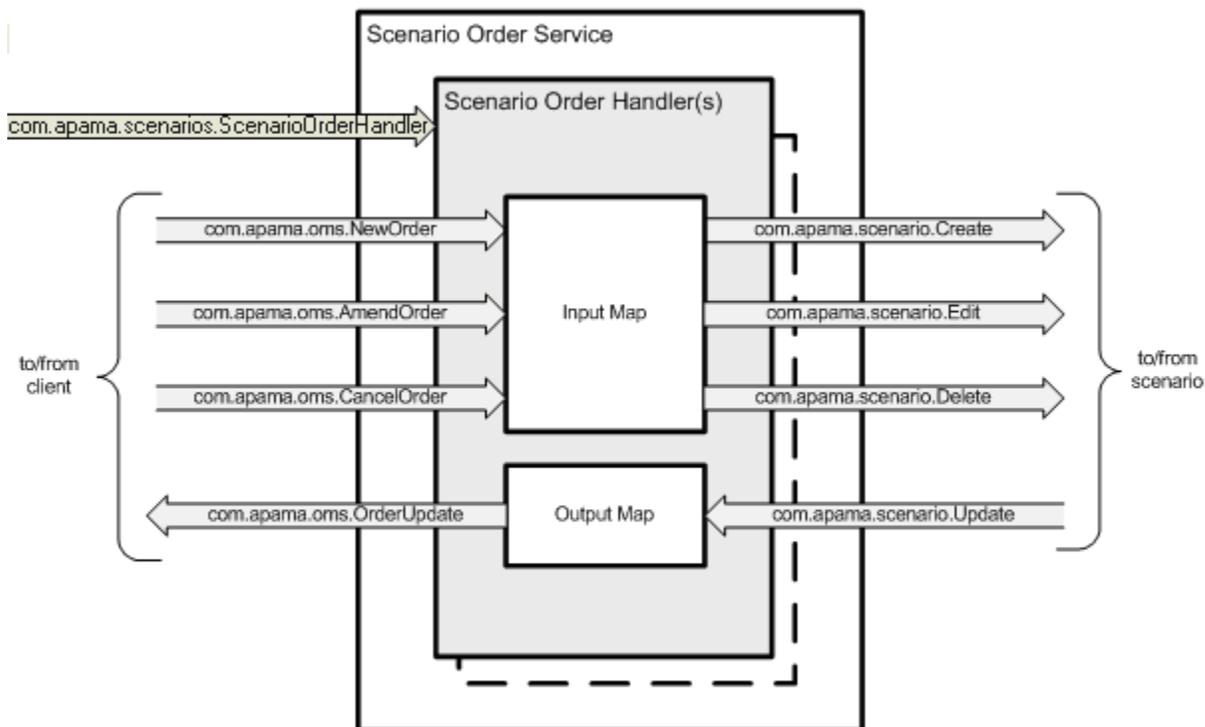
The `ScenarioOrderService` is designed around the notion of a scenario order handler which is an object that translates order events for a particular scenario. The scenario order service can contain many handlers and therefore can manage many scenarios simultaneously. A handler is configured by sending in the following event:

```
com.apama.scenarios.ScenarioOrderHandler {
  string id;
  string scenarioId;
  dictionary<string,string> inputMap;
  dictionary<string,string> outputMap;
  dictionary<string,string> configuration;
}
```

The `id` allows the handler to be referenced, the `scenarioId` references the scenario to be used, the input and output maps define the mapping between orders and scenario instances (see ["Figure " on page 64](#)) and the configuration defines the runtime behavior. Note, the scenario referenced by the scenario id must be loaded at the time this event is received as the input/output maps will be validated against the scenario definition.

When a new order event is received the handler id must be specified in the `brokerId` field. If the broker is not specified or the handler cannot be found the order is rejected.

FIX Order Service



Input and output mappings

The input and output mappings determine how the handler translates `com.apama.oms.*` events to and from `com.apama.scenario.*` events. Each mapping is configured as a set of `<target>:<source>` pairs where `<target>` is the field or extra parameter that will be set in the target event and `<source>` is the source field/parameter or an literal value if preceded with a “#” character.

For example, the following input map sets the scenario input `orderPrice` to be the order field `price`, the scenario input `orderQuantity` to be the order field `qty` and the scenario input `VWAPWindow` to be the literal value “100” upon receiving a new order:

```
{"orderPrice":"price", "orderQuantity":"qty", "VWAPWindow":"#100"}
```

When the scenario issues a `com.apama.scenario.Update` event its outputs are mapped to a `com.apama.oms.OrderUpdate` event based on the output mapping. For example the following output map sets the order update field `qtyExecuted` to be the scenario output `orderQtyExecuted` and sets the order update field `9101` to be the literal value “101”:

```
{"qtyExexecuted":"orderQtyExecuted", "9101":"#101"}
```

The scenario order handler does not distinguish between “real” top level fields and extra parameters. If the named field exists in the target then it will be set, otherwise an extra parameter of that name will be set. The same works in reverse except that if a source parameter or field doesn't exist, it will be set as blank in the target.

In the case of handling `OrderCancel` events, there are two options. The default option is to simply delete the scenario instance. However this may not be useful in many cases as it does not allow the scenario to do anything before exiting or even reject the cancel request. The second option is to nominate a scenario input (using the configuration parameter `CancelFlag`) to be edited to true thereby allowing the scenario to take appropriate action before exiting.

Market data server

The `FIX_MarketDataServer` translates incoming fix market data request messages into their Apama equivalents like `com.apama.mds.SubscribeDepth` and `com.apama.mds.SubscribeTick` events.

A FIX MarketData Server can be configured and instantiated by sending the following event:

```
com.apama.fix.MDServerConfiguration {
    string connection;
    dictionary<string,string> configuration;
}
```

Configuration properties

The following session configuration parameters can be used to configure the FIX market data server.

Name	Type	Description
RequestKeyParams	sequence Default: * (include all extraParams	Specifies the parameters or tags that should be considered for creating the subscription key for Depth/Tick subscriptions. The value is specified as a string with elements separated by spaces. <ul style="list-style-type: none"> ■ An asterisk (*) includes all extraParams of MarketDataRequest. ■ An empty string ("") includes none of the extraParams of MarketDataRequest. ■ "336 625" includes tags 336, 625 and their values only. ■ "* 336 625" includes all except tags 336 and 625.
FixChannel	String Default: FIX	The channel to emit upstream events to. See "Session configuration" on page 28.
TargetService	String Default: MarketSimulator	The ID of the target service.
MDS.RecordLatency	boolean	Adds support for latency measurement on MarketDataServer.
MDS.LogLatency	boolean	Print MarketDataServer latency, also means that MDS.RecordLatency will be enabled.
MDS. CleanMDReqIDOnSessionDown	boolean Default: True	Clean Market Request IDs on session down.
FIX.TagsToSupress	sequence Defaults: "35 52 34 8 "	Provide a list of tags that needs to be removed from the payload of events emitted from the correlator in the upstream direction. .
MultiContextSupport	boolean	To support multi-context. By enabling this parameter, FIX Server handles all requests in multi-context.

Example `com.apama.fix.MDServerConfiguration` event:

```
com.apama.fix.MDServerConfiguration("MarketDataSimulator",
  {"RequestKeyParams": "* 336 625"})
```

Supported features

The FIX_MarketDataServer supports the following features:

- Client authentication
- Tick subscription and unsubscription
- Depth subscription and unsubscription
 - Single snapshot
 - Snapshot + updates (full and incremental)
- MarketData request reject by application
- Configurable subscription key construction parameters

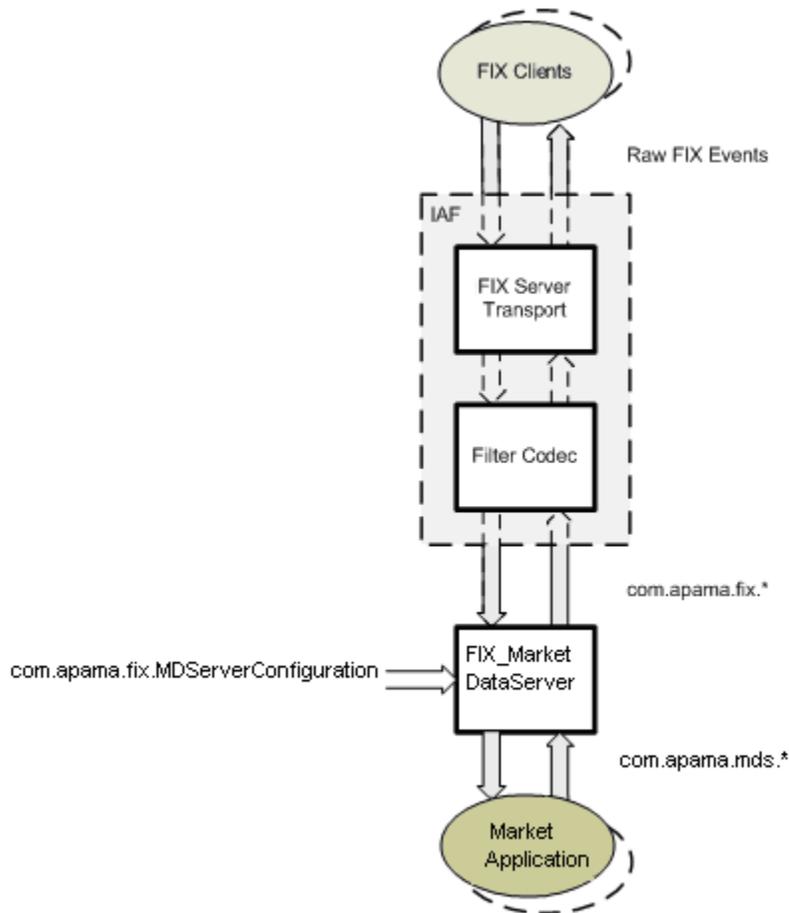
For more information about authentication related configurations, see [“Client authentication” on page 77](#).

MarketData Server uses `com.apama.fix.*` events to interact with the server transport and `com.apama.mds.*` events from `MarketDataServerSupport.mon` to interact with an application.

When the server receives a `com.apama.fix.MarketDataRequest` event from its transport, it creates a `com.apama.mds.SubscribeDepth/com.apama.mds.SubscribeTick` event with a unique `MDReqID` for the specified `TargetService`.

For each `com.apama.mds.MDServerDepth` or `com.apama.mds.MDServerTick` event received from an application for known subscriptions, the server generates a `com.apama.fix.MarketDataSnapshotFullRefresh` and `com.apama.fix.MarketDataIncrementalRefresh` based on the request type.

FIX adapter as a market data server



Creating contexts at startup and dynamically

By enabling `MultiContextSupport` session parameter, MarketData Server processes all MarketData requests in multi context mode. By using multi contexts, the performance improvements can be seen in terms of throughput and decreased latency because the wait time will be reduced.

Context mapped to symbol can be done at startup or dynamically. At Startup, only user(BA) can map symbol to context by `com.apama.mds.MDServerContextRequest` event. If the symbol is already mapped to one context, then MarketData Server will reject that request and acknowledge to user with error message by `com.apama.mds.MDServerContextResponse` event.

If Symbol is not mapped to any context while handling subscriptions, then MarketData Server sends a request to user(BA) for create a new Context by `com.apama.mds.CreateContextRequest` event and expect response by `com.apama.mds.CreateContextResponse` event. If the user doesn't respond to request, then the subscription will handle in the main context only.

Quote server

The `FIX_QuoteServer` translates incoming fix quote request messages into its Apama equivalents like `com.apama.mds.SubscribeDepth` events.

Quote Server can be configured and instantiate by sending the following event:

```
com.apama.fix.QServerConfiguration {
    string connection;
    dictionary<string,string> configuration;
}
```

Configuration parameters

The following session configuration parameters can be used to configure the quote server.

Name	Type	Description
RequestKeyParams	sequence Default: * (include all extraParams	Specifies the parameters or tags that should be considered for creating the subscription key for Depth/Tick subscriptions. The value is specified as a string with elements separated by spaces. <ul style="list-style-type: none"> ■ An asterisk (*) includes all extraParams of MarketDataRequest. ■ An empty string ("") includes none of the extraParams of MarketDataRequest. ■ "336 625" includes tags 336, 625 and their values only. ■ "* 336 625" includes all except tags 336 and 625.
FixChannel	String Default: FIX	The channel to emit upstream events to. See "Session configuration" on page 28 .
TargetService	String Default: MarketSimulator	The ID of the target service.
FIX.TagsToSupress	sequence Defaults: "35 52 34 8 "	Provide a list of tags that needs to be removed from the payload of events emitted from the correlator in the upstream direction. .
MultiContextSupport	boolean	To support multi-context. By enabling this parameter, FIX Server handles all requests in multi-context.
UseSharedSessionManager	boolean Default: False	If vendor is expecting both Quotes and Orders in a single session, you can share Quote session with Order using this parameter.

Example:

```
com.apama.fix.QServerConfiguration("QuoteSimulator",  
{"RequestKeyParams": "* 336 625"})
```

Supported features

- Client authentication
- Quote Request
- Quote Cancel
- Quote Request Reject by Business Application
- Quote Acknowledgement

Quote Server uses `com.apama.fix.*` events to interact with Server Transport and `com.apama.mds.*` events from `MarketDataServerSupport.mon` to interact with the application.

On receiving a `com.apama.fix.QuoteRequest` event from its transport, the server creates `com.apama.mds.SubscribeDepth` with unique `QuoteReqID` for specified `TargetService`.

For each `com.apama.mds.MDServerDepth` received from the application for known subscriptions, it will generate `com.apama.fix.Quote`.

Creating contexts at startup and dynamically

By enabling `MultiContextSupport` session parameter, Quote Server processes all Quote requests in multi context mode. By using multi contexts the performance improvements can be seen in terms of throughput and decreased latency because the wait time will be reduced.

Context mapped to symbol can be done at startup or dynamically. At Startup, only user (BA) can map symbol to context by `com.apama.mds.MDServerContextRequest` event. If the symbol is already mapped to one context, then Quote Server will reject that request and acknowledge to user with error message by `com.apama.mds.MDServerContextResponse` event.

If Symbol is not mapped to any context while handling subscriptions, then Quote Server sends a request to user(BA) for create a new context by `com.apama.mds.CreateContextRequest` event and expect response by `com.apama.mds.CreateContextResponse` event. If the user doesn't respond to request, then the subscription will handle in the main context only.

Drop Copy Session

The Drop Copy service allows users to receive real-time copies of Execution Report and Acknowledgment messages as they are sent over Order entry system sessions.

To enable drop copy session user should inject below monitor.

```
"${APAMA_HOME}/adapters/FIX/monitors/FIX_DropCopyManager.mon".
```

To initiate the session, you must configure with below event

```
com.apama.fix.DropCopyConfiguration(<Transport_Name>,  
{"SERVICEID":<value>,"FixChannel":<value>})
```

Sample configuration

```
com.apama.fix.DropCopyConfiguration
("connection", {"SERVICEID":"DROP-COPY","FixChannel":"FIX_DROPCOPY"})
```

IAF FIX adapter

FIX MDA Transport

The FIX IAF transport links with the QuickFIX library to provide connectivity, session management, and event handling to and from a FIX system. It is possible to run several transports within a single IAF process and thereby run several FIX clients and/or servers simultaneously.

The following IAF Transport configuration properties have been added in 5.0 to support interaction of the FIX adapter with MDA, as well as to provide various customization to the FIX adapter:

- **IAF_CHANNEL.** As the MDA sources are self-registering components, you must provide the IAF channel as a Transport property to make the correlator components aware of the adapter's presence.
- **CorrelatorHBTimeout.** The timeout interval after which the connection to the correlator can be considered as stale.
- **AdapterConfiguration.** The adapter configuration details required by the FIX adapter while registering a source with MDA. This is used to provide the list capabilities supported by that particular FIX session, and the details about the module providing that capability. For the supported configuration options, refer to the file `FIXTransport-BaseFIXLibrary.xml` that is shipped with the adapter.

Transport properties

Each transport can be configured through a number of properties as outlined below:

QuickFIX properties

The transport has been designed so that it is possible to configure the QuickFIX engine via the IAF configuration file by prefixing a transport property with "QuickFIX". For example, the following transport property sets the target host of its embedded QuickFIX engine:

```
<property name="QuickFIX.SocketConnectHost" value="server1.abc.com"/>
```

Any QuickFIX property can be set in this way, a full list of which along with the rules for their use can be found at <http://www.quickfixengine.org/quickfix/doc/html/configuration.html>.

General transport properties

- **Static fields**

Allows static values to be defined for outgoing (upstream) messages. The definition specifies whether the field is inserted into the header or body, the message type and the tag number. Wildcards (“*”) can be specified for the message type and tag number.

```
StaticField.[header|body].[msg].[tag]
```

Examples:

```
<property name="StaticField.body.D.1" value="foo"/>
<property name="StaticField.header.*.50" value="bar"/>
```

The first example sets body tag 1 in msg type D (`NewOrderSingle`) to the value `foo` whereas the second example sets header tag 50 in all messages to the value `bar`.

Specific message types have precedence over the wildcard, and existing fields in the message will not be over-ridden. Note that if there are two `StaticField` entries which update the same tag and message type, the second is ignored.

■ Header field maps

Maps a header field from an incoming (downstream) FIX message to the payload (`extraParams`) of the event sent to the correlator. The name of the generated payload field will be of the form `Header:<tag>`.

```
MapHeaderField.[msgType].[tag]
```

Examples:

```
<property name="MapHeaderField" value="D.52"/>
<property name="MapHeaderField" value="*.50"/>
```

In upstream messages, you can add this information along with extra parameters, which can be useful when you want to place a tag whose value is dynamic in the header part of the message.

To include a tag in the header part of outgoing (upstream) FIX messages, provide `“Header:<tag>”:<value>”` in the `extraParams` of the event sent to the correlator.

■ Fatal reject messages

Defines a string that will cause the session to be terminated upon receiving a session level reject containing it.

Example:

```
<property name="FatalRejectMessage" value="Range of messages to
  resend is greater than maximum allowed [2500]."/>
```

In this case, the session will be terminated if the sequence number gap exceeds 2500.

■ Controlling FIX message replay

The FIX session protocol maintains a unique (within a session) sequence numbering of messages transferred in both directions between the FIX client and server. By default, when a connection is established both the client and server will resend any messages from the same FIX session that have not yet been acknowledged by the other party. This means that all messages sent by

either party will be processed even if the session is unexpectedly disconnected (e.g. by a network failure) and later resumed.

However, it is sometimes not desirable for all “old” messages to be processed when a session is resumed. For example, orders submitted immediately before a disconnection but only processed when the session resumes might trade against stale prices. In the event that the FIX server does not automatically reject such stale orders, the FIX transport can be configured not to resend FIX New Order Single messages when the session is established. If the `SendNewOrdersOnInitialResend` transport property is set to true (the default), all unacknowledged messages will potentially be replayed by the transport when the session is established. If this property is set to false, unacknowledged New Order Single messages will not be replayed. All other messages types will still be replayed if requested by the server.

■ Logging FIX messages

By default, the FIX transport logs all incoming and outgoing FIX messages at DEBUG level. The `LogFixMsgAtInfoLevel` transport property can be used to control this behavior. If the value of this property is false (the default), the transport will log all messages at DEBUG level as usual. If this property is set to true, incoming and outgoing FIX messages will be logged at INFO level instead.

■ Calculating FIX adapter latency

The FIX transport supports the IAF adapter latency measuring framework introduced in Apama 4.0. Please see the IAF documentation for details of how to configure this feature in general.

If timestamp recording is enabled in the FIX transport, timestamps will be attached to FIX New Order Single, Order Cancel/Replace Request and Execution Report messages, and made available to the adapter service monitors on downstream messages.

If timestamp logging is enabled, the FIX transport will log upstream, downstream and round-trip latency for incoming and outgoing messages. In addition, the service monitors can be configured to add timestamps to outgoing orders and amend/cancel requests, allowing the round-trip latency for an order and subsequent acknowledgment or execution to be calculated. See the details of the `OrderManager.LogLatency` service monitor parameter for more information on this feature.

Transport properties configuration

Property	Description
<code>QuickFIX.FileStorePath</code>	<p>This parameter controls logging of persisted FIX messages (used for replay and gap recovery). This property must be specified to a valid path. This property accepts relative as well as absolute paths.</p> <p>For example</p> <ul style="list-style-type: none"> ■ Windows(Absolute):- D:\XYZ\FIX_Log. Directory FIX_Log will be created if it does not exist.

Property	Description
	<ul style="list-style-type: none"> Windows(Relative):- Fix_Log. Directory FIX_Log will be created in current directory from where the IAF instance started.
QuickFIX.FileLogPath	This parameter controls logging of all incoming and outgoing FIX messages. If this property is not mentioned, the logging is disabled. This property accepts relative as well as absolute paths.
QuickFIX.SendBufferSize QuickFIX.ReceiveBufferSize	<p>These properties can be used to set the QuickFIX TCP socket send/receive buffer sizes to the number of bytes specified.</p> <p>It defaults to OS specific.</p> <p>For example</p> <pre data-bbox="743 785 1360 911"><property name="QuickFIX.SendBufferSize" value="1024" /> <property name="QuickFIX.ReceiveBufferSize" value="1024" /></pre>
ConvertNativeStrings (boolean, default false)	If true, any STRING-type fields in FIX messages should be converted from the native character encoding UTF-8 for downstream messages and from UTF-8, the native encoding for upstream messages. If false, all strings are assumed to be UTF-8.
NativeEncoding (string, default empty)	<p>If non-empty, this is the name of the native encoding to use for conversions to and from UTF-8. If not specified, the system encoding should be used - this parameter is to override it per-transport.</p> <p>For example</p> <pre data-bbox="743 1388 1360 1514"><property name="ConvertNativeStrings" value="true" /> <property name="NativeEncoding" value="IS08859-1" /></pre>
ExcludeTagsFromMessage (string, default empty)	<p>This property is used to filter field tags from a given message that needs to be sent to exchange. Field tag be part of header/body/trailer. This property applies to upstream messages. Also field tag cannot be a group tag or tag in group. The format is Message type followed by field tag separated with a space.</p> <p>For example, BE 43 97 50 553 554.</p>

Property	Description
	<p>The above message specifies that from message type 'BE', the field tags 43, 97, 50, 553, 554 must be removed from the message.</p> <p>Each message group should be separated from another message group with a comma</p> <p>D 11 55, BE 43 97 50 553 554</p> <pre data-bbox="841 527 1458 625"><property name="ExcludeTagsFromMessage" value="D 11 55, BE 43 97 50 553 554" /></pre>
MaxSessionLogonAttempts	<p>Configures the maximum number of session logon attempts. For unlimited attempts, you can configure as -1. Defaults to -1.</p> <pre data-bbox="841 770 1458 835"><property name="MaxSessionLogonAttempts" value="5" /></pre> <p>Once the session reaches maximum limit, you will be notified using com.apama.fix.NotifyUser event followed by stopping connection. You have to retry by sending com.apama.fix.doLogin event.</p>
ValidateLogonRequests (boolean)	<p>Enable it to authenticate clients in server scenarios. (valid only if QuickFIX.ConnectionType = acceptor). Defaults to false.</p>
LogonValidationTimeout (in milliseconds)	<p>Configure a timeout for client connection validation. Considered only if ValidateLogonRequests is enabled. Defaults to 2000.</p>
StopAcceptorOnWatchdogTimeout	<p>Prevents Acceptor connection from stopping if connection is lost between correlator and IAF. Default is true. For example,</p> <pre data-bbox="841 1446 1458 1545"><property name="StopAcceptorOnWatchdogTimeout" value="false" /></pre>
FIXLog.ExcludeMsgTypes	<p>Enable it to exclude writing fix messages to the transaction log on the basis of Message types or message groups. It accepts comma separated values of message types/groups.</p> <p>Supported groups:</p>

Property	Description
	<ul style="list-style-type: none"> ■ ADMIN. Contains {HeartBeat, TestRequest, ResendRequest, SequenceReset, Logon, Logout, UserRequest, UserResponse} ■ MDRESPONSE. Contains {MarketDataSnapshotFullRefresh, MarketDataIncrementalRefresh, MarketDataRequestReject} ■ MDREQUEST. Contains {MarketDataRequest} ■ QUOTEREQUEST. Contains {QuoteRequest, RFQRequest} ■ QUOTERESPONSE. Contains {Quote, QuoteCancel, QuoteRequestReject, QuoteResponse, MassQuote, MassQuoteAcknowledgement} ■ ORDERREQUEST. Contains {NewOrderSingle, OrderCancelReplaceRequest, OrderCancelRequest} ■ ORDERRESPONSE. Contains {ExecutionReport, OrderCancelReject} ■ REJECT. Contains {Reject, BusinessMessageReject} <p>Examples</p> <pre data-bbox="743 1262 1360 1444" style="background-color: #f0f0f0; padding: 5px;"> <property name="FIXLog.ExcludeMsgTypes" value="A,REJECT" /> <property name="FIXLog.ExcludeMsgTypes" value="W,X" /> <property name="FIXLog.ExcludeMsgTypes" value="MDRESPONSE,ORDERREQUEST" /> </pre>
ForwardRefMessageonSessionReject	Enable it to forward the rejected message details on SessionReject. The default value is false.
MessageIdentifier	<p>Identifies the context of an upstream message so that the message encoding can be performed in parallel and sent to the counter party, thus improving the overall performance of the adapter. This parameter is valid only if QuickFIX.ConnectionType = acceptor.</p> <p>For example:</p>

Property	Description
	<pre><property name="MessageIdentifier" value="262"/></pre> <p>Where, 262 is the FIX Tag MDReqID.</p>
PopulateLastMsgSeqNumProcessed	<p>Enable it to set the last message sequence number processed 369 tag in the logon message 35=A.</p> <p>For example:</p> <pre><property name="PopulateLastMsgSeqNumProcessed" value="Y"/></pre>
DecimalPrecision	<p>Enable it to set the desired decimal precision for float fix values. For example:</p> <pre><property name="DecimalPrecision" value="8" /></pre> <p>If you are using connectivity plug-in, add this property in FixMessageCodec.</p>
SetNextSeqNumFromLogout	<p>Enable it to start the logon with the next expected sequence number detailed in the tag 789 in the last logout message from CME. For example:</p> <pre><property name="SetNextSeqNumFromLogout" value="true"/></pre>

Latency measurement

The transport supports the Apama 4.x adapter latency measurement framework. See "The IAF configuration File" section of the "Developing Adapters" book in the Apama 4.x documentation set for full details on configuring the transport to record high-resolution timestamps on downstream events.

To enable timestamp logging and timestamp propagation to marketdata events, you must pass the SessionConfiguration parameters:

```
"SubscriptionManager.LogLatency":"true"
"OrderManager.LogLatency":"true"
```

Client authentication

When the FIX adapter configured as an order server or market data server, it allows the Apama application to authenticate client sessions. To enable client authentication, set the `ValidateLogonRequests` property in `<transport>` section of the adapter's configuration file to `true` and create an application to validate client sessions by listening for `com.apama.fix.Logon` request(s) and replying with `com.apama.fix.LogonAuthenticationStatus` events.

When the `ValidateLogonRequests` property is enabled for a server transport, for each logon request from a client it will wait for up to the number of milliseconds specified by the `LogonValidationTimeout` property to receive a `LogonAuthenticationStatus` event from the Apama application. On timeout it rejects logon requests with reason "Logon validation timeout".

For example, to enable client authentication, the relevant section of the configuration file might look like this:

```
<transport name="ABC_MARKET_DATA" library="FIXTransport">
  <!-- Validate Logon requests, defaults to false -->
  <property name="ValidateLogonRequests" value="true" />
  <!-- Logon validation timeout in milliseconds, defaults to 2000 -->
  <property name="LogonValidationTimeout" value="4000" />
  ...
</transport>
```

FIX order management transport

Client transports

For a transport to act as a FIX client the QuickFIX parameter `ConnectionType` must be set to `initiator`. For example, the following FIX transport is configured to connect to `server1.abc.com` on port `10606` with a `senderCompId` of `CLIENT1`:

```
<transport name="ABC_MARKET_DATA" library="FIXTransport">
  <property name="QuickFIX.ConnectionType" value="initiator" />
  <property name="QuickFIX.SocketConnectHost" value="server1.abc.com"/>
  <property name="QuickFIX.SocketConnectPort" value="10606"/>
  <property name="QuickFIX.SenderCompId" value="CLIENT1" />
  <property name="QuickFIX.TargetCompId" value="SERVER" />
  <property name="QuickFIX.ReconnectInterval" value="60" />
  ...
</transport>
```

When acting as a client, the transport creates and maintains a single FIX session to an external server.

Event mappings

The IAF configuration files distributed with the FIX adapter come with a standard set of event mappings that include all the messages defined in the FIX 4.2 - 4.4 specifications. Each message maps to a distinct Apama event in the `com.apama.fix` package.

Only fields that are either mandatory in the FIX specification or extremely common are mapped as top level fields. All other fields are mapped into the payload. This also works in reverse in that upstream (outgoing) events can set fields other than those defined as top levels fields by adding them to the payload.

Event routing

In addition to the message fields themselves, each event has two special fields which are used for message routing:

- **TransportName.** Indicates which transport the event has originated from (incoming events) or intended for (outgoing events).
- **Session** indicates which session the event has originated from (incoming events) or intended for (outgoing events). For a client transport, this field will always be blank as there is only one session. For a server transport this field indicates the originating or intended client.

Repeating groups

Many FIX messages contain repeating groups of fields which must be mapped to Apama sequences. Each repeating group has a corresponding sub-event defined in the event mapping.

For example, the following event mapping for `MarketDataIncrementalRefresh` defines a repeating group (`NoMDEntries`) which is defined as a sequence of `MarketDataIncrementalRefresh_MDEntry` events:

```
<event name="MarketDataIncrementalRefresh" encoder="FilterCodec"
  direction="both" copyUnmappedToPayload="true" inject="true"
  package="com.apama.fix">
  <id-rules>
    <downstream>
      <id fields="35" test="==" value="X"/>
    </downstream>
    <upstream/>
  </id-rules>
  <mapping-rules>
    <map apama="TRANSPORT" type="string" default="" transport="transportName"/>
    <map apama="SESSION" type="string" default="" transport="SESSION"/>
    <map apama="MDReqID" type="string" default="" transport="262"/>
    <map apama="NoMDEntries" type="reference" referencetype="sequence
      &lt;MarketDataIncrementalRefresh_MDEntry&gt;" default="[]"
      transport="268"/>
    <map type="string" default="X" transport="35"/>
  </mapping-rules>
</event>
<event name="MarketDataIncrementalRefresh_MDEntry" encoder="FilterCodec"
  direction="both" copyUnmappedToPayload="false" inject="true"
  package="com.apama.fix">
  <id-rules>
    <downstream>
      <id fields="35" test="==" value="_MarketDataIncrementalRefresh_MDEntry"/>
    </downstream>
    <upstream/>
  </id-rules>
  <mapping-rules>
    <map apama="MDUpdateAction" type="string" default="" transport="279"/>
    <map apama="MDEntryType" type="string" default="" transport="269"/>
    <map apama="MDEntryID" type="string" default="" transport="278"/>
    <map apama="Symbol" type="string" default="" transport="55"/>
    <map apama="MDEntryPx" type="float" default="0.0" transport="270"/>
    <map apama="MDEntrySize" type="float" default="0.0" transport="271"/>
    <map apama="OrderID" type="string" default="" transport="37"/>
    <map apama="_extraParams" type="reference" referencetype="dictionary
      &lt;integer, string&gt;" default="{}/>
    <map type="string" default="_MarketDataIncrementalRefresh_MDEntry"
      transport="35"/>
  </mapping-rules>
</event>
```

```
</event>
```

Currently, the IAF Normalised Event does not provide a means of representing repeating groups of fields so these sequences must be encoded as strings when passed to or from the transport.

In order for the transport to be able to encode/decode repeating groups it is necessary to specify the structure of each sub-event as a transport property. For example, the following property defines the structure of the `MarketDataIncrementalRefresh_MDEntry` event:

```
<property name="X:268"
  value="com.apama.fix.MarketDataIncrementalRefresh_MDEntry{
  string 279; string 269; string 278; string 55; float 270; float 271;
  string 37; }" />
```

The IAF configuration files distributed with the FIX adapter define properties for all sub-events in the FIX 4.2 specification and a subset of the most useful sub-events in the FIX 4.3 and 4.4 specifications. It is usually not necessary to modify these in any way. However, these properties must be included in any transport that is added. The distributed template configuration files demonstrate the XML "XInclude" syntax used to include the standard repeating group definitions into a FIX adapter configuration file.

For convenience, FIX adapter ships `FIX-static-common.xml` and `FIX-static-groups.xml` with mappings required for all standard FIX message types.

Filter Codec

The filter codec provides a flexible way of removing fields from events upon certain conditions. This is useful in FIX where there are certain rules pertaining to event structure in some situations. For example, when sending a market order to a trading system it may be a requirement that no price field is set. With the filter codec it is possible to say remove the price field if it is zero.

Each property to the filter codec that begins with `filter.` defines a new filter and takes the form:

```
<property name="filter.<direction>.<field>" value="<value>"/>
```

If no `<field>` is specified then the filter will be applied globally to all fields and likewise for `<direction>`. The `<value>` is the field value that must be used to invoke the filter.

For example, the following configuration removes any field with an empty value in both directions and removes field 44 if its value is "0" in the upstream direction.

```
<codec name="FilterCodec" library="FilterCodec">
  <property name="removeTransportField" value="false"/>
  <property name="TransportFieldName" value="transportName"/>
  <property name="filter" value=""/>
  <property name="filter.upstream.44" value="0"/>
</codec>
```

For convenience, FIX adapter ships `FIX-static-codecs.xml` which defines the standard codec configuration.

FIX legacy market data transport

Configuration for the FIX legacy market data transport is identical to the FIX Order Management transport.

Server transports

For a transport to act as a FIX server the `ConnectionType` must be set to `acceptor`. It is then necessary to create a `TargetCompID` for each client that wishes to connect to the server. For example, the following transport is set up for two clients, `CLIENT1` and `CLIENT2` and is listening for connections on port 10602:

```
<transport name="ABC_MARKET_DATA" library="FIXTransport">
  <property name="QuickFIX.ConnectionType" value="acceptor" />
  <property name="QuickFIX.SocketConnectPort" value="10602"/>
  <property name="QuickFIX.SenderCompId" value="SERVER" />
  ...
  <!--clients -->
  <property name="QuickFIX.TargetCompId" value="CLIENT1" />
  <property name="QuickFIX.TargetCompId" value="CLIENT2" />
</transport>
```

When acting as a server, the transport creates a server capable of maintaining several external client sessions.

Connectivity plug-in FIX adapter

FIX order management

You can use connectivity plug-in transport for handling order management sessions. The FIX adapter connectivity chain uses new connectivity plug-in architecture to connect to FIX compliant systems. For more information on connectivity plug-in, see Apama documentation.

Connectivity configuration

The YAML file `FIX-OM-Connectivity.yaml` describes the configuration of chain and its components. It also contains place holders for the necessary configuration parameters to connect FIX adapter engine/service. Connection-related configuration is specified in the `fixSession` stanza of the `FixConnectivityTransport` instance. You must update the `fixSession` stanza with the relevant values.

Session configuration

The FIX trading sessions must be configured with the following configuration parameters:

- `FixChannel = FIX_OM_Connectivity.`
- `CONNECTIVITY_PLUGIN = "true"`

For example:

```
com.apama.fix.SessionConfiguration("FIX_OM_Connectivity",  
    {"FixChannel":"FIX_OM_Connectivity","CONNECTIVITY_PLUGIN":"true"})
```

Service monitor injection order

FIX MDA session

This section lists the service monitor injection order when establishing a FIX session using CMF MDA.

1. `${APAMA_HOME}/monitors/StatusSupport.mon`
2. `${APAMA_FOUNDATION_HOME}/ASB/projects/general_infrastructure/Foundation Utilities/Foundation Utilities.cdp`
3. `${APAMA_FOUNDATION_HOME}/projects/general_infrastructure/Service Framework/Service Framework.cdp`
4. `${APAMA_FOUNDATION_HOME}/ASB/projects/general_infrastructure/Data Views/Data View.cdp`
5. `${APAMA_HOME}/adapters/monitors/IAFStatusManager.mon`
6. `${APAMA_FOUNDATION_HOME}/ASB/projects/general_infrastructure/General Utils/Utils.cdp`
7. `${APAMA_FOUNDATION_HOME}/ASB/projects/general_infrastructure/Status Utils/Status Utils_eventdefs-objects.cdp`
8. `${APAMA_FOUNDATION_HOME}/ASB/projects/Session Manager/Session Manager_eventdefs-classes.cdp`
9. `${APAMA_FOUNDATION_HOME}/ASB/projects/New Market Data API/New Market Data API_events-classes.cdp`
10. `${APAMA_FOUNDATION_HOME}/ASB/projects/New Market Data API/New Market Data API_monitors.cdp`
11. `${APAMA_FOUNDATION_HOME}/ASB/projects/general_infrastructure/Status Utils/Status Utils_monitors.cdp`
12. `${APAMA_FOUNDATION_HOME}/ASB/projects/Session Manager/Session Manager_monitors.cdp`

FIX order management session

This section lists the service monitor injection order when using FIX adapter service monitors.

1. `${APAMA_HOME}/monitors/ScenarioService.mon`
2. `${APAMA_HOME}/adapters/monitors/IAFStatusManager.mon`
3. `${APAMA_FOUNDATION_HOME}/ASB/projects/Legacy Finance Support/Legacy Finance Support.cdp`

4. `${APAMA_HOME}/monitors/StatusSupport.mon`
5. `${APAMA_HOME}/adapters/FIX/monitors/FIX_Events.mon`
6. `${APAMA_HOME}/adapters/FIX/monitors/FIX_SessionManager.mon`
7. `${APAMA_HOME}/adapters/FIX/monitors/FIX_DataManager.mon`
8. `${APAMA_HOME}/adapters/FIX/monitors/FIX_OrderManager.mon`
9. `${APAMA_HOME}/adapters/FIX/monitors/FIX_EventViewer.mon`
10. `${APAMA_HOME}/adapters/FIX/monitors/FIX_StatusManager.mon`

Using a single FIX Session for both Order Management and (MDA) Market Data Sessions

For venues which use a single FIX Session (connection) for MarketData and OrderManagement, the following additional monitors need to be injected.

Note:

Ensure that you inject all monitors listed in the above two sections before injecting the following monitors.

- `${APAMA_HOME}/adapters/FIX/monitors/FIX_NMDA_StatusManager_Bridge.mon`

Note:

APAMA_HOME is the location of the directory where Apama is installed and APAMA_FOUNDATION_HOME is the location of the directory where CMF is installed.

FIX Legacy Market data session

This section lists the service monitor injection order when using FIX adapter service monitors.

1. `${APAMA_HOME}/monitors/ScenarioService.mon`
2. `${APAMA_HOME}/adapters/monitors/IAFStatusManager.mon`
3. `${APAMA_FOUNDATION_HOME}/ASB/projects/Legacy Finance Support/Legacy Finance Support.cdp`
4. `${APAMA_HOME}/monitors/StatusSupport.mon`
5. `${APAMA_HOME}/adapters/FIX/monitors/FIX_Events.mon`
6. `${APAMA_HOME}/adapters/FIX/monitors/FIX_SessionManager.mon`
7. `${APAMA_HOME}/adapters/FIX/monitors/FIX_DataManager.mon`
8. `${APAMA_HOME}/adapters/FIX/monitors/FIX_SubscriptionManager.mon`
9. `${APAMA_HOME}/adapters/FIX/monitors/FIX_OrderManager.mon`
10. `${APAMA_HOME}/adapters/FIX/monitors/FIX_EventViewer.mon`

11. `${APAMA_HOME}/adapters/FIX/monitors/FIX_StatusManager.mon`

Connectivity plug-in FIX MDA session

Inject the monitors listed in “FIX MDA session” on page 82, followed by:

- `${APAMA_HOME}/monitors/ConnectivityPluginsControl.mon`
- `${APAMA_HOME}/monitors/ConnectivityPlugins.mon`

Connectivity plug-in order management session

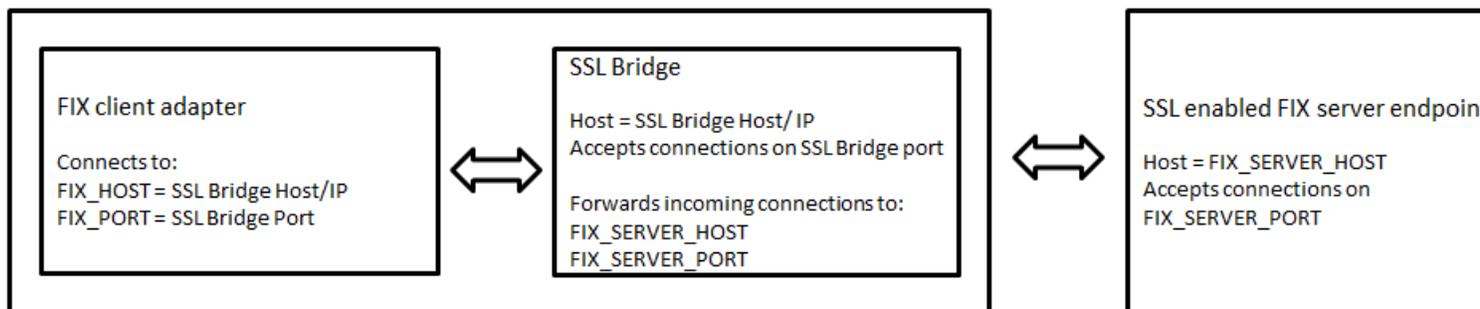
Inject the monitors listed in “FIX order management session” on page 82, followed by:

- `${APAMA_HOME}/monitors/ConnectivityPluginsControl.mon`
- `${APAMA_HOME}/monitors/ConnectivityPlugins.mon`

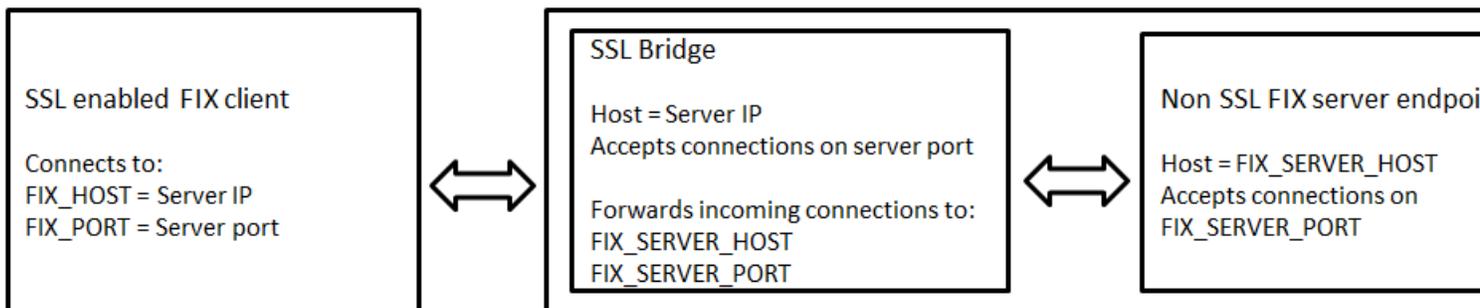
SSL enabled FIX sessions

The FIX Adapter does not support SSL natively. You can use an SSL wrapper to wrap the FIX connection inside SSL. For example, Stunnel (www.stunnel.org). The setup will be something like the following:

FIX client connection to SSL enabled FIX adapter



SSL enabled FIX client connecting to FIX server



For example, if the remote server is running on `remotehost:443`, then in the stunnel configuration file on localhost the following configuration is needed:

```
[fix-adapter]
accept = 12345 (or any unused port number)
connect = remotehost:443
```

Then in the FIX Adapter configuration file, for the transport property `QuickFIX.SocketConnectHost` the value should be "localhost", and for property `QuickFIX.SocketConnectPort` the value should be "12345".

Troubleshooting

Although the FIX adapter has a large number of configuration options, it is relatively simple to set up and manage it. In many cases it is sufficient to use the default configuration files distributed with the adapter and simple modifications of the key parameters to suit your environment (host, port, compIds etc). However problems can and do arise and the adapter produces a large amount of diagnostic information to help in resolving such issues.

FIX log files

The FIX adapter produces a number of log files and it is essential to be aware of these when diagnosing problems. Essentially there are three main logs to consider:

The service log

To configure the `logLevel` and `logFile` details, see "Setting logging attributes for packages, monitors and events" in *Deploying and Managing Apama Applications* in the Apama documentation.

For example, to change the verbosity of the FIX adapter to `DEBUG`:

```
engine_management --setApplicationLogLevel com.apama.fix=DEBUG
```

The IAF log

This is only applicable for FIX IAF Adapter. For FIX Connectivity adapter, the same information is captured in the correlator log. The IAF log contains all logging output from the transport(s) and codec as well as the IAF as a whole. The log will be sent to the IAFs `stdout` unless configured via the log switch (`-f`).

The FIX logs

For each FIX session (that is, transport) the embedded QuickFIX engine will produce a set of log files under the directory that is specified in the `FileLogPath` transport property. By default, this is a directory called `FIX_Logs` and is created under the IAF (Correlator incase of FIX Connectivity adapter) working directory. Usually there will be six files for each session although from a debugging point of view you are primarily interested in:

- `FIX.4.2-<senderCompID>-<targetCompID>.messages.current` – Contains a raw copy of each FIX message that is sent or received by the session.

- `FIX.4.2-<senderCompID>-<targetCompID>.seqnum` – Contains two integers separated by a colon, the first of which is our sequence number and the second of which is the server’s sequence number.

When investigating the FIX message log, it is helpful to know the meanings of the various tags. A useful resource for this is FIXimate (<http://fiximate.fixtrading.org/latestEP>) which has a full listing of all messages within FIX and their corresponding tags.

Diagnosing connectivity/session problems

The first place to look when diagnosing connectivity problems is the Adapter log (IAF log in case of FIX IAF adapter and correlator log in case of FIX Connectivity adapter). Each FIX session should produce the following series of messages upon successfully connecting and logging on to the remote system:

```
...
2007-11-30 10:56:12.070 INFO [2756] -
  FIX.4.2-<Sender>-<Target>::Connecting to
  <host> on port <port>
2007-11-30 10:56:12.480 INFO [2756] -
  FIX.4.2-<Sender>-<Target>::Initiated logon
  request
2007-11-30 10:56:13.462 INFO [2756] -
  FIX.4.2-<Sender>-<Target>::Received logon
  response
...
```

If after attempting to connect you do not see “Initiated logon request”, it is likely that a connection to the specified host/port cannot even be established. In this instance you must investigate your environment/network and establish that a route to the target system is actually possible.

If you do see “Initiated logon request” but no response is received or the connection is terminated by the peer in some way then it is likely that the target system has rejected the logon attempt. In this case you need to look at the FIX message log and investigate the message exchange. It may be that you have provided incorrect details or some essential tag is missing from the logon message.

Another possibility in this instance is that there is a sequence number mismatch. The target system will report this to you and tell you what it expects the sequence number to be. In this case you must shutdown the adapter, edit the sequence number file (discussed above) and make sure that the sequence numbers are correctly aligned with what you and the target system are expecting before restarting.

Once a FIX session is properly established and as long as the sessions have been properly configured by sending in the relevant `SessionConfiguration` event(s) (See “[Client session](#)” on page 28, the session manager will notify all the other FIX service monitors that the session(s) are now up. This will also be logged in the service log as:

```
...
[2007-11-30 10:56:13.462 INFO] FIX Session Manager
  [<CONNECTION_NAME>]: Session
  <CONNECTION_NAME>'s IAF is connected
[2007-11-30 10:56:13.462 INFO] FIX Session Manager
  [<CONNECTION_NAME>]: Session
  <CONNECTION_NAME> has been logged on
```

...

The first message tells us that the session manager is receiving heartbeats from the IAF and the second message tells us that the session manager has received a logon message for that connection.

Diagnosing application problems

As mentioned, all FIX service monitors will output diagnostic information to the service logs. This information can be useful in diagnosing application related problems such as missing or incorrectly specified parameters.

For example, upon receiving a new market data subscription request the subscription manager will output information regarding its current reference counts for that symbol and what (if anything) it intends to send to the market.

However, sometimes it is necessary to dig deeper and establish exactly what is being sent to and received from the target system. In these cases it is best to refer to the FIX message log (as discussed above) to diagnose the problem.

Creating a support case

If you encounter a problem that you cannot solve and need to contact support, there are a number of guidelines (in addition to those set out in the preface) that should be followed to ensure your case is resolved as quickly as possible.

Firstly a full description of the problem and the conditions that resulted in its discovery along with any other information that may help in resolving the issue (Such as order ids, subscription information etc). Secondly the following files should be provided:

- FIX IAF adapter
 - IAF configuration file
 - IAF log file
- All FIX logs
- Service log
- Correlator input log

3 Bank of America FXtransact FIX Adapter

■ Prerequisites	90
■ IAF configuration	90
■ Service monitor extensions	90
■ Service monitor injection order	90
■ Session configuration	90
■ Quote subscription or unsubscription	91
■ BOA InstinctFX Trading Platform	92

This chapter describes the design, implementation and usage of the Bank of America FIX adapter.

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

IAF configuration

When acting as a client to Bank Of America (BOA) FXtransact, IAF should be configured with the Bank Of America (BOA) specific configuration file, the distribution version of which can be found here: `${APAMA_HOME}/adapters/config/FIX-BOA.xml.dist`

Note that the requirement for a BOA-specific configuration file means that an IAF using this configuration file must be used for BOA FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for BOA operation. If you need to connect to other FIX servers as well as BOA, the non-BOA sessions should be configured in a separate IAF instance. However, BOA and non-BOA session can co-exist safely within the same correlator instance. BOA provided host, port, sender and target CompID parameters must be supplied in the IAF configuration file.

Service monitor extensions

The file `FIX_BOA_Support.mon` provides support for subscribing or unsubscribing quotes. Therefore, the following EPL file must be injected into the correlator after the standard FIX adapter service monitors.

- `FIX_BOA_Support.mon`

Service monitor injection order

Inject the monitors listed in [“FIX order management session” on page 82](#) and [“FIX Legacy Market data session” on page 83](#), followed by:

- `${APAMA_HOME}/adapters/monitors/FIX_BOA_Support.mon`

Session configuration

BOA sessions must be configured with the following configuration parameters:

- `FixChannel = BOA_FIX`
- `SERVICEID = BOA-FIX`
- `OrderManager.GenerateNewStyleOrderIds = true`

For example,

```
// For BARX Trading Session
com.apama.fix.SessionConfiguration("BOA_TRADING", {"FixChannel":"BOA_FIX",
  "SERVICEID":"BOA-FIX", "OrderManager.GenerateNewStyleOrderIds":"true"})
// For BARX MARKET DATA Session
com.apama.fix.SessionConfiguration("BOA_MARKET_DATA", {"FixChannel":"BOA_FIX",
  "SERVICEID":"BOA-FIX"})
```

Quote subscription or unsubscription

Request For Quotes (Spots)

```
#Request for base currency: EUR/USD Buy EUR 101,000 Spot
com.apama.marketdata.SubscribeDepth("BOA-FIX", "BOA_MARKET_DATA", "EUR/USD",
  {"Quote":"Y","303":"102","15":"EUR","38":"101000","64":"Spot","54":"1"})
com.apama.marketdata.UnsubscribeDepth("BOA-FIX", "BOA_MARKET_DATA", "EUR/USD",
  {"Quote":"Y","303":"102","15":"EUR","38":"101000","64":"Spot","54":"1"})
#Request for 2-way Base currency: GBP/USD 2-way GBP 103,000 Spot
com.apama.marketdata.SubscribeDepth("BOA-FIX", "BOA_MARKET_DATA", "GBP/USD",
  {"Quote":"Y","303":"102","15":"GBP","38":"103000","64":"Spot"})
com.apama.marketdata.UnsubscribeDepth("BOA-FIX", "BOA_MARKET_DATA", "GBP/USD",
  {"Quote":"Y","303":"102","15":"GBP","38":"103000","64":"Spot"})
#Request for term currency: USD/CAD Sell CAD 105,000 Spot
com.apama.marketdata.SubscribeDepth("BOA-FIX", "BOA_MARKET_DATA", "USD/CAD",
  {"Quote":"Y","303":"102","15":"CAD","38":"105000","64":"Spot","54":"2"})
com.apama.marketdata.UnsubscribeDepth("BOA-FIX", "BOA_MARKET_DATA", "USD/CAD",
  {"Quote":"Y","303":"102","15":"CAD","38":"105000","64":"Spot","54":"2"})
```

Request For Quotes (Forwards)

```
#Request for base currency: EUR/USD Buy EUR 201,000 TOD
com.apama.marketdata.SubscribeDepth("BOA-FIX", "BOA_MARKET_DATA", "EUR/USD",
  {"Quote":"Y","303":"102","15":"EUR","38":"201000","64":"Today","54":"1"})
com.apama.marketdata.UnsubscribeDepth("BOA-FIX", "BOA_MARKET_DATA", "EUR/USD",
  {"Quote":"Y","303":"102","15":"EUR","38":"201000","64":"Today","54":"1"})
#Request for 2-way term currency: EUR/JPY 2-way JPY 204,000 1M
com.apama.marketdata.SubscribeDepth("BOA-FIX", "BOA_MARKET_DATA", "EUR/JPG",
  {"Quote":"Y","303":"102","15":"JPG","38":"204000","64":"1M"})
com.apama.marketdata.UnsubscribeDepth("BOA-FIX", "BOA_MARKET_DATA", "EUR/JPG",
  {"Quote":"Y","303":"102","15":"JPG","38":"204000","64":"1M"})
#Requests for term currency - Broken date (3m+3d): EUR/USD Buy USD 205,000 3M+3
days
com.apama.marketdata.SubscribeDepth("BOA-FIX", "BOA_MARKET_DATA", "EUR/USD",
  {"Quote":"Y","303":"102","15":"USD","38":"205000","64":"20100430","54":"1"})
com.apama.marketdata.UnsubscribeDepth("BOA-FIX", "BOA_MARKET_DATA", "EUR/USD",
  {"Quote":"Y","303":"102","15":"USD","38":"205000","64":"20100430","54":"1"})
```

Request For Stream (Spots)

```
#Requests one-way stream base currency: EUR/USD Buy EUR 401,000 Spot
com.apama.marketdata.SubscribeDepth("BOA-FIX", "BOA_MARKET_DATA", "EUR/USD",
  {"Quote":"Y","303":"103","15":"EUR","38":"401000","64":"Spot","54":"1"})
com.apama.marketdata.UnsubscribeDepth("BOA-FIX", "BOA_MARKET_DATA", "EUR/USD",
  {"Quote":"Y","303":"103","15":"EUR","38":"401000","64":"Spot","54":"1"})
```

Request For Stream (Forwards)

```
#Requests 2-way stream term currency: EUR/JPY 2-way JPY 504,000 1M
com.apama.marketdata.SubscribeDepth("BOA-FIX", "BOA_MARKET_DATA", "EUR/JPY",
    {"Quote": "Y", "303": "103", "15": "JPY", "38": "504000", "64": "1M"})
com.apama.marketdata.UnsubscribeDepth("BOA-FIX", "BOA_MARKET_DATA", "EUR/JPY",
    {"Quote": "Y", "303": "103", "15": "JPY", "38": "504000", "64": "1M"})
```

NewOrders

Previously-quoted orders:

- QuoteID (Tag 117) must be copied from the Quote (reported to the application as `com.apama.marketdata.Depth`) to `NewOrder`.
- `HandlInst`(tag 21) must be set to "1" in the extraparams of `NewOrder`.
- Orders can be placed in the base or term currency. The appropriate price must be chosen from `Depth` when placing an order.

For example,

```
com.apama.oms.NewOrder("BOA:1", "EUR/USD", 1.40065, "BUY", "PREVIOUSLY QUOTED",
    101000, "BOA-FIX", "", "", "BOA_TRADING", "", "",
    {"117": "te-1-4-21515252-22:sm-3:12981", "15": "EUR", "21": "1"})
```

Market orders:

- `HandlInst`(tag 21) must be set to "1" in the extraparams of `NewOrder`.
- Orders can be placed in the base or term currency. The Price will be set to 0 even if provided.

For example,

```
com.apama.oms.NewOrder("BOA:1", "EUR/USD", 0, "BUY", "FOREX MARKET", 6,
    "BOA-FIX", "", "", "BOA_TRADING", "", "", {"15": "EUR", "21": "1", "40": "C", "6215": "Spot"})
```

BOA InstinctFX Trading Platform

IAF configuration

When acting as a client to BOA, the IAF should be configured with the BOA FIX specific configuration file, the distribution version of which can be found here:

```
${APAMA_HOME}/adapters/config/FIX-BOA-MDA.xml.dist
```

For Market data, the path of the correct adapter configuration file should also be provided using the property:

```
<property name="AdapterConfiguration" value="${APAMA_HOME}/adapters/config/FIXTransport-BaseFIXLibrary-BOA.xml" />
```

Note that the requirement for a BOA specific configuration file means that, an IAF using this configuration file must be used for BOA FIX 4.4 sessions only. The FIX adapter will not inter-operate correctly with other FIX servers once it has been configured for BOA operation. If you need to connect to other FIX servers as well as BOA, the non-BOA sessions should be configured in a separate IAF instance. However, BOA and BOA session can co-exist safely within the same correlator instance.

Below properties allows user to add comma separated tag list to get in Extra parameters of Quote of respective level:

- BidLevelFixTagsSet - comma separated tag list which allows to add in bid level EP
- AskLevelFixTagsSet - comma separated tag list which allows to add in ask level EP

```
<property name="AskLevelFixTagsSet" value="634,355"/>
```

```
<property name="BidLevelFixTagsSet" value="632,653"/>
```

Service monitor injection order

Inject the monitors to connect to MDA Session. See [“FIX MDA session” on page 82](#) and [“FIX order management session” on page 82](#).

Session configuration

BOA adapter uses `com.apama.oms` interface for placing orders and CMF MDA for marketData.

Adapter configuration and subscription :

The Adapter configuration file `FIXTransport-BaseFIXLibrary-BOA.xml` contains properties which are used for sending subscription and unsubscription Requests.

BOA-FIX Trading sessions must be configured with the following configuration parameters:

- `FixChannel = BOA_FIX`
- `OrderManager.updateKeyParamsOnStateChange = true`
- `OrderManager.KillOrdersOnSessionDown = false`
- `ServiceID = BOA-FIX`

You should subscribe for the symbol before placing the order.

For example,

```
com.apama.fix.SessionConfiguration("BOA_TRADING", {"FixChannel":"BOA_FIX",
"SERVICEID":"BOA-FIX", "OrderManager.GenerateNewStyleOrderIds":"true",
"OrderManager.KillOrdersOnSessionDown":"false"})
```

MDA adapter configuration and subscription

The MDA adapter configuration file `FIXTransport-BaseFIXLibrary-BOA.xml` contains properties which are used for sending subscription and unsubscription Requests. BOA supports SPOT, Outright and SAWP type subscriptions.

For subscription, the following fields are required. These values should send in `com.apama.session.CtrlParams` parms while subscription.

- "QUANTITY_BANDS" - comma separated Quantity requesting for Quote
- "REQUEST_TYPE" - Quote Request type . Valid Values RFQ and RFS. Default is RFS.
- "Currency" - If User won't provide base currency will take by default.

See BOA specification for details.

Additional properties

`AskLevelFixTagsSet` or `BidLevelFixTagsSet`

User will get below tags in level Ep. More tags can be configured with `AskLevelFixTagsSet` and `BidLevelFixTagsSet` properties.

Bid level default Tags are:

- `BidSpotRate(188)`
- `BidForwardPoints(189)`
- `BidPx2(6050)`
- `BidSpotRate2(6162)`
- `BidForwardPoints2(642)`
- `ValidUntilTime(62)`

Ask Level default Tags are:

- `OfferSpotRate(188)`
- `OfferForwardPoints(189)`
- `OfferSize2(6052)`
- `OfferPx2(6050)`
- `OfferSpotRate2(6162)`
- `OfferForwardPoints2(642)`
- `ValidUntilTime(62)`

RfqQuoteTimeout. As RFQ subscriptions are ceased after particular time or once an order against RFQ quote got placed, use the property RfqQuoteTimeout to define RFQ timeout interval that defaults to 120 seconds.

Fx RestingOrders

Example for NewOrder of StopLoss strategy:

```
com.apama.oms.NewOrder("BOA:1000","EUR/USD",1.11781,"BUY","STOP",1000000,
  "BOA-FIX","","","BOA_TRADING","","",{ "15":"EUR","21":"1",
  "64":"20160917","Party0":"SAG,D,Entering Firm","1":"POLON-LON",
  "581":"1","59":"3","847":"1000","99":"1.11685","168":"00000000-00:00:00",
  "432":"00000000","9017":"Bid","9027":"Y","9028":"1.117","9032":"1.13","9034":"1.11671"})
```

Example for cancel order:

```
com.apama.oms.CancelOrder("BOA:1000","BOA-FIX",
  {"453":["com.apama.fix.ExecutionReport_Party(\"SAG\",
  \"D\", \"Entering Firm\", [], {})]","54":"1"})
```


4 Barclays FX FIX Adapter

■ Prerequisites	98
■ IAF configuration	98
■ Service monitor injection order	99
■ Session configuration	99
■ MDA adapter configuration	99
■ Quote subscription and unsubscription	100
■ Example quote subscriptions and unsubscriptions	100

This chapter describes the design, implementation and usage of the FIX BARX extensions. This version of the FIX-BARX Adapter conforms to the Version 4.6.1 of the Rules of Engagement Document released for 'BARX FX Via FIX'

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

IAF configuration

- BARX Adapter can be configured to run in two modes - Legacy or MDA
- MDA configuration makes use of the Market Data API of the Apama CMF
- Order handling is done using the Legacy Mode even when Quote (Market data) handling is done using MDA mode

When acting as a client to BARX, the IAF should be configured with the BARX specific configuration file, the distribution version of which can be found here:

```
${APAMA_HOME}/adapters/config/FIX-BARX.xml.dist (for MDA Mode)
```

or

```
${APAMA_HOME}/adapters/config/FIX-BARX-Legacy.xml (for Legacy Mode)
```

Note that the requirement for a BARX-specific configuration file means that an IAF using this configuration file must be used for BARX FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for BARX operation. If you need to connect to other FIX servers as well as BARX, the non-BARX sessions should be configured in a separate IAF instance. However, BARX and non-BARX session can co-exist safely within the same correlator instance.

BARX uses a custom data dictionary (FIX42-BARX.xml). This file must be referenced by the "QuickFIX.DataDictionary" transport property. For example:

```
<property name="QuickFIX.DataDictionary"  
value="${APAMA_HOME}/adapters/config/FIX42-BARX.xml"/>
```

The IAF configuration file should contain appropriate host, port, sender and target CompID parameters.

For the MDA Mode the path of the correct adapter configuration file should also be provided using the property -

```
<property name="AdapterConfiguration"  
value="${APAMA_HOME}/adapters/config/FIXTransport-BaseFIXLibrary-Barx.xml" />
```

Service monitor injection order

Inject the monitors listed in “[FIX order management session](#)” on page 82 and “[FIX Legacy Market data session](#)” on page 83, followed by:

- `${APAMA_HOME}/adapters/monitors/FIX_BARX_Support.mon`

Session configuration

BARX sessions must be configured with the following configuration parameters:

TRADING SESSIONS:

- `OrderManager.GenerateNewStyleOrderIds = true`
- `FixChannel = BARX_FIX`

MARKET DATA SESSIONS (Legacy Mode):

- `FixChannel = BARX_FIX`
- `SubscriptionManager.EnableMassQuoteAcknowledgement = true`
- `SERVICEID = BARX-FIX`

For example, for most BARX client installations the following session configuration could be used:

```
// For BARX Trading Session
com.apama.fix.SessionConfiguration("BARX_TRADING", {"FixChannel":"BARX_FIX",
"OrderManager.GenerateNewStyleOrderIds":"true"})
// For BARX MARKET DATA Session
com.apama.fix.SessionConfiguration("BARX_MARKET_DATA", {"FixChannel":"BARX_FIX",
"SubscriptionManager.EnableMassQuoteAcknowledgement":"true","SERVICEID":"BARX-FIX"})
```

MDA adapter configuration

The MDA adapter configuration file `FIXTransport-BaseFIXLibrary-Barx.xml` contains properties which are used for sending subscription and unsubscription requests.

Note:

In case multiple tags need to be specified to be updated/added in outgoing unsubscribe requests, then the format will be

```
<property name="CtrlParamsToUpdate" value="6065:-1 167:FXNDF"/>
```

where, 6065 and 167 are the two tags to be added/updated with their respective values specified after a colon. The tags are separated from one another by a space.

Service monitor injection order

Inject the monitors to connect to MDA Session. See [“FIX MDA session” on page 82](#) and [“FIX order management session” on page 82](#).

Quote subscription and unsubscription

- Symbol field must be currency pair CCY1/CCY2. Here CCY1 is called the "Primary/base currency" and CCY2 is called the "Secondary currency". The ordering of symbols is dependent on ranking of currencies. CCY1 is a higher ranking currency than CCY2. The ranking of symbols is given in section "APPENDIX 2 - CURRENCY RANKING" of the specification "BARX Rules of Engagement Global Fix Connectivity - Foreign Currency version 1"
- Quotes can be subscribed in Primary currency(using "15": "CCY1" in extra parameters) or in Secondary currency (using "15": "CCY2" in extra parameters).
- To subscribe for one-way quotes side must be specified in extra parameters (Ex BUY side ("54": "1") Sell side ("54": "2"). If this parameter is not specified 2-way quotes are subscribed.
- Settlement date (SettlDate tag 64) can be a standard tenor or broken date(YYYYMMDD) and must be specified in extra parameters (Eg "64": "SP", "64": "20091118"). Standard tenors are in the section "How pricing works? Symbology and Product Information " in the specification "BARX Rules of Engagement Global Fix Connectivity - Foreign Currency version 1"
- OrderType is always "C", specified in extra parameters as "40": "C"
- Quotes can be subscribed in different volume bands, by specifying in OrderQty(tag 38) in extra parameters. Ex "38": "1000000" for 1 million band. Standard bands are 1M, 3M, 5M and 10M.
- Snapshot quote subscriptions can be done specifying tag 6065 in extra parameters. Ex "6065": "10" to subscribe quotes for 10 sec. If "6065": "0" then quotes are subscribed till logout.

Example quote subscriptions and unsubscriptions

For Market Data (Quotes Requests) the below are valid examples of Legacy mode scenarios.

```
For Streaming 2-way SPOT prices for GBP/USD for 1M GBP (streaming prices in base
currency)
com.apama.marketdata.SubscribeDepth("FIX", "BARX_MARKET_DATA", "GBP/USD",
{"Quote": "Y", "40": "C", "167": "FOR", "15": "GBP", "38": "1000000", "64": "SP", "6065": "0"})
com.apama.marketdata.UnsubscribeDepth("FIX", "BARX_MARKET_DATA", "GBP/USD",
{"Quote": "Y", "40": "C", "167": "FOR", "15": "GBP", "38": "1000000", "64": "SP", "6065": "0"})
For Streaming 2-way SPOT prices for GBP/USD for 3M GBP (streaming prices in base
currency in different volume band )
com.apama.marketdata.SubscribeDepth("FIX", "BARX_MARKET_DATA", "GBP/USD",
{"Quote": "Y", "40": "C", "167": "FOR", "15": "GBP", "38": "3000000", "64": "SP", "6065": "0"})
com.apama.marketdata.UnsubscribeDepth("FIX", "BARX_MARKET_DATA", "GBP/USD",
{"Quote": "Y", "40": "C", "167": "FOR", "15": "GBP", "38": "3000000", "64": "SP", "6065": "0"})
For Streaming 2-way SPOT prices for GBP/USD for 1M USD (streaming prices in
secondary currency)
com.apama.marketdata.SubscribeDepth("FIX", "BARX_MARKET_DATA", "GBP/USD",
{"Quote": "Y", "40": "C", "167": "FOR", "15": "USD", "38": "1000000", "64": "SP", "6065": "0"})
com.apama.marketdata.UnsubscribeDepth("FIX", "BARX_MARKET_DATA", "GBP/USD",
```

```

{"Quote":"Y","40":"C","167":"FOR","15":"USD","38":"1000000","64":"SP","6065":"0"}
For Streaming 1-way SPOT prices for GBP/USD for 1M GBP (streaming prices in base
currency)
com.apama.marketdata.SubscribeDepth("FIX", "BARX_MARKET_DATA", "GBP/USD",

{"Quote":"Y","54":"1","40":"C","167":"FOR","15":"GBP","38":"1000000","64":"SP","6065":"0"}
com.apama.marketdata.UnsubscribeDepth("FIX", "BARX_MARKET_DATA", "GBP/USD",

{"Quote":"Y","54":"1","40":"C","167":"FOR","15":"GBP","38":"1000000","64":"SP","6065":"0"}
For Snapshot 2-way SPOT prices for GBP/USD for 1M GBP (snapshot prices
in base currency for 10 sec)
com.apama.marketdata.SubscribeDepth("FIX", "BARX_MARKET_DATA", "GBP/USD",

{"Quote":"Y","54":"1","40":"C","167":"FOR","15":"GBP","38":"1000000","64":"SP","6065":"10"}
com.apama.marketdata.UnsubscribeDepth("FIX", "BARX_MARKET_DATA", "GBP/USD",

{"Quote":"Y","54":"1","40":"C","167":"FOR","15":"GBP","38":"1000000","64":"SP","6065":"10"}
For Streaming 2-way FORWARD 2M prices for GBP/USD for 1M GBP (streaming prices in base
currency)
com.apama.marketdata.SubscribeDepth("FIX", "BARX_MARKET_DATA", "GBP/USD",
{"Quote":"Y","40":"C","167":"FOR","15":"GBP","38":"1000000","64":"2M","6065":"0"}
com.apama.marketdata.UnsubscribeDepth("FIX", "BARX_MARKET_DATA", "GBP/USD",
{"Quote":"Y","40":"C","167":"FOR","15":"GBP","38":"1000000","64":"2M","6065":"0"}
For Streaming 2-way FORWARD broken date prices for GBP/USD for 1M GBP
(streaming prices in base currency)
com.apama.marketdata.SubscribeDepth("FIX", "BARX_MARKET_DATA", "GBP/USD",

{"Quote":"Y","40":"C","167":"FOR","15":"GBP","38":"1000000","64":"20091118","6065":"0"}
com.apama.marketdata.UnsubscribeDepth("FIX", "BARX_MARKET_DATA", "GBP/USD",

{"Quote":"Y","40":"C","167":"FOR","15":"GBP","38":"1000000","64":"20091118","6065":"0"}

```

NewOrders

- SecurityType(tag 167) and HandlInst(tag 21) must be specified in extra parameters for all orders. (Ex: "167":"FOR","21":"2")

- At-market orders example:

```

SELL 1M GBP
com.apama.oms.NewOrder("124","GBP/USD",0, "SELL", "FOREX MARKET", 1000000, "FIX",
"", "", "BARX_TRADING", "", "", {"167":"FOR","21":"2","15":"GBP","64":"SP"})
BUY 1M GBP, date 20091113
com.apama.oms.NewOrder("128","GBP/USD",0, "BUY", "FOREX MARKET", 2000000, "FIX",
"", "", "BARX_TRADING", "", "", {"167":"FOR","21":"2","15":"GBP","64":"20091113"})

```

- Previously-quoted orders:
 - SettlDate(Tag 64), QuoteID (Tag 117) must be copied from the Quote (reported to the application as com.apama.marketdata.Depth) to NewOrder.
 - Orders can be placed in the base or secondary currency. The section "Determining whether to use Bid or Offer Price" in the specification tells the appropriate price to be used when placing an order.
 - Examples

```
SELL 2M GBP in base currency.
```

```
com.apama.oms.NewOrder("BARX:1","GBP/USD",1.65614,"SELL","PREVIOUSLY
QUOTED",2000000,
  "FIX","","","BARX_TRADING","","",{ "117":"ZZOAXJPR15574031999897416",
  "15":"GBP","167":"FOR","21":"2","64":"SP"})
BUY 1M USD in secondary currency.
com.apama.oms.NewOrder("BARX:28","GBP/USD",1.65763,"BUY","PREVIOUSLY
QUOTED",1000000,
  "FIX","","","BARX_TRADING","","",{ "117":"FNLHTMMK15574031999845857",
  "15":"USD","167":"FOR","21":"2","64":"SP"})
```

5 Bloomberg B-PIPE Adapter

■ Prerequisites	104
■ Transport configuration	104
■ Service monitor injection order	105
■ Symbol normalization	106
■ Market data service	106
■ Market depth service	107
■ Market depth data service	108
■ Configurability	108
■ Reference data service	110
■ Custom service	116

The BPIPE adapter uses the Apama Integration Adapter Framework (IAF) and the Bloomberg API library to connect to the Bloomberg market data system.

The BPIPE adapter supports below services of Bloomberg for accessing the Market data.

- Market data service (//blp/mktdata)
- Market Depth Service (//blp/mktdepth)
- Market Depth Data Service (//blp/mktdepthdata)
- Request Based Service (//blp/refdata)

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3
- BPIPE library version 3.8
- Developers Guide Link: <http://www.bloomberglabs.com/api/libraries/>

Transport configuration

To configure the transport, make a copy of the `Bloomberg_BPIPE.xml.dist` configuration file (for example, copy to `Bloomberg_BPIPE_PROD.xml`) and edit the copy as described. All user-configurable parameters are marked with `@PARAMETER_NAME@` text in the template configuration. No other part of the transport configuration file should be changed.

Set the parameters in the IAF configuration file:

Parameter	Description
BPIPE_TRANSPORT	Replace "@BPIPE_TRANSPORT@" in the <transport> element with the name of the transport instance.
BPIPE.API(@BPIPE_CLIENT_MODE@)	Name of the transport instance.
BPIPE.Host and BPIPE.Port (@BPIPE_HOST@ & @BPIPE_PORT@)	Host name and IP address of Bloomberg to connect.

For Server Mode

BPIPE.uuid(@BPIPE_UUID@)

Set user identify (UUID) of BLOOMBERG PROFESSIONAL service(BPS).

BPIPE.ipAddress(@BPIPE_IP_ADDRESS@)

The IP address of the host where that user logs to the BLOOMBERG PROFESSIONAL service.

Code snippet to add in transport parameter properties to Enable Server Mode:

```
<!-- Server API parameter parameters -->
<property name="BPIPE.uuid" value="@BPIPE_UUID@" />
<property name="BPIPE.ipAddress" value="@BPIPE_IP_ADDRESS@" />
```

where @BPIPE_UUID@ and @BPIPE_IP_ADDRESS@ are replaced with UUID and IP address respectively.

For BPIPE Mode :(Following are Authentication Options)

BPIPE.AuthenticationMode(@BPIPE_AUTHENTICATION_MODE@)

used to set the Authentication mode of Bloomberg Session. The valid options are USER_ONLY or APPLICATION_ONLY or USER_AND_APPLICATION.

BPIPE.AuthenticationType(@BPIPE_AUTHENTICATION_TYPE@)

Required for USER_ONLY or USER_AND_APPLICATION Mode. The valid options are "OS_LOGON" or "DIRECTORY_SERVICE"

BPIPE.ApplicationName(@BPIPE_APPLICATION_NAME@)

Required for APPLICATION_ONLY or USER_AND_APPLICATION Mode.

Note:

The above configuration properties are attached to the Bloomberg SessionOptions object and thus passed into the session when it is created for details, refer to Bloomberg API documents. Application should be a Feed Adaptor for using BPS (Bloomberg Professional Service).

CorrelatorHBTimeout (@CORRELATOR_HB_TIMEOUT@)

A natural number telling the transport how many seconds it should wait for a heartbeat from the service monitors before deciding that the connection had been lost. If not specified in the configuration file its default value of 15 seconds is used.

LogSeverity(@BPIPE_LOG_VERBOSITY@)

Set the threshold of Severity to Bloomberg for logging the messages. The valid severity are : OFF, ERROR, WARN, INFO, DEBUG, TRACE.

```
<property name="LogSeverity" value="WARN" />
```

DispatcherThreads

This is for Bloomberg Session object and used to set the no of dispatcher threads which process the events.

the default value is 3.

```
<property name="DispatcherThreads" value="3" />
```

Service monitor injection order

Inject the monitors to connect to MDA Session listed in [“FIX MDA session” on page 82](#), followed by:

- `${APAMA_HOME}/adapters/monitors/BPIPE_Support.mon`
- `${APAMA_HOME}/adapters/monitors/BPIPE_ReqBasedService.mon`

Symbol normalization

For subscription to Bloomberg, a security must conform to the following syntax:

```
/[Topic Prefix]/SYMBOLGY[@Pricing Source][Exchange] [Yellow Key]
```

Topic Prefix is one of the following:

```
ticker/cusip/wpk/isin/buid/sedol1/sedol2/sicovam/common/bsid/svm/cins/cats/bbgid
```

[SYMBOLGY]	Is the symbol/ security of Exchange.
[Exchange]	it is optional and is a two character mnemonic for the exchange where the security is traded.
[Yellow Key]	it is the text equivalent of one of the Bloomberg yellow function keys. Govt/Corp/Mtge/M-Mkt/Muni/Pfd/Equity/Comdty/Index/Curncy/Client
[Pricing Source]	it is provider's pricing for a specific security or for a universe of securities.

You can configure the above fields by following control parameters of Connect stream:

for [Topic Prefix]	"TOPIC"
for [Exchange]	"EXCHANGE"
for [Yellow Key]	"YELLOW_KEY"
for [Pricing Source]	"PRICING_SOURCE"

If user does not specify value for [Topic Prefix] and [Yellow Key] then adapter set the default value as "ticker" and "Equity".

Example for Market data service:

If Symbol is "TT314347" and Connection CtrlParams as "YELLOW_KEY":"corp","PRICING_SOURCE":"UKRB", the subscription key is `//blp/mktdata/ticker/TT314347@UKRB corp`.

Market data service

The Market Data service (MarketDataService) enables retrieval of streaming data for securities which are priced intra-day, by using the Bloomberg API subscription paradigm. All fields desired must be explicitly listed in the subscription to receive their updates.

Streams supported:

- DEPTH
- TRADE
- BBA

Based on stream type, the adapter sends the default field list for a subscription.

For Depth subscription, following fields are listed as a default:

- BEST_BID1 thru BEST_BID5 ->First thru tenth best bid price in ten levels of market depth.
- BEST_BID1_SZ thru BEST_BID5_SZ Size of first thru tenth best bid in ten levels of market depth.
- BEST_ASK1 thru BEST_ASK5 First thru tenth best ask price in ten levels of market depth.
- BEST_ASK1_SZ thru BEST_ASK5_SZ Size of first thru tenth best ask in ten levels of.

For Tick data, following Bloomberg fields are listed as a default:

- EVT_TRADE_PRICE_RT ,EVT_TRADE_SIZE_RT,EVT_TRADE_DATE_RT and EVT_TRADE_TIME_RT.

For BBA subscription, following fields are listed as a default for any subscription.

- EVT_QUOTE_BID_PRICE_RT,EVT_QUOTE_BID_SIZE_RT,EVT_QUOTE_ASK_PRICE_RT and EVT_QUOTE_ASK_SIZE_RT.

You can subscribe other fields by setting "Fields" parameters of the Connect Stream.

Example for Depth subscription:

if Symbol is "TT314347@UKRB" and Connection CtrlParams are "YELLOW_KEY":"corp" and "Fields":"LAST_TRADE,SIZE_LAST_TRADE" then subscription will be
 "//blp/mktdata/ticker/TT314347@UKRB corp?fields=LAST_TRADE,SIZE_LAST_TRADE,
 BEST_ASK3_SZ:BEST_ASK2_SZ:BEST_ASK1:BEST_ASK1_SZ:BEST_ASK4:BEST_BID1_SZ:BEST_ASK2:
 BEST_ASK3:BEST_ASK5:BEST_ASK4_SZ:BEST_ASK5_SZ: BEST_BID1:
 BEST_BID2:BEST_BID2_SZ:BEST_BID3:BEST_BID3_SZ:
 BEST_BID4:BEST_BID4_SZ:BEST_BID5:BEST_BID5_SZ".

Market depth service

The Market Depth service is subscription-based and allows the subscription to all levels of market depth data. From this service user can obtain both "Market By Order" (MBO) and aggregated Market By Level (MBL) data books. Adapter supports DEPTH/ORDERBOOK stream type of books and performed the corresponding action on that.

Example: Symbol is "TT314347@UKRB" and CtrlParams parameter "YELLOW_KEY":"corp"

for Depth //blp/mktdepth/ticker/TT314347@UKRB corp?type=MBL"

for ORDERBOOK //blp/mktdepth/ticker/TT314347@UKRB corp?type=MBO"

To support the market depth service you must enable the transport configuration property (SupportLegacyMarketDepthService).

Market depth data service

This service same as Market Depth Service(//blp/mktdepthdata).it also support both "Market By Order" (MBO) and aggregated Market By position (MBP) data books.

Example: Symbol is "TT314347@UKRB" and CtrlParams parameter "YELLOW_KEY":"corp"

for Depth //blp/mktdepthdata/ticker/TT314347@UKRB corp?type=MBL"

for ORDERBOOK //blp/mktdepthdata/ticker/TT314347@UKRB corp?type=MBO"

Configurability

Configurability is a comfortable way of providing fields through predefined MODELS rather than explicitly giving them as input during subscriptions.

Format: XML

Config File has three sections:

- Fields
- Models
- Streams

Fields

Fields are used for providing information regarding Bloomberg field. Where name is used for the BPIPE field and description for information regarding the field.

Models

Every Model has 2 properties mappings,fields. Models have entity 'depends' which allows them to inherit other models fields and mappings. Mappings are used to determine the stream type for a model. Interface in mappings is used to define the streamType of a model. Mappings have map entity. Each map entity has field and name. Name defines the values used for the streamType and field defines the Bloomberg Symbol that will be used while subscription.

Streams

This is used for defining the default model for a stream type.

Examples:

1. Add a field "BEST_BID" with the description "Best bid value - Real Time"

Here BEST_BID is the BPIPE field that can be accessed from the Bloomberg BPS.

```
<field name="BEST_BID" description="Best bid value - Real Time"/>
```

Add the above line in the fields section for adding a field.

- Define a Model "BBAQuoteFields" with streamType "BBA" and fields BID,BID_SIZE,ASK,ASK_SIZE

```
<model name="BBAFields">
  <mappings interface="BBA">
    <map name="BID_PRICE" field="BID"/>
    <map name="BID_SIZE" field="BID_SIZE"/>
    <map name="ASK_PRICE" field="ASK"/>
    <map name="ASK_SIZE" field="ASK_SIZE"/>
  </mappings>
</model>
```

Note:

Names in the BBA fields (BID_PRICE,BID_SIZE,ASK_PRICE,ASK_SIZE) are mandatory.

- Define a Model "TradeFields" with streamType "Trade" and fields EVT_TRADE_PRICE_RT,EVT_TRADE_SIZE_RT

```
<model name="TradeFields">
  <mappings interface="Trade">
    <map name="TRADE_PRICE" field="EVT_TRADE_PRICE_RT"/>
    <map name="TRADE_SIZE" field="EVT_TRADE_SIZE_RT"/>
  </mappings>
</model>
```

Note:

Names in the Trade fields(TRADE_PRICE,TRADE_SIZE) are mandatory.

- Define a Model "Level2Fields" with streamType "MBP" that contains two level fields BEST_BID1,BEST_BID1_SZ,BEST_ASK1,BEST_ASK1_SZ, BEST_BID2,BEST_BID2_SZ,BEST_ASK2,BEST_ASK2_SZ

```
<model name="Level2Fields">
  <mappings interface="MBP">
    <map name="BID1" field="BEST_BID1"/>
    <map name="BID1_SIZE" field="BEST_BID1_SZ"/>
    <map name="ASK1" field="BEST_ASK1"/>
    <map name="ASK1_SIZE" field="BEST_ASK1_SZ"/>
    <map name="BID2" field="BEST_BID2"/>
    <map name="BID2_SIZE" field="BEST_BID2_SZ"/>
    <map name="ASK2" field="BEST_ASK2"/>
    <map name="ASK2_SIZE" field="BEST_ASK2_SZ"/>
  </mappings>
</model>
```

- Define a Model BBAQuoteWithEP from models BBAQuoteFields and BBAQuoteEPFields

```
<model name="BBAQuoteWithEP" depends="BBAQuoteFields,BBAQuoteEPFields">
</model>
```

Note:

When a model is dependent on two models of different stream types, apart from the streamType requested other fields will be considered as ExtraParameters

Configuration of the feature from transport parameter properties:

```
<property name="ConfigurationFileName" value="@CONFIGURATION_FILE@" />
```

Replace @CONFIGURATION_FILE@ with the complete path to configuration File.

Reference data service

Note:

The reference data service will be deprecated in a future release.

This service is an EP service but supports BBA (Best Bid Ask) for ReferenceDataRequests. To support this service, you must enable the transport parameter property SupportAdditionalServices.

Example for enabling the service:

```
<property name="SupportAdditionalServices" value="Y" />
```

This service supports six types of requests. All are EP (Extra Parameter) Data except for ReferenceDataRequests that supports BBA. List of available requests are as follows:

- ReferenceDataRequest
- HistoricalDataRequest
- IntradayTickRequest
- IntradayBarRequest
- PortfolioDataRequest
- BeqsRequest

Note:

Refer to BPIPE_support.mon for event definitions.

1. ReferenceDataRequest

Field Types	Values and Description
Streams Supported	BBA, EP
Fields	Fields the user wants. fields can be specified using Fields Parameter or through MODELS. See “Configurability” on page 108 .
Overrides	Overrides event passed through controlParams. A sequence of overrideInfo events.

For example:

```
Overrides = com.apama.bpipe.overrides([com.apama.bpipe.overrideInfo("\PX_BID\","\13\")])
```

```
Fields = BID,ASK,BID_SIZE,ASK_SIZE or MODELS = BBAFields
```

Note:

For BBA stream type, default model should be BBAFields. Edit the BLOOMBERG.xml file to change default model.

Example Response:

Keys	Values
Fields	ASK,PX_BID,ASK_SIZE,BID,BID_SIZE
Symbol	TT314347@UKRB
Yellow Key	Corp

Adding above field to response parameters:

```
com.apama.bpipe.overrides overrides := new com.apama.bpipe.overrides;

com.apama.bpipe.overrideInfo ft := new com.apama.bpipe.overrideInfo;
ft.fieldId:="PX_BID";
ft.value := "13";
overrides.overrideInfo.append(ft);

com.apama.session.CtrlParams ctrlParams := new com.apama.session.CtrlParams;
ctrlParams.addParam("overrides",overrides.toString());
ctrlParams.addParam("Fields","BID,ASK,BID_SIZE,ASK_SIZE"); or
ctrlParams.addParam("MODELS","BBAFields");
```

The response will be a referenceDataResponse event.

```
ReferenceDataResponse :
com.apama.bpipe.ReferenceDataResponse("TT314347@UKRB corp",
com.apama.bpipe.fieldData({"ASK":"140.100000","PX_BID":
"13.000000","ASK_SIZE":"25000","BID":"13.000000","BID_SIZE":"25000"}))
```

2. HistoricalDataRequest

Field Types	Variable Name	Description
Streams supported		EP
Fields required		
	Fields	Fields the user wants. fields can be specified using Fields Parameter or through MODELS. See “Configurability” on page 108.
	startDate	The start date in a year/month/day format.
	endDate	The end date in a year/month/day format. This

Field Types	Variable Name	Description
		will default to the current day if not specified.
OptionalFields		
	calendarOverridesInfo	Consists of two fields , calendarOverrides and calendareOverridesOperation calendarOverridesInfo must be passed as an calendarOverridesInfo event through control params.
	calendarOverrides	A sequence of two character calendar code.
	calendareOverridesOperation	Can be CDR_AND (Returns the intersection of trading days) or CDR_OR (Returns the union of trading days)

For more optional fields, look into BPIPE Developers Guide.

Examples

Keys	Values
Fields	BID, ASK
startDate	20150401
endDate	20150403
calendarOverridesInfo	com.apama.bpipe.calendarOverridesInfo("CDR_AND",com.apama.bpipe.calendarOverrides("UN","JN"))

Adding the above fields to control parameters:

```
com.apama.bpipe.calendarOverridesInfo
    calenderOverridesInfo := new com.apama.bpipe.calendarOverridesInfo;
com.apama.bpipe.calendarOverrides
    caloverrides := new com.apama.bpipe.calendarOverrides;
calendarOverrides.exchange:=["UN", "JN"];
calendarOverridesInfo.calendarOverrides.append(caloverrides);
calenderOverridesInfo.Operation:="CDR_AND";
com.apama.session.CtrlParams ctrlParams := new com.apama.session.CtrlParams;
ctrlParams.addParam("calendarOverridesInfo",calenderOverridesInfo.toString());
ctrlParams.addParam("Fields", "BID,ASK");
ctrlParams.addParam("startDate", "20150401");
ctrlParams.addParam("endDate", "20150403");
```

HistoricalDataResponse. The response will be an HistoricalDataResponse event.

3. IntradayTickRequest

Field Types	Variable Name	Description
Streams supported		EP
Fields required		
	Fields	Fields the user wants. fields can be specified using Fields Parameter or through MODELS. See “Configurability” on page 108.
	startDateTime	The start date and time.
	endDateTime	The end date and time.
	eventType	Can be one or more of the TRADE, BID, ASK, BID_BEST, ASK_BEST, MID_PRICE, AT_TRADE, BEST_BID, BEST_ASK
OptionalFields		For more optional fields look into BPIPE Developers Guide.

Examples

Keys	Values
Fields	BID, ASK
startDateTime	2010-04-27T15:55:00
endDateTime	2010-04-27T16:00:00
calendarOverridesInfo	cTRADE, BID, ASK

Adding the above fields to control parameters:

```
com.apama.session.CtrlParams ctrlParams := new com.apama.session.CtrlParams;
ctrlParams.addParam("Fields","BID,ASK");
ctrlParams.addParam("endDateTime","2010-04-27T15:55:00");
ctrlParams.addParam("endDateTime","2010-04-27T16:00:00");
ctrlParams.addParam("eventType","TRADE,BID,ASK");
```

The response will be an IntradayTickResponse event.

4. IntradayBarRequest

Field Types	Variable Name	Description
Streams supported		EP
Fields required		
	Fields	Fields the user wants. fields can be specified using Fields Parameter or through MODELS. See “Configurability” on page 108.
	startDateTime	The start date and time.
	endDateTime	The end date and time.
	eventType	Can be one of the TRADE, BID, ASK, BID_BEST, ASK_BEST, MID_PRICE, AT_TRADE, BEST_BID, BEST_ASK
OptionalFields		For more optional fields look into BPIPE Developers Guide.

Examples

Keys	Values
Fields	BID, ASK
startDateTime	2010-04-27T15:55:00
endDateTime	2010-04-27T16:00:00
calendarOverridesInfo	TRADE

Adding the above fields to control parameters:

```
com.apama.session.CtrlParams
  ctrlParams := new com.apama.session.CtrlParams;
  ctrlParams.addParam("Fields","BID,ASK");
  ctrlParams.addParam("endDateTime","2010-04-27T15:55:00");
  ctrlParams.addParam("endDateTime","2010-04-27T16:00:00");
  ctrlParams.addParam("eventType","TRADE");
```

The response will be an IntradayBarResponse event.

5. PortfolioDataRequest

Field Types	Variable Name	Description
Streams supported		EP

Field Types	Variable Name	Description
Symbol		UXXXXXXX-X (use PRTU<GO> in BPS)
YELLOW_KEY		Client
Fields required		
	Fields	Fields can be one or more of PORTFOLIO_MEMBER, PORTFOLIO_MPOSITION, PORTFOLIO_MWEIGHT & PORTFOLIO_DATA
OptionalFields		For more optional fields look into BPIPE Developers Guide.

Examples

Keys	Values
Fields	PORTFOLIO_MEMBER,PORTFOLIO_MPOSITION

Adding the above fields to control parameters:

```
com.apama.session.CtrlParams
  ctrlParams := new com.apama.session.CtrlParams;
  ctrlParams.addParam("Fields","PORTFOLIO_MWEIGHT");
```

The response will be an PortfolioDataResponse event.

6. BeqsRequest

Field Types	Variable Name	Description
Streams supported		EP
Fields required		UXXXXXXX-X (use PRTU<GO> in BPS)
	screenName	The name of the screen to execute. It can be a user defined EQS screen or one of the Bloomberg Example screens on EQS <GO> on the BPS
	ScreenType	Use PRIVATE for user-defined EQS screen. Use GLOBAL for Bloomberg EQS screen.

Field Types	Variable Name	Description
OptionalFields		For more optional fields look into BPIPE Developers Guide.

Examples

Keys	Values
ScreenName	Global Volume Surges
ScreenType	GLOBAL

```
com.apama.session.CtrlParams
  ctrlParams := new com.apama.session.CtrlParams;
  ctrlParams.addParam("ScreenName","Global Volume Surges");
  ctrlParams.addParam("ScreenType","Global");
```

Response will be a BeqsResponse event.

Custom service

Custom service is an Extra Parameter (EP) based service where you can configure the adapter to connect to any request-response based service which is offered by Bloomberg.. To configure the adapter, you must know the requests and response schemas. For more information on requests and response schemas, see the Bloomberg documentation. In this EP based service, you must define the events to capture the response.

Note:

All the requests receive a response and must unsubscribe on receiving the response. There can be multiple response events in case of fragmented response. Final response can be identified from payload dictionary with key EventType and value RESPONSE.

Configuring the adapter

To configure the adapter

1. You must add a new custom service as a comma separated list to the IAF configuration file *UniqueServiceName:bloombergservice*. For example:

```
<property name="CustomServiceList"
  value="RequestResponseService://blp/refdata,IntradaybarService://blp/intradaybar"
/>
```

To configure the new service, you must use the *UniqueServiceName* provided in the IAF configuration file. These custom services can service more than one type of requests. For example, *//blp/refdata* can service requests of type *HistoricalDataRequest*, *IntradayTickRequest*, *PortfolioDataRequest* and so on.

2. To connect to the new custom services that are configured and to send a request to the service, you must set the values to the fields declared in the schema for that request. To do so, use the

control parameters. The control parameter `REQ_TYPE` sets the request in the service being targeted and the rest are string key value pairs with the key being the field from the request schema. All the fields are case sensitive. For example, the request `IntradayBarRequest` requires the following fields in the schema:

- `Security`(STRING)
- `eventType`(ENUMERATION)
- `startDateTime`(DATETIME)
- `endDateTime`(DATETIME)
- `interval`(INT32)
- `gapFillInitialBar`(BOOL)
- `returnEids`(BOOL)
- `adjustmentNormal`(BOOL)
- `adjustmentAbnormal`(BOOL)
- `adjustmentSplit`(BOOL)
- `adjustmentFollowPDDF`(BOOL)
- `maxdataPoints`(INT32)
- `maxDataPointsOrigin`(ENUMERATION)

You must set the control parameters for the above fields in the service request. For example:

```
com.apama.session.CtrlParams({"YELLOW_KEY":"corp",
  "REQ_TYPE":"IntradayBarRequest","eventType":"BID",
  "endDateTime":"2018-04-09T15:55:00",
  "startDateTime":"2018-04-08T15:55:00",
  "interval":"10"})
```

3. The requests receive a response from the Bloomberg. These can be translated into Apama events that can be used in the application. To start with, add event mappings for the response messages in a static groups definition file.

```
<property name="RequestResponseService:barTickData"
  value="com.apama.bpipe.reqbasedservice.BarTickData{string time;
  float open;float high;float low;float close;integer volume;}" />
<property name="RequestResponseService:IntradayBarResponse"
  value="com.apama.bpipe.reqbasedservice.IntradayBarResponse
  {string eidData;group barTickData;}" />
```

Each event must be named identical to the response name. For example, `IntradayBarResponse` from Bloomberg must be mapped to an Apama event of type `IntradayBarResponse`.

The fields of type sequence in the schema must be mapped to a separate event and must be used a group as shown above for `barTickData`.

All the schema fields will be mapped to the variables in the event matching the names. For example, ELEMENT time of IntradayBarResponse from Bloomberg will be mapped to the parameter time in the Apama event IntradayBarResponse.

All unmapped fields of Bloomberg responses will be added to __payload dictionary with key being the Bloomberg field.

4. Add the event definitions in an EPL file to support the above events.

```
event BarTickData{
    string time;
    float open;
    float high ;
    float low;
    float close;
    integer volume;
    dictionary <integer, string> _extraParams;
}
event IntradayBarResponse
{
    string eidData;
    sequence<barTickData> NoBarTickData;
    dictionary<string,string> __payload;
}
```

Sample

The reference data service `\\blp\refdata` is a request response schema based service offered by Bloomberg.

The list of available requests are as follows:

- ReferenceDataRequest
- HistoricalDataRequest
- IntradayTickRequest
- IntradayBarRequest
- BeqsRequest
- IntradayBarDateTimeChoiceRequest

1. ReferenceDataRequest

Control parameters	Values and description
REQ_TYPE	ReferenceDataRequest
Fields	Comma separated list of fields. Fields can be specified using the fields parameter or through MODELS. See “Configurability” on page 108 .

Control parameters	Values and description
overrides	A string format sequence of <code>com.apama.bpipe.reqbasedservice.Overrides</code> events.

Example on set parameter of reference request:

```
com.apama.session.CtrlParams params := new com.apama.session.CtrlParams;
sequence<com.apama.bpipe.reqbasedservice.Overrides>
  overrideList := new sequence<com.apama.bpipe.reqbasedservice.Overrides>;
com.apama.bpipe.reqbasedservice.Overrides
  overrides := new com.apama.bpipe.reqbasedservice.Overrides;
overrides.fieldId:="PX_BID";
overrides.value := "13";
overrideList.append(overrides);
com.apama.bpipe.reqbasedservice.Overrides
  overrides2 := new com.apama.bpipe.reqbasedservice.Overrides;
overrides2.fieldId:="PRICING_SOURCE";
overrides2.value := "CG";
overrideList.append(overrides2);
params.addParam("overrides",OverrideList.toString());
params.addParam("YELLOW_KEY", "Corp")
params.addParam("REQ_TYPE", "ReferenceDataRequest")
params.addParam("Fields", "PX_LAST,BID,ASK")
```

2. HistoricalDataRequest

Control parameters	Values and description
REQ_TYPE	HistoricalDataRequest
Fields	Comma separated list of fields. Fields can be specified using the fields parameter or through MODELS. See “Configurability” on page 108 .
startDate	The start date in a year/month/day format. For example: 20180418.
endDate	The end date in a year/month/day format. This will default to the current day if not specified.
calendarOverridesInfo	A string format sequence of <code>com.apama.bpipe.reqbasedservice.CalendarOverridesInfo</code> events. Consists of two fields: <code>calendarOverrides</code> and <code>calendarOverridesOperation</code>

For more optional fields, look into BPIPE Developers Guide.

Examples

Keys	Values
Fields	BID, ASK

Keys	Values
startDate	20150401
endDate	20150403

Adding the above fields to control parameters:

```
sequence<com.apama.bpipe.reqbasedservice.CalendarOverridesInfo>
  CalendarOverridesList :=
    new sequence<com.apama.bpipe.reqbasedservice.CalendarOverridesInfo>;
com.apama.bpipe.reqbasedservice.CalendarOverridesInfo
  coInfo := new com.apama.bpipe.reqbasedservice.CalendarOverridesInfo;
coInfo.calendarOverrides := "UN,JN";
coInfo.calendarOverridesOperation:="CDR_AND";
CalendarOverridesList.append(coInfo);
params.addParam("calendarOverridesInfo",CalendarOverridesList.toString());
```

HistoricalDataResponse. The response will be an HistoricalDataResponse event.

3. IntradayTickRequest

Control parameters	Value and description
REQ_TYPE	IntradayTickRequest
Fields	Comma separated list of fields. Fields can be specified using the fields parameter or through MODELS. See “Configurability” on page 108 .
startDateTime	The start date and time.
endDateTime	The end date and time.
eventTypes	Can be one of the TRADE, BID, ASK, BID_BEST, ASK_BEST, MID_PRICE, AT_TRADE, BEST_BID, BEST_ASK

Examples

Keys	Values
Fields	BID, ASK
startDateTime	2010-04-27T15:55:00
endDateTime	2010-04-27T16:00:00

The response will be an IntradayTickResponse event.

4. IntradayBarRequest

Control parameters	Value and description
REQ_TYPE	IntradayBarRequest
Fields	Comma separated list of fields. Fields can be specified using the fields parameter or through MODELS. See “Configurability” on page 108 .
startDateTime	The start date and time.
endDateTime	The end date and time.
eventTypes	Can be one of the TRADE, BID, ASK, BID_BEST, ASK_BEST, MID_PRICE, AT_TRADE, BEST_BID, BEST_ASK

Examples

Keys	Values
Fields	BID, ASK
startDateTime	2010-04-27T15:55:00
endDateTime	2010-04-27T16:00:00

The response will be an IntradayBarResponse event.

5. BeqsRequest

Control parameters	Value and description
REQ_TYPE	BeqsRequest
screenName	The name of the screen to execute. It can be a user defined EQS screen or one of the Bloomberg Example screens on EQS <GO> on the BPS
screenType	Use PRIVATE for user-defined EQS screen. Use GLOBAL for Bloomberg EQS screen.

Examples

Keys	Values
screenName	Vehicle-Engine-Parts
screenType	GLOBAL
languageId	GERMAN

```
com.apama.session.CtrlParams
```

```
ctrlParams := new com.apama.session.CtrlParams;
ctrlParams.addParam("screenName","Vehicle-Engine-Parts");
ctrlParams.addParam("screenType","Global");
ctrlParams.addParam("languageId","GERMAN");
```

Response will be a BeqsResponse event.

6. IntradayBarDateTimeChoiceRequest

Control parameters	Value and description
REQ_TYPE	IntradayBarDateTimeChoiceRequest
eventTypes	Can be one of the TRADE, BID, ASK.
security	Security to subscribe to.
interval	Interval for each update.
dateTimeInfo	Choice of setting start and end datetime or list start datetimes and duration. Can have the value of "dateTimeRange" or "startDateDuration".

dateTimeRange requires the following control parameters:

- **startDateTime**. Start time of interval.
- **endDateTime**. End time of interval.

startDateDuration. Provides an array of start datetime and duration for every day. Requires the following control parameter:

- **rangeStartDateTimeList**. List of comma separated string of data and times.

Note:
All timelist should be of the format YYYY-MM-DDTHH:MM:SS. For example, 2018-04-08T15:55:00 for the above request.

- **duration**. Time required to receive the data.

The response will be an IntradayBarResponse event.

6 Bloomberg FXGO FIX Server Adapter

■ Prerequisites	124
■ IAF configuration	124
■ Service monitor injection order	124
■ Working with Bloomberg	124
■ Working of the Quote Provider	125
■ Quote Management	126

This file describes the steps that must be undertaken in addition to standard FIX configuration when connecting to the Bloomberg FXGO system. It describes the Financial Information Exchange (FIX) interface used by liquidity providers to connect to BLOOMBERG® FXGO Trading system. A liquidity provider is a banking institution which provides an automated stream of deal able prices to BM FXGO Trading.

The adapter is certified against v4.1.2 of the Bloomberg FX FIXBOOK API and supports MiFID-II requirements.

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

IAF configuration

BloombergFXGO uses a custom data dictionary (FIX42-BBG.xml). This file must be referenced by the "QuickFIX.DataDictionary" transport property.

For example: `<property name="QuickFIX.DataDictionary" value="{APAMA_HOME}/adapters/config/FIX42-BBG.xml"/>`

The supplied sample configuration file (Fix-BBG.xml.dist) contains place holders for the necessary configuration parameters. This configuration file defines three connections (TRADING, MARKET DATA, QUOTES) using the event channel "FIX" for communication with the correlator.

Service monitor injection order

Inject the monitors listed in [“FIX order management session” on page 82](#), followed by:

- `{APAMA_HOME}/adapters/FIX/monitors/FIX_OrderServer.mon`
- `{APAMA_HOME}/adapters/FIX/monitors/FIX_MarketDataServer.mon`
- `{APAMA_HOME}/adapters/FIX/monitors/FIX_QuoteServer.mon`
- `{APAMA_HOME}/adapters/FIX/monitors/FIX_BBG_Events.mon`
- `{APAMA_HOME}/adapters/FIX/monitors/FIX_BBG_Support.mon`

Working with Bloomberg

Configure the Bloomberg session.

Example:

Market data sessions:

```
com.apama.fix.MDServerConfiguration("MDS",{"RequestKeyParams":""})
```

Trading sessions:

```
com.apama.fix.ServerConfiguration("OMS",
  {"OrderServer.AckPendingNewState":"false",
   "OrderServer.ForwardPartyInfo":"true",
   "FIX.TagsToSupress":"78 768 1907 2593 2668"})
```

Quote Sessions:

```
com.apama.fix.QServerConfiguration("RFQ",
  {"SERVICEID":"BBG-FIX","RequestKeyParams":"5082 40"})
```

Note:

If you are just interested in working on Quote/Trading, then Bloomberg expects both to be in a single session. In this case, you can share Quote session with order session using extra parameter `UseSharedSessionManager` and just use Order session.

```
com.apama.fix.ServerConfiguration("OMS",
  {"OrderServer.AckPendingNewState":"false",
   "OrderServer.ForwardPartyInfo":"true",
   "FIX.TagsToSupress":"78 768 1907 2593 2668"})
com.apama.fix.QServerConfiguration("OMS",
  {"SERVICEID":"BBG-FIX","RequestKeyParams":"508240",
   "UseSharedSessionManager":"true"})
```

Working of the Quote Provider

The System consists of three parts:

- Server (Bloomberg server)
- FIX Provider (The Apama adapter)
- Business Application (To be made by the user)

The Provider establishes a FIX connection to the server as a initiator.

Market Data

The Server sends FIX `MarketDataRequest` messages to the FIX Provider. These messages are converted to `com.apama.mds.SubscribeDepth` events with Service ID "MarketSimulator". Such messages are processed by the Application. The Application in turn keeps track of subscriptions and sends appropriate `com.apama.mds.MDServerDepth` events. These are converted to FIX Snapshot events and sent to the Server.

Message Flow

Application	FIX order Provider	Server
	<---- oms.NewOrder <----	NewOrderSingle
	ExecutionReport(reject) ---->	
oms.OrderUpdate	----> ExecutionReport ---->	

Application	FIX order Provider	Server
	<---- oms.CancelOrder <----	OrderCancelRequest

Quote Management

Once you establish a session with Bloomberg, it will start sending Quote requests/Market data requests. The received requests will be first converted to `com.apama.fix.bloomberg.QuoteRequest` then will be converted to `com.apama.mds.SubscribeDepth` and routed to external Business application. Business application needs to send `com.apama.mds.MDServerDepth` to the adapter. The adapter processes `MDServerDepth` and sends `com.apama.fix.Quote` to Bloomberg.

Example:

```
com.apama.fix.QuoteRequest("RFQ","FIX.4.2:APAMA_RFQ_TEST->Bloomberg_FX_TEST",
"1",[com.apama.fix.QuoteRequest_RelatedSym("EUR/USD",1000,{15:"EUR",40:"C",
54:"0",60:"20010909-01:46:40",64:"20131119",167:"FOR",6215:"1M"})],
{}),
{"1":"FXPV","35":"R","5082":"2","52":"20130924-10:24:37.118","6065":"15",
"75":"20131119","78":[]}])
com.apama.fix.Quote("RFQ","FIX.4.2:APAMA_RFQ_TEST->Bloomberg_FX_TEST",
"EUR/USD_Depth:5082:2:40:C",
"EUR/USD","",{1250:392213.589957,1499:392213.589959}),{"1":"FXPV",
"131":"1","15":"EUR","167":"FOR","38":"1000","40":"C","5082":"2",
"54":"0","60":"20010909-01:46:40","6065":"15","6215":"1M","64":"20131119",
"75":"20131119","78":[]},"RelatedPayload":{"15:"EUR","38:"1000","40:"C\
",54:"0","60:"20010909-01:46:40","64:"20131119","167:"FOR","6215:"1M"}])
```

If business application wants to stop streaming prices, it can send `com.apama.mds.MDServerDepth` with an extraparam 'isCancelled' set true. This will send `QuoteCancel` to Bloomberg.

```
com.apama.fix.QuoteCancel("RFQ","FIX.4.2:APAMA_RFQ_TEST->Bloomberg_FX_TEST",
"*",1,[],{1250:392477.121503,1499:392477.121512}),{"35":"Z","131":"1",
"52":"20131125-13:20:20.452"})
```

Application	FIX Quote Provider	Server
	<-- mds.SubscribeDepth <--	QuoteRequest
	QuoteCancel -->	
<code>mds.MarketDataRequestReject</code>	--> QuoteAcknowledgement -->	
<code>mds.MDServerDepth</code>	--> Quote -->	

7 BM+F Bovespa UMDF Adapter

■ Prerequisites	128
■ Service monitor injection order	128
■ Transport properties configuration	129
■ Configuration for UMDF PUMA 2.0	135
■ Service monitor and session configuration	136
■ subscriptions management	143
■ Instrument static data management	144
■ Status reporting events	146
■ Unified news channel	147
■ Support for UMDF 2.0	148
■ Latency measurement	148

The BVMF_UMDF adapter uses the Apama Integration Adapter Framework (IAF) to connect to BM&F Bovespa Unified Market Data Feed (UMDF) 2.0 platform for both derivatives and equities Segments. The BVMF_UMDF adapter supports the following functionality of the BM&F Bovespa Unified Market Data Feed:

- Market depth
- Trade reports
- Instrument static data (security definitions)

The standard `com.apama.marketdata` and `com.apama.status` event interfaces are supported.

All FIX/FAST message types used by the UMDF feed are supported, although only the following are used directly by the adapter:

- Market Data Snapshot Full Refresh (35=W)
- Market Data Snapshot Incremental Refresh (35=X)
- Security List (35=y)

The following message types are decoded, passed to the correlator and logged, but not currently used by the service monitors:

- News (35=B)
- Security Status (35=f)

All other UMDF message types are only used internally by the IAF transport plugin.

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3
- QuickFast v1.2 library (shipped with the adapter)
- QuickFIX v1.14.3 library (shipped with the adapter)
- Xerces v3.1.0 library (shipped with the adapter)
- FAST template file
- Template xsd file
- Connections Details of the interested feeds

Service monitor injection order

The following monitors need to be injected into the correlator in the listed order to support the BVMF_UMDF adapter.

1. Standard Apama event interface definitions:
 - a. `${APAMA_HOME}/monitors/StatusSupport.mon` (in the Apama "monitors" directory)
 - b. `${APAMA_HOME}/adapters/monitors/IAFStatusManager.mon` (in the Apama "adapters/monitors" directory)
2. Legacy Finance Support interface package
 - a. `${APAMA_FOUNDATION_HOME}/ASB/projects/Legacy Finance Support/Legacy Finance Support.cdp`
3. Support monitors for the BVMF_UMDF adapter:
 - a. `${APAMA_HOME}/adapters/monitors/BVMF_UMDF_DepthUtils.mon`
 - b. `${APAMA_HOME}/adapters/monitors/BVMF_UMDF_DepthPublisher.mon`
 - c. `${APAMA_HOME}/adapters/monitors/BVMF_UMDF_TickUtils.mon`
 - d. `${APAMA_HOME}/adapters/monitors/BVMF_UMDF_TickPublisher.mon`
 - e. `${APAMA_HOME}/adapters/monitors/BVMF_UMDF_StatusPublisher.mon`
4. Event definitions for the BVMF_UMDF adapter:
 - a. `${APAMA_HOME}/adapters/monitors/BVMF_UMDF_IAFEvents.mon`
 - b. `${APAMA_HOME}/adapters/monitors/BVMF_UMDF_ConfigEvents.mon`
 - c. `${APAMA_HOME}/adapters/monitors/BVMF_UMDF_SecurityDefinitionEvents.mon`
5. Service monitors for the BVMF_UMDF adapter:
 - a. `${APAMA_HOME}/adapters/monitors/BVMF_UMDF_SnapshotManager.mon`
 - b. `${APAMA_HOME}/adapters/monitors/BVMF_UMDF_SessionManager.mon`
 - c. `${APAMA_HOME}/adapters/monitors/BVMF_UMDF_MarketDataManager.mon`
 - d. `${APAMA_HOME}/adapters/monitors/BVMF_UMDF_SecurityDefinitionManager.mon`

The `BVMF_UMDF_SecurityDefinitionManager` monitor will attempt to load the `BVMF_UMDF_Plugin` correlator plug-in, which provides support for the cache files used by the monitor. The plug-in library must be in a directory on the correlator's library search path.

Transport properties configuration

Transport name

Replace `"@BVMF_UMDF_TRANSPORT@"` in the `<transport>` element with the name of this transport instance. For example, `"UMDF_ADAPTER"`.

Replace "@BVMF_UMDF_CHANNEL@" in the "channels" attribute of the <source> element at the end of the configuration file with the name of the channel name the user is planning to use, for example "UMDF_CHANNEL".

These two parameters (@BVMF_UMDF_TRANSPORT@ and @BVMF_UMDF_CHANNEL@) have to be provided by the user in the session configuration event.

The Service monitors by default assume the name of the channel to be same as transport name.

Correlator configuration

Replace "@CORRELATOR_HOST@" and "@CORRELATOR_PORT@" with the hostname and listening port of your correlator instance.

Message template file configuration

The FIX/FAST decoder uses a template file supplied by BVMF_UMDF to describe all of the message structures on the FIX/FAST feed. The most recent template file can be downloaded from <ftp://ftp.bmf.com.br/FIXFAST/templates/>.

Replace "@BVMF_UMDF_TEMPLATES_FILE@" in the transport properties with the full path to the template file.

Channel definitions file configuration

Market Data Channels provided by BM&F Bovespa Unified Market Data Feed can be downloaded from http://www.bmfbovespa.com.br/en_us/services/trading/bm-fbovespapuma-trading-system/for-developers-and-vendors/umdf-unified-market-data-feed/.

A Market data channel consists of:

- Market Recovery Feed (UDP connection)
- Incremental Feed (UDP connection)
- Instrument Definition Feed (UDP connection)

This information can be provided to the adapter in the form of an XML configuration file, which is of the following format:

```
<configuration environment="Certification" updated="2010/09/20-21:10:50">
  <channel id="001" label="FX Futures">
    <connections>
      <connection id="001SA">
        <type feed-type="S">Market Recovery</type>
        <protocol>UDP/IP</protocol>
        <ip>233.111.180.96</ip>
        <port>30001</port>
        <feed>A</feed>
      </connection>
      <connection id="001IA">
        <type feed-type="I">Incrementals</type>
      </connection>
    </connections>
  </channel>
</configuration>
```

```

    <protocol>UDP/IP</protocol>
    <ip>233.111.180.97</ip>
    <port>20001</port>
    <feed>A</feed>
  </connection>
  <connection id="001NA">
    <type feed-type="N">Instrument Definition</type>
    <protocol>UDP/IP</protocol>
    <ip>233.111.180.96</ip>
    <port>10001</port>
    <feed>A</feed>
  </connection>
</connections>
</channel>
<channel id="002" label="FX Options">
  <connections>
    <connection id="002SA">
      <type feed-type="S">Market Recovery</type>
      <protocol>UDP/IP</protocol>
      <ip>233.111.180.96</ip>
      <port>30002</port>
      <feed>A</feed>
    </connection>
    .
    .
    .
  </connection>
</connections>
</channel>
</configuration>

```

Things to Know about the XML tags.

Tag	Description
<configuration>	Configuration to be used for the current session. <ul style="list-style-type: none"> ■ environment. (Optional) Configuration environment ■ updated. (Optional) Time when this file is last updated
<channel>	Contains the feed details for the given Market Data Channel. <ul style="list-style-type: none"> ■ id. Unique identifier for the market data channel ■ label. Label identifying the channel description
<connections>	Contains all the feed connection details for the market data channel.
<connection>	Indicates the start of a new connection. <ul style="list-style-type: none"> ■ id. Identifier for the connection
<type>	Indicates the type of the feed. Accepted values for feed-type are: <ul style="list-style-type: none"> ■ S - Indicates Market Recovery Feed ■ I - Indicates Incremental Feed

Tag	Description
	<ul style="list-style-type: none"> ■ N - Indicates Instrument Definition Feed ■ H - Indicates Replay Connection Feed
<protocol>	Indicates the protocol to be used for this feed. TCP/IP or UDP/IP.
<ip>	Indicates the Multicast Feed Address to join. Required for UDP Feeds.
<host-ip>	Indicates the TCP host to connect. Required for TCP Feeds.
<port>	Indicates the port to which the application needs to connect. Applies both for TCP and UDP feeds.
<feed>	If there are multiple feeds relaying the same data, this tag is used to distinguish between them. For example, if a market data channel contains two feeds which are emitting the same data then this can be used to identify between them.

See `sample_config-UMDF.xml` file for basic architecture of the configuration file that needs to be supplied to the adapter.

Replace `@BVMF_UMDF_CHANNEL_FILE@` in the transport properties with the full path to this file. For example:

```
<property name="ConfigFile" value="sample_config-UMDF.xml" />
```

FIX session properties

The adapter utilizes the QuickFIX library for establishing the FIX session over the TCPReplay Feed.

Configure the FIX session as follows:

```
QuickFIX.<FIX_PROPERTY>
  or
UMDFChannel.<FEED_ID>.<FIX_PROPERTY>
```

where,

- `FIX_PROPERTY` is the FIX configuration property that needs to be provided to the QuickFIX library. A full list of FIX configuration properties can be found at <http://www.quickfixengine.org/quickfix/doc/html/configuration.html>.
- `FEED_ID` is the feed for which the configuration is being provided. Valid values are 'A', 'B' etc. This signifies the Primary Feed versus Secondary Feed.

Priorities among various ways of providing FIX Session Properties (Highest to Least)

1. `UMDFChannel.<FEED_ID>.<FIX_PROPERTY>`
2. `UMDFChannel.*.<FIX_PROPERTY>`

3. QuickFIX.<FIX_PROPERTY>

Example:

```
<property name="QuickFIX.ConnectionType" value="initiator" />
<property name="QuickFIX.ReconnectInterval" value="@RECONNECT_INT@" />
<property name="UMDFChannel.A.SenderCompID" value="@FIX_SENDER_COMPID_A@" />
<property name="UMDFChannel.B.SenderCompID" value="@FIX_SENDER_COMPID_B@" />
```

- **QuickFIX.FileStorePath.** This parameter controls logging of persisted FIX messages (used for replay and gap recovery). This property must be specified and does not point to invalid path. Takes relative as well as absolute paths. For example:
 - **Windows(Absolute).** `D:\XYZ\FIX_Log`. Directory `FIX_Log` will be created if not exists if path `D:\XYZ` exists.
 - **Windows(Relative).** `Fix_Log`. Directory `FIX_Log` will be created in the current directory from where the IAF instance started if directory does not exists.
- **QuickFIX.FileLogPath.** This parameter controls logging of all incoming and outgoing FIX messages. If this property is not mentioned, the logging is disabled. It accepts relative as well as absolute paths.
- **QuickFIX.SendBufferSize** and **QuickFIX.ReceiveBufferSize.** These properties can be used to set the QuickFIX TCP socket send/receive buffer sizes to the number of bytes specified. It defaults to OS specific. For example:

```
<property name="QuickFIX.SendBufferSize" value="1024" />
<property name="QuickFIX.ReceiveBufferSize" value="1024" />
```

- **Static fields:** Allows static values to be defined for outgoing (upstream) messages. The definition specifies whether the field is inserted into the header or the body, the message type and the tag number. Wildcards (*) can be specified for the message type and the tag number. `StaticField.[header|body].[msg].[tag]`. Examples:

```
<property name="StaticField.body.A.141" value="Y"/>
```

- The example sets body tag 141 in message type A (Logon) to the value Y.

Additional transport properties

The following additional transport configuration properties are available:

- **Additional configuration file.** BM&F Bovespa UMDF provides the option of switching to a backup feed in case of primary feed failure. You can enable this functionality in the adapter by providing an additional configuration file with the connection details for the backup feed. For example:

```
<property name="ConfigFile" value="config_backup_feed.xml" />
```

Note:

The adapter can handle multiple instances of `ConfigFile` property name.

- `EnableFixLogging`. If enabled, all the decoded fix messages will be logged to the folder mentioned by `FIXLogFolder` transport property. If `FIXLogFolder` is not specified, the messages are logged to `UMDF_FIX_LOG`. For example:

```
<property name="EnableFixLogging" value="true" />
```

Note:

It is not recommended to enable FIX logging in production environments as it increases the latency considerably.

- `FIXLogFolder`. Only useful when `EnableFIXLogging` transport property is set to true. Fix messages logs are stored in the folder mentioned this parameter. For example:

```
<property name="FIXLogFolder" value="UMDF_FIX_LOG" />
```

- `ConvertNativeStrings`. If true, any `STRING`-type fields in `FAST` decoded messages should be converted from the native character encoding to `UTF8`. If false, all strings are assumed to be `UTF-8`. This value defaults to false. For example:

```
<property name="ConvertNativeStrings" value="true" />
```

- `NativeEncoding`. If non-empty, this is the name of the native encoding to use for conversions to/from `UTF-8`. This parameter overrides the system encoding. For example:

```
<property name="NativeEncoding" value="ISO8859-1" />
```

- `MulticastInterface`. IP address (as a dotted quad, for example, "1.2.3.4") of the network interface on the machine running the adapter that should be used to subscribe to `UDP` multicast traffic. If not specified, all interfaces will be used. For example:

```
<property name="MulticastInterface" value="1.2.3.4" />
```

- `LogMessages`. FIX message that should be logged when they are received from `UMDF`. The message contents will be logged at `INFO` level. The following message types are supported:

- `Heartbeat`
- `SequenceReset`
- `News`
- `SecurityStatus`
- `MarketDataFullRefresh`
- `MarketDataIncrementalRefresh`
- `SecurityDefinition`
- `TCPReplayMessages` (used to log the `TCPReplay Session` FIX messages at `INFO`)
- `Unknown` (any unrecognised message type)

For example:

```
<property name="LogMessages" value="Unknown" />
```

- **LogDebug.** A comma separated list of extra debug logging categories that should be enabled. The additional logging produced by an enabled category will be logged at the INFO level unless otherwise stated. Currently the following categories are supported:
 - **UpstreamEvents.** Enable logging of all events sent to the adapter by the correlator ("upstream" events). These will include subscription and unsubscription requests, commands to enable and disable the feed of security definitions, and heartbeat events.
 - **SockReader.** Enable detailed logging on all socket reader. This should be disabled by default for performance reasons.
 - **EventConstruction.** Detailed logging of the transformation of an incoming FIX message into an Apama event.
 - **ChannelMessageRate.** Log the number of raw FIX/FAST update messages, received on each channel, every 60 seconds.
 - **Recovery.** Enable detailed logging in regions related to recovery procedure (initial or recovery after Gap). This should be disabled by default for performance reasons.
 - **ThreadStates.** Logs information about the start and stop of threads.

For example:

```
<property name="LogDebug" value="ChannelMessageRate" />
```

Note:

"LogMessages" and "LogDebug" can produce extreme amounts of output and should be used with caution in a production environment.

- **TcpFeedRecoveryLimit.** By this parameter, you can set the limit for the number of messages that can be recovered using TCP Replay. The default is set to 2000. For example:

```
For PUMA 2.0
<!-- Max value supported for UMDF 2.0 -->
<property name="TcpFeedRecoveryLimit" value="10000" />
```

Configuration for UMDF PUMA 2.0

UMDF PUMA uses a single TCP Replay Session instead of separate FIX Sessions for each individual channel. So users can provide the session properties using the global `QuickFIX.<property>` identifier. Since the IP/Port for the TCP Replay Feed are common for all the channels, you can choose not to include the TCP replay feed details in the channel configuration file and use `QuickFIX.SocketConnectHost` and `QuickFIX.SocketConnectPort` instead.

```
<property name="ConnectToNonPUMAFed" value="false" />
<property name="QuickFIX.SenderCompID" value="@FIX_SENDER_COMPID@" />
<property name="QuickFIX.TargetCompID" value="@FIX_TARGET_COMPID@" />
<property name="QuickFIX.SocketConnectHost" value="@FIX_HOST@" />
<property name="QuickFIX.SocketConnectPort" value="@FIX_PORT@" />
<!-- Username and Password are not required in the certification environment-->
<property name="StaticField.header.A.553" value="@USERNAME@" />
<property name="StaticField.header.A.554" value="@PASSWORD@" />
```

Market data channels:

- For certification: http://www.bmfbovespa.com.br/pt-br/servicos/download/MarketDataChannels_PUMA_CERT.pdf
- For production: http://www.bmfbovespa.com.br/pt-br/servicos/download/MarketDataChannels_PUMA_PROD.pdf

Template file location:

- For certification: <ftp://ftp.bmf.com.br/FIXFAST/templates/Certification/templates-PUMA.xml>
- For production: <ftp://ftp.bmf.com.br/FIXFAST/templates/Production/templates-PUMA.xml>

Service monitor and session configuration

The BVMF_UMDF service monitors must be configured by routing the following event after all of the monitors are injected:

```
event com.apama.bvmf_umdf.SessionConfiguration
{
  string connection;
  dictionary<string,string> configuration;
}
```

The "connection" field gives the name of the BVMF_UMDF session. This must match the transport name in the IAF configuration file.

The "configuration" dictionary specifies configuration settings for the session. Currently the following keywords are supported:

Parameter	Type	Description
channel	string Default: Transport name	Channel on which the IAF is configured to listen.
logFile	string Default: stdout	Path to log file for the BVMF_UMDF service monitors.
logLevel	string Default: INFO	Logging level for the BVMF_UMDF service monitors. Any level known to the Apama LoggingManager plugin may be used.
logAppend	boolean Default: True	If set to "true", the log file will be appended to rather than being truncated when the adapter starts.
heartbeatInterval	float Default: 5.0	Rate at which the heartbeats have to be sent to the transport.

Parameter	Type	Description
reconnectInterval	integer Default: 60	Time before which the adapter should try to reconnect to the backup feed if the primary feed fails to send any data.
updateLogging	boolean Default: False	If set to "true", detailed information on incoming market data snapshot and updates for all subscribed symbols will be logged to the session log file. Note that these log messages will only be visible if the updateLoggingLevel parameter is also set to a level greater than or equal to the logLevel of the session.
updateLoggingLevel	string Default: DEBUG	Logging level for detailed logging of all incoming market data messages for subscribed symbols. If this is equal to or greater than the logLevel, the extra logging will appear in the log file. Note that this should only be enabled for debugging purposes with a small set of symbols, as the extra logging can be extremely verbose.
aggregatePrices	boolean Default: True	Market Data received from UMDF can be of either price depth, order depth book or top of the book. If aggregatePrices is set to "true", market data of instruments for which order depth book (MBO) is being received will be published as price depth book(MBP)
publishDepthSourceKeys	boolean Default: False	If set to "true", com.apama.marketdata.Depth events generated by the adapter will contain extraParams keys identifying the "source" (that is, the outright or implied book) of each price level.
publishUpdateInfoKeys	boolean Default: True	If set to "true", com.apama.markertdata.Depth and Tick events will include extraParams keys identifying the "source" of each update, that is the channel, update type and message sequence numbers: Channel. FIX/FAST channel number

Parameter	Type	Description
		<p>UpdateType. "S" for Snapshot, "I" for Incremental, "G" for Gap, "F" for Security Status update</p> <p>SeqNum. Per-channel message sequence number</p> <p>RptSeq. Highest per-instrument update sequence number that contributed to this event</p> <p>RptSeq0. Lowest per-instrument update sequence number that contributed to this event</p> <p>LastMsgSeq. (Snapshots only) highest incremental message sequence number that contributed to the contents of the snapshot message</p>
publishExtraFIXKeys	boolean Default: True	If set to "true", <code>com.apama.markertdata.Depth</code> and <code>Tick</code> events will include additional FIX tags from the FIX/FAST messages used to generate the events, in the <code>extraParams</code> dictionary. The set of extra tags that will be included is subject to change.
publishEmptyDepthOnGap	boolean Default: True	If set to "true", a <code>com.apama.marketdata.Depth</code> event with no bid or ask price levels will be published for all effected symbols when a sequence gap is seen on the FIX/FAST feed. Note that regardless of the setting of this parameter, receipt of a gap report is signalled by publishing both <code>com.apama.marketdata.DepthSubscriptionError</code> and <code>com.apama.marketdata.TickSubscriptionError</code> events for all affected subscriptions as described in the next section. Note also that setting this parameter to "false" does not prevent empty bid or ask books generated during normal operation of the market from being published.
publishMarketStatistics	boolean Default: False	If set to "true", market statistics will be added to the <code>extraParams</code> dictionary of all

Parameter	Type	Description
		the Tick events and the Depth events constructed from market snapshots published to applications. The following set of extraParams keys is used:
		IndexValue - Data related to indexes and ETFs
		Open - The opening price of the security (first trade)
		TheoreticalOpeningPrice - Theoretical Opening Price
		TheoreticalOpeningQuantity - Theoretical Closing Price
		Close - The closing price of the security (previous day's last trade)
		Settle - The settlement price of the security.
		High - The highest price traded for the security in the trading session.
		Low - The lowest price traded for the security in the trading session
		VWAPPrice - the ratio of the value traded to total volume traded over the trading session
		ImbalanceBuyers - Imbalance Buyers
		ImbalanceSellers - Imbalance Sellers
		TradeVolume - The total volume traded for that security in the trading session
		OpenInterest - Total number of contracts in a commodity or options market that are still open; that is, they have not been exercised, close out, or allowed to expire.
		PriceBandHigh - Price Band High Information
		PriceBandLow - Price Band Low Information

Parameter	Type	Description
		<p>TradingReferencePrice - Trading Reference Price</p> <p>NetChange - Net change from previous day's closing price versus last traded price.</p> <p>TotalTradedQuantity - Total traded quantity (shares, contracts, exercised contracts) of the trading day</p> <p>AvgDailyTradedQty - Daily average shares traded within 30 days(Quantity band Information).</p> <p>MaxTradeVol - The maximum order quantity that can be submitted for a security.(Quantity band)</p> <p>UnderlyingPrice - To inform the price composition of BDR indexes.(Composite Underlying Price)</p>
PublishExtraStatistics		<p>Except Market statistics, if user needs any of the extra fields delivered by the feed to the Depth and Tick events then you have to set this parameter with MDEntryType followed by Tags. Asterisk (*) symbol also for all parameters. Example:</p> <pre>MDEntryType<space>Tags, MDEntryType<space>Tags... com.apama.bvmf_umdf.SessionConfiguration ("UMDF_TRANSPORT",{ "channel": "UMDF_CHANNEL, ""PublishExtraStatistics": "B 270 271, g 6939 83, C *"})</pre>
recordTimestamps	<p>boolean</p> <p>Default: False</p>	<p>If set to "true", the service monitors will record microsecond resolution timestamps for incoming Snapshot and Incremental events and add these to <code>__timestamps</code> parameter in the <code>extraParams</code> dictionaries of all Depth and Tick events sent to clients. <code>__timestamps</code> is a string dictionary containing the following key:value timestamp pairs:</p> <p>1000 - time that the message was read from the network (if supplied by the transport)</p>

Parameter	Type	Description
		<p>1249 - time that the event was ready to send to the correlator (if supplied by the transport)</p> <p>5200 - time that the event reached the monitors</p> <p>5250 - time that the event was ready to publish</p> <p>Note that these timestamps are relative to an unspecified epoch so should only be used for latency calculations on a single machine. Note also that actual publication of trade updates may be delayed until the end of a trade "event" on the feed is detected. This delay is not reflected in the 5250 timestamp value but can easily be calculated by an application.</p>
logLatency	boolean Default: False	If set to "true", and timestamp recording has been enabled both in the transport and in the service monitors, the end-to-end latency (value@5250 - value@1000) will be calculated and logged at INFO level for all Depth and Tick events published to clients.
dropCrossedBooks	boolean Default: False	If set to "true", prevents all Depth events having best bid price >= best offer price from being routed.
publishSecurityStatus	boolean Default: False	If set to "true", publishes "SecurityTradingStatus" and "SecurityTradingPhase" updates in the extraParams of the depth event.
logNewsMessages	boolean Default: False	Set this to "true" to logs News messages.
ignoreRetransmittedUpdates	boolean Default: True	Ignore incoming Incremental Updates with QuoteCondition set as being Retransmitted.
forwardSecurityId	boolean Default: False	If set to "true", forwards the SecurityID used in the UMDF session in the Depth and Tick extraParams as "SecurityId".

Parameter	Type	Description
useContractMultiplier	boolean Default: False	If set to "true", The ContractMultiplier provided in SecurityList (Instrument Definition) is used to multiply the quantity values in Depth and Tick updates.
usePriceDivisor	boolean Default: True	If set to "true", The PriceDivisor provided in SecurityList (Instrument Definition) will be used to divide the Price field to produce the actual order price.
publishBustTrades	boolean Default: True	If set to "true", a Tick update will be published in the event of a trade bust with "tradeBust":"true" in extraParams. Else the update will be discarded.
publishSortedOrderBook	boolean Default: False	If set to "true", the adapter publishes order book based on prices that is, Decreasing Order on bidSide and Increasing Order on askSide. This is not a suggested configuration if the instrument is in Auction state or Close state where no prices are received from the exchange.
maxDepthLevel	integer Default: 0	Number of price levels to be present in the depth update. If set to 0, publishes all the levels received from the exchange.
useMarketDepthFromSnapshot	boolean Default: False	MarketDepth for an instrument can be retrieved using the SecurityList message and also by using the Snapshot message. If set to "true", the MarketDepth present in the snapshot message will be used as the size of the orderbook.
		Note: maxDepthLevel overwrites this value.
forwardBrokerInformation	boolean Default: False	If set to "true", publish the Broker information (that is Broker code,time) of MDEntryType (Bid/Ask) in the extraParams of the depth event. The following tag are Broker information of MDEntryType(Bid/Ask): <ul style="list-style-type: none"> ■ Tag 288 -> MDEntryBuyer ■ Tag 289 -> MDEntrySeller

Parameter	Type	Description
		<ul style="list-style-type: none"> ■ Tag 273 -> MDEntryTime ■ Tag 272 -> MDEntryDate ■ Tag 37016 -> MDInsertDate ■ Tag 37017 -> MDInsertTime
excludeStreamIdFromMarketStats	boolean Default: False	Set to "true" for Connecting to Feeds prior to UMDF 2.0. This will disable publishing of market statistics based on StreamID, and will only publish statistics for the default stream ("E" - Electronic).

Note:

If you want to minimise the size of Depth and Tick events by eliminating extraParams that are not required for a given application should set the "publishDepthSourceKeys", "publishUpdateInfoKeys" and "publishExtraFIXKeys" session parameters to "false" as required.

Sample configuration event

```
com.apama.bvmf_umdf.SessionConfiguration("UMDF_ADAPTER", {"channel":"UMDF_CHANNEL"})
```

subscriptions management

subscriptions to market depth are made by sending the correlator a standard `com.apama.marketdata.SubscribeDepth()` event. For example:

```
com.apama.marketdata.SubscribeDepth("BVMF_UMDF", "UMDF_ADAPTER", "DTCY1", {} )
```

For Tick subscription:

```
com.apama.marketdata.SubscribeTick("BVMF_UMDF", "UMDF_ADAPTER", "DTCY1", {} )
```

- The `serviceId` should always be "BVMF_UMDF"
- The `marketId` is the session name from the `SessionConfiguration` event
- The `symbol` should be from Bovespa UMDF

subscriptions to market depth or trade reports are cancelled by sending the correlator standard `com.apama.marketdata.UnsubscribeDepth` or `com.apama.marketdata.UnsubscribeTick` events. For example:

```
com.apama.marketdata.UnsubscribeDepth("BVMF_UMDF", "UMDF_ADAPTER", "ZLZ9", {})  
com.apama.marketdata.UnsubscribeTick("BVMF_UMDF", "UMDF_ADAPTER", "ZLZ9", {})
```

Market depth is reported in `com.apama.marketdata.Depth` events. Several `extraParams` keys are used to annotate the depth data, if the `publishDepthSourceKeys` session parameter is set to true:

- "Bid1"- "Bid5" - These keys can take several different values:
 - "A" - the price at this bid level includes actual orders
 - "I" - the price at this bid level includes implied orders
 - "AI" - the price at this bid level includes actual and implied orders
- "Ask1"- "Ask5" - As for the corresponding "BidN" keys.

Additional subscription parameters

channelID. Channel to which the subscription has to be made to get the market data.

With the release of UMDF PUMA , a single MarketData channel is split in to multiple channels with each channel dissipating data which is either Market By Order (MBO), Market By Price (MBP) or Top Of Book (TOB). But the instrument definition feed across all these three channels remain the same. So if you want to subscribe for a particular kind of MarketData, you must provide an additional identifier in the subscription Request.

```
com.apama.marketdata.SubscribeDepth("BVMF_UMDF","UMDF_TRANSPORT","BBDC4",{ "ChannelID": "50" })
com.apama.marketdata.SubscribeDepth("BVMF_UMDF","UMDF_TRANSPORT","ALLL3",{ "ChannelID": "150" })
com.apama.marketdata.UnsubscribeDepth("BVMF_UMDF","UMDF_TRANSPORT","BBDC4",{ "ChannelID": "50" })
com.apama.marketdata.UnsubscribeDepth("BVMF_UMDF","UMDF_TRANSPORT","ALLL3",{ "ChannelID": "150" })
```

Note:

The user has to make sure that the Security Definitions for the corresponding channels are configured properly using SecurityDefinitionChannelConfiguration event.

Instrument static data management

Security Definition records are managed on a per-channel basis by the BVMF_UMDF_SecurityDefinitionManager service monitor. The monitor is controlled and configured by the following configuration event (in package com.apama.bvmf_umdf):

```
event SecurityDefinitionChannelConfiguration {
  string connection;
  integer channel;
  dictionary<string, string> configuration;
  // actions not shown here
}
```

One of these events should be sent to the correlator to initialise the security definition handler for each channel of interest. The "connection" field should match the session name from the SessionConfiguration event, and the channel should match a ChannelID in the IAF configuration file.

The "extraParams" field recognises the following keys and values:

- "shutdown" (boolean, default is false): If true, delete the handler for this channel.
- "refreshNow" (boolean, default is false): If true, request an immediate refresh of security definitions on this channel from the adapter transport.

- "loadNow" (boolean, default is true on channel creation, false for subsequent configurations): If true, immediately reload the security definitions for the channel from the cache file.
- "saveNow" (boolean, default is false): If true, immediately save the security definitions for the channel to the cache file.
- "enableAutoRefresh" (boolean, default is false): If true, enable scheduled automatic refresh of the security definitions for the channel. Only has any effect if one of "refreshAt" or "refreshEvery" is also specified.
- "refreshAt" (string, default is ""): A string with 5 comma-separated elements to be used as the arguments to an EPL "on all at()" listener(minute,hour,day,month,dayofweek). The listener will request a refresh from the network every time it fires. For example, to refresh every day at midnight, use "0,0,*,*,*".
- "refreshEvery" (float, default is 0.0): A number of seconds to wait between automatic refresh requests. Ignored if "refreshAt" is also specified.
- "enableCache" (boolean, default is true): Enable loading/saving of security definitions for the channel from/to a local cache file.
- "cacheFile" (string, default is <connection>_<channel>_SecurityDefinition_CACHE.dat): The name of the local cache file for security definitions from this channel.

A channel may be reconfigured at any time by sending another `SecurityDefinitionChannelConfiguration` event with the same connection and channel fields, and a different set of extraParams keys/value. For example, the security definition handlers for channels 1 (FX Futures) and 55 (Stock Indexes and ETFs) might be configured as follows:

```
com.apama.bvmf_umdf.SecurityDefinitionChannelConfiguration
  ("UMDF_ADAPTER",1,{"refreshNow":"true","enableAutoRefresh":"true",
    "refreshAt":"0,0,*,*,*"})
com.apama.bvmf_umdf.SecurityDefinitionChannelConfiguration
  ("UMDF_ADAPTER",55,{"enableAutoRefresh":"true","refreshEvery":"3600.0",
    "cacheFile":"55.dat"})
```

Once configured, the handler can be queried for security definitions using the following event (again, in the `com.apama.bvmf_umdf` package):

```
event SecurityDefinitionRequest {
  string connection;
  integer channel;
  string symbol;
  dictionary<string, string> extraParams;
}
```

The "connection" and "channel" parameters are the same as for the channel configuration. The "symbol" field holds the symbol description (FIX tag 107) for the security. The "extraParams" field holds any additional data needed for the lookup, although currently no keys are defined for this event. Each lookup request will route a `com.apama.bvmf_umdf.SecurityDefinitionResponse` event in reply:

```
event SecurityDefinitionResponse {
  string connection;
  integer channel;
```

```
string symbol;
boolean found;
SecurityDefinition record;
dictionary<string, string> extraParams;
}
```

The fields of this event will be identical to the request except that the "record" field - holding the returned security definition - is added, and the "found" field is set to "true" if the requested record was found. If the record was *not* found, "found" will be "false" and the "record" field should be ignored.

To make it easier to write monitors that use the security definition manager, one of the following events will be routed whenever a load, refresh or load operation completes, respectively:

```
event SecurityDefinitionLoadComplete {
    string connection;
    integer channel;
}
event SecurityDefinitionRefreshComplete {
    string connection;
    integer channel;
}
event SecurityDefinitionSaveComplete {
    string connection;
    integer channel;
}
```

The "connection" and "channel" fields will be identical to those used in the configuration event.

Status reporting events

Status reporting uses the standard `com.apama.statusreport` event interface, defined in the `StatusSupport.mon` file. Heartbeats and status messages between the service monitors and the `transport/codec` are managed by the `IAFStatusManager.mon` service monitor. Both of these monitors must be injected in the order listed in the "Monitor injection order" section.

Applications should subscribe to status by sending the following event to the correlator:

```
com.apama.statusreport.SubscribeStatus("BVMF_UMDF", <object>, "", <connection>)
```

Where *connection* is the connection name as used in the `SessionConfiguration` event, and *object* is `Adapter`, `Channel`, or `SecurityDefinition`.

subscriptions where all fields are "wildcards" are also supported:

```
com.apama.statusreport.SubscribeStatus("", "", "", "")
```

In this case, the subscriber will receive all status reports from the `BVMF_UMDF` adapter as well as every other status-reporting entity in the system. A `com.apama.statusreport.UnsubscribeStatus` event with the same field values can be used to cancel a previous status subscription.

Status reports are routed as `com.apama.statusreport.Status` objects, with the "serviceId", "object" and "connection" fields set as above for subscriptions. The reported status is as follows:

- **Adapter** object. This object reports on the overall state of the connection between the feed, transport and service monitors. If the "available" flag is true, the adapter is "up", in contact with the feed and ready to handle requests. The value "Connected" will appear in the "summaries" field in this case. Otherwise, the adapter is unavailable and clients should wait until it becomes available before trying to use it.
- **Channel** object. This object reports on the status of an individual market data channel, based on the content of any SecurityStatus messages received from the feed for that channel (as distinct to the status of a security). The "summaries" sequence will contain a single element giving the status of the channel, and the "available" field will be true if the status indicates that the channel is "ready to trade".
- **SecurityDefinition** object. This object reports on the status of a security definition channel handler. The following values appear in the "summaries" sequence:
 - Load Started
 - Load Completed
 - Load Failed
 - Refresh Started
 - Refresh Completed
 - Refresh Failed
 - Save Started
 - Save Completed
 - Save Failed
 - Ready

The "available" field will be true if the channel handler is ready to handle lookup requests for security definition records.

Unified news channel

UMDF provides a separate Unified news channel reserved for global news broadcast that will be used to send encoded headlines and text. You can subscribe to this channel using `com.apama.bvmf_umdf.SubscribeUnifiedNews` event.

```
event SubscribeUnifiedNews{
  string TRANSPORT;
  integer channel;
  dictionary<string,string> __payload;
}
```

This news will be published as the `com.apama.bvmf_umdf.News` event. You have to listen to this event to get the updates.

```
event News {
  string TRANSPORT;
```

```
integer channel;
integer MsgSeqNum;
string Headline;
sequence <News_RelatedSym> NoRelatedSym;
sequence <News_RoutingIDs> NoRoutingIDs;
sequence <News_LinesOfText> NoLinesOfText;
dictionary<integer,float> __timestamps;
dictionary<string,string> __payload;
}
```

Support for UMDF 2.0

Starting from UMDF v2.0 support, all real-time statistics are stored separately based on venue (Stream ID). The following set of MDStreamID (tag 1500) are used:

- E - Electronic
- X - Ex-pit
- O - Option Exercise
- T - Termo
- N - Index
- L - Lending (BTC)
- A - All

The default value is E.

Each market data statistics will appear as: MDStreamID + "_" + "statisticsType".

MDEntryType "D"(Composite Underlying Price) statistics appear as: MDStreamID + "_" + "UnderlyingSecurityID" + "_" + "statisticsType".

Example:

```
com.apama.marketdata.Depth("ALLL3",*,*,{"E_AvgDailyTradedQty":"28",
"E_Close":"8.96","E_1_UnderlyingPrice":"1.232",
"E_TotalTradedQuantity":"1634500", "T_1_UnderlyingPrice":"1.232",
"Market":"BVMF_UMDF_TRANSPORT", "T_AvgDailyTradedQty":"18",
"T_MaxTradeVol":"28","UpdateType":"I"})
```

```
com.apama.marketdata.Tick("ALLL3",*,*,{"E_AvgDailyTradedQty":"28",
"E_Close":"8.96","E_High":"9.08","E_TotalTradedQuantity":"1634500",
"Market":"BVMF_UMDF_TRANSPORT","T_AvgDailyTradedQty":"18",
T_MaxTradeVol":"28","UpdateType":"I"})
```

Latency measurement

The transport supports the latency measurement framework. For information on configuring the transport to record high-resolution timestamps on downstream events, see "The IAF configuration File" section in *Connecting Apama Applications to External Components*.

8 BM+F FIX Adapter

■ Prerequisites	150
■ IAF configuration	150
■ Service monitor extensions	150
■ Service monitor injection order	151
■ Session configuration for BMF Support monitors	151
■ Session configuration	152
■ Sample SymbolSet event	154
■ Marketdata subscription or unsubscriptions	154
■ Placing order	154
■ Sample events	156
■ New EntryPoint system	156

The BMF FIX system deviates from standard FIX in a number of ways which have made it necessary to provide a number of extensions to the FIX adapter. The extensions fall into two areas: IAF configuration and service monitor extensions.

Bell FIX is being deprecated by Bovespa, Customers should migrate to BMF EntryPoint.

Apama BME EntryPoint FIX adapter is certified for BME EntryPoint Equities/Derivatives 2.0.8.

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

IAF configuration

When acting as a client to BMF, the IAF must be configured with the BMF specific configuration file, the distribution version of which can be found here: `config/FIX-BMF.xml.dist`

Note that the requirement for an BMF-specific configuration file means that an IAF using this configuration file must be used for BMF FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for BMF operation. If you need to connect to other FIX servers as well as BMF, the non-BMF sessions should be configured in a separate IAF instance.

This configuration file differs from the standard file in that it defines a number of new events. In addition, it provides the following static FIX tags that are necessary for interaction with the BMF server:

```
<!-- RawData (BMF Password) -->
<property name="StaticField.body.A.96" value="<BMF Password>"/>
<!-- RawDataLength -->
<property name="StaticField.body.A.95" value="<BMF Password Length>"/>
<!-- SubscriptionRequestType -->
<property name="StaticField.body.x.263" value="1"/>
<property name="StaticField.body.V.263" value="1"/>
<!-- SecurityIDSource -->
<property name="StaticField.body.D.22" value="8"/>
<property name="StaticField.body.F.22" value="8"/>
<property name="StaticField.body.G.22" value="8"/>
<!-- TimeInForce -->
<property name="StaticField.body.D.59" value="0"/>
<!-- TradeAllocIndicator -->
<property name="StaticField.body.D.826" value="1"/>
```

Service monitor extensions

The file `FIX_BMF_Support.mon` provides a Market data and Order preprocessing support.

This monitor retrieves the requests to subscribe to map the IDs by Symbol.

When receives a market data update this monitor retrieves the correct ID for the subscription and passes to the primary monitor.

Therefore, the following EPL files must be injected into the correlator, after the standard FIX adapter service monitors:

```
FIX_EventViewer.mon
```

```
FIX_BMF_Support.mon
```

The events defined in the BMF configuration file fall into two packages and therefore must be placed into two separate event files which must both be injected in the order shown below:

```
FIX_Events.mon
```

```
FIX_BMF_Events.mon
```

Service monitor injection order

Inject the monitors listed in “[FIX order management session](#)” on page 82, followed by:

- `${APAMA_HOME}/adapters/monitors/FIX_SecurityListManager.mon`
- `${APAMA_HOME}/adapters/monitors/FIX_BMF_Events.mon`
- `${APAMA_HOME}/adapters/monitors/FIX_BMF_Support.mon`

Session configuration for BMF Support monitors

First `com.apama.fix.bmf.SessionConfiguration` event must be sent to correlator:

Preprocessing of orders and market data updates are handled in support service monitors so user needs to send the following configuration event for initialising the corresponding sessions with respect to global session configuration event.

Here is the configuration event definition.

```
event com.apama.fix.bmf.SessionConfiguration {
  string marketDataConnection;
  string orderDataConnection;
  dictionary<string,string> configuration;
}
```

If Orders and Market data processing is done in single session then mention the same for fields "marketDataConnection" and "orderDataConnection" of the BMF configuration event.

For example `com.apama.fix.bmf.SessionConfiguration("BMF_TRANSPORT","BMF_TRANSPORT", {})`

If the market names are different:

```
com.apama.fix.bmf.SessionConfiguration("BmfFixTransportMD","BmfFixTransportOMS", {})
```

If more than one MD or OMS is needed, send:

```
com.apama.fix.bmf.SessionConfiguration("BmfFixTransportMD2","BmfFixTransportOMS2", {})
```

Session configuration

com.apama.fix.SessionConfiguration event might have following fields:

SecurityList.CreateSecurityList = true sets this parameter parameter will requests for the SecurityList and cache all the securities and their corresponding data for each security available in exchange.

SubscriptionManager.RepeatingGroupTags = [146 55 48 22 207]

SubscriptionManager.UseDefaultDepthEntryTypes = false

SubscriptionManager.AdditionalMDEntryTypes = "e f" (for BMF Book Aggregation)

SubscriptionManager.AdditionalMDEntryTypes = "Z X" (for Top of Book entries)"e f" (for BMF Book Aggregation)

SecurityList.RequestKeyID = BMF_TRANSPORT or some unqiue key

SubscriptionManager.SubscriptionKeyTags = "22 207" . It is the Key for SecurityId requests from user sessions. If user creating separate Market data session and Order manager session then both sessions can use the same store to get the SecurityId definitions instead of both populating and caching the security IDs.

Note:

SecurityId requests are listened on key supplied in this parameter "SecurityList.RequestKeyID". So user must set this field to some unique keyID (preferably MarketId of the Market data session).

SubscriptionManager.SecDefRequestTags = 48

OrderManager.SecDefRequestTags = 48

SubscriptionManager.MDupdateInPlace = true to enable in place update of Market data processing. Position based updates are applied in place.

SubscriptionManager.MaxHardDepthLevel = 0 to disable restriction of max number of book entries (default)

SubscriptionManager.MaxHardDepthLevel = 0 if MarketDepth(264) is available in MarketDataSnapshot message, the max number of book entries it set to it. Else, the number of book entries is set to the value specified here.

Note:

This parameter is used to support Price-depth Bottom Row Handling in BMF. This has been deprecated. Use SubscriptionManager.UseBMFBottomRowHandling instead.

SubscriptionManager.UseBMFBottomRowHandling = false to disable restriction of max number of book entries (default)

SubscriptionManager.UseBMFBottomRowHandling= true if MarketDepth(264) is available in MarketDataSnapshot message, the max number of book entries it set to it.

For example,

```
com.apama.fix.SessionConfiguration("BMF_TRANSPORT",
  {"FixChannel":"BMF_FIX", "SecurityList.CreateSecurityList":"true" ,
   "SecurityList.RequestKeyID":"BMF_TRANSPORT",
   "SubscriptionManager.RepeatingGroupTags":"[146 55 48 22 207]",
   "SubscriptionManager.SecDefRequestTags":"48",
   "OrderManager.SecDefRequestTags":"48",
   "SubscriptionManager.MDupdateInPlace":"true",
   "SubscriptionManager.SubscriptionKeyTags":"22 207"})
```

or with Aggregation

```
com.apama.fix.SessionConfiguration("BMF_TRANSPORT", {"FixChannel":"BMF_FIX",
  "SecurityList.CreateSecurityList":"true",
  "SecurityList.RequestKeyID":"BMF_TRANSPORT",
  "SubscriptionManager.RepeatingGroupTags":"[146 55 48 22 207]",
  "SubscriptionManager.SecDefRequestTags":"48",
  "OrderManager.SecDefRequestTags":"48",
  "SubscriptionManager.MDupdateInPlace":"true",
  "SubscriptionManager.UseDefaultDepthEntryTypes":"false",
  "SubscriptionManager.AdditionalMDEntryTypes":"ef",
  "SubscriptionManager.SubscriptionKeyTags":"22 207"})
```

In case of different market names for MD and OMS:

```
# MarketData Session with BMF Book Aggregation
com.apama.fix.SessionConfiguration("BmfFixTransportMD",
  {
  "FixChannel":"BmfFixChannel",
  "SecurityList.CreateSecurityList":"true",
  "SecurityList.RequestKeyID":"BmfSecurityListRequestKeyID",
  "SubscriptionManager.RepeatingGroupTags":"[146 55 48 22 207]",
  "SubscriptionManager.SecDefRequestTags":"48",
  "SubscriptionManager.MDupdateInPlace":"true",
  "SubscriptionManager.UseDefaultDepthEntryTypes":"false",
  "SubscriptionManager.AdditionalMDEntryTypes":"e f"
  })
# MarketData Session without BMF Book Aggregation
com.apama.fix.SessionConfiguration("BmfFixTransportMD",
  {
  "FixChannel":"BmfFixChannel",
  "SecurityList.CreateSecurityList":"true",
  "SecurityList.RequestKeyID":"BmfSecurityListRequestKeyID",
  "SubscriptionManager.RepeatingGroupTags":"[146 55 48 22 207]",
  "SubscriptionManager.SecDefRequestTags":"48",
  "SubscriptionManager.MDupdateInPlace":"true"
  })
# OMS Session
com.apama.fix.SessionConfiguration("BmfFixTransportOMS",
  {
  "FixChannel":"BmfFixChannel",
  "SecurityList.RequestKeyID":"BmfSecurityListRequestKeyID",
  "OrderManager.IgnoreStatusOnOrderCancelReject":"true",
  "OrderManager.SecDefRequestTags":"48",
  "OrderManager.GenerateNewStyleOrderIds":"true",
  "OrderManager.KillOrdersOnSessionDown":"false"
  })
```

In the case of different market names for MD and OMS the SessionConfiguration for MD must be sent before the SessionConfiguration of OMS. This is done so that the SecurityListManager of

MD replies to GetFIXSecurityID requests before OM session. Since "SecurityList.CreateSecurityList": "true" is being used in MD session parameter, the Security info is stored by it, so it has to reply to all the queries.

Sample SymbolSet event

```
com.apama.ata.SymbolSet(
  "BMF", "FIX", "BmfFixTransportMD", "",
  {"22": "8", "207": "XBMF"},
  "BMF-FIX", "BmfFixChannel", "BmfFixTransportOMS", "",
  {"22": "8", "207": "XBMF", "Alloc0": "123456,99", "Party0": "XXX,D,36",
  "Party1": "999,D,7", "Party2": "YYY,D,54"},
  ["DOLF11", "INDF11"], [], {})
```

Marketdata subscription or unsubscriptions

Sample events for subscription and unsubscription to market data.

subscriptions

Without book aggregation: `com.apama.marketdata.SubscribeDepth("FIX", "BMF_TRANSPORT", "DI1V12", {"207": "XBMF", "22": "8", "AggregatedBook": "N"})`

With book aggregation: `com.apama.marketdata.SubscribeDepth("FIX", "BMF_TRANSPORT", "DI1V12", {"207": "XBMF", "22": "8", "AggregatedBook": "Y"})`

Unsubscriptions

without book aggregation: `com.apama.marketdata.UnsubscribeDepth("FIX", "BMF_TRANSPORT", "DI1V12", {"207": "XBMF", "22": "8", "AggregatedBook": "N"})`

with book aggregation: `com.apama.marketdata.UnsubscribeDepth("FIX", "BMF_TRANSPORT", "DI1V12", {"207": "XBMF", "22": "8", "AggregatedBook": "Y"})`

Placing order

BMF orders are preprocessed in support mons and then passed onto primary monitors for further processing. Support mons listens for the orders on serviceId "BMF-FIX". So send the all the oms orders to using this serviceID.

ServiceId to be used for order placing : "BMF-FIX". Appropriate Parties info and account should be filled in the extra parameters.

Examples

LIMIT orders with validity DAY

```
com.apama.oms.NewOrder("BMF:1000", "WDLF15", 20.0, "BUY", "LIMIT", 100,
  "BMF-FIX", "", "", "BMF_TRANSPORT", "", "", {"Party0": "8,D,7",
  "Party2": "XPROG01,D,36", "1": "1002"})
com.apama.oms.AmendOrder("BMF:1000", "BMF-FIX", "WDLF15", 21.0, "BUY", "LIMIT",
  300, {"453": "[com.apama.fix.ExecutionReport_Party(\"8\", \"D\", \"7\", [], {}),
  com.apama.fix.ExecutionReport_Party(\"LIN\", \"D\", \"54\", [], {}),
```

```

com.apama.fix.ExecutionReport_Party
  (\XPROG01\,\D\,\36\,[],{ })]" , "1": "1002"}
com.apama.oms.CancelOrder("BMF:1000", "BMF-FIX",
  {"453": "[com.apama.fix.ExecutionReport_Party(\8\,\D\,\7\,[],{ }),
  com.apama.fix.ExecutionReport_Party(\LIN\,\D\,\54\,[],{ }),
  com.apama.fix.ExecutionReport_Party
  (\XPROG01\,\D\,\36\,[],{ })]" , "1": "1002"}

```

#LIMIT orders with validity GTC

```

com.apama.oms.NewOrder("BMF:2010", "WDLF15", 20.0, "BUY", "LIMIT", 200,
  "BMF-FIX", "", "", "BMF_TRANSPORT", "", "", {"Party0": "8,D,7",
  "Party1": "LIN,D,54", "Party2": "XPROG01,D,36", "1": "1002", "59": "1"})
com.apama.oms.AmendOrder("BMF:2010", "BMF-FIX", "WDLF15", 21.0, "BUY", "LIMIT", 300,
  {"453": "[com.apama.fix.ExecutionReport_Party(\8\,\D\,\7\,[],{ }),
  com.apama.fix.ExecutionReport_Party(\LIN\,\D\,\54\,[],{ }),
  com.apama.fix.ExecutionReport_Party
  (\XPROG01\,\D\,\36\,[],{ })]" , "1": "1002", "59": "1"})
com.apama.oms.CancelOrder("BMF:2010", "BMF-FIX",
  {"453": "[com.apama.fix.ExecutionReport_Party(\8\,\D\,\7\,[],{ }),
  com.apama.fix.ExecutionReport_Party(\LIN\,\D\,\54\,[],{ }),
  com.apama.fix.ExecutionReport_Party
  (\XPROG01\,\D\,\36\,[],{ })]" , "1": "1002"}

```

#MARKET TO LIMIT orders with validity IOC

```

com.apama.oms.NewOrder("BMF:13000", "WDLF15", 0.0, "SELL",
  "MARKET WITH LEFTOVER AS LIMIT", 100, "BMF-FIX", "", "", "BMF_TRANSPORT",
  "", "", {"Party0": "8,D,7", "Party1": "LIN,D,54",
  "Party2": "XPROG01,D,36", "1": "1002", "59": "3"})

```

#MARKET WITH PROTECTION orders with validity GTD

```

com.apama.oms.NewOrder("BMF:24000", "WDLF15", 0.0, "SELL", "MARKET", 100,
  "BMF-FIX", "", "", "BMF_TRANSPORT", "", "", {"Party0": "8,D,7",
  "Party1": "LIN,D,54", "Party2": "XPROG01,D,36", "1": "1002",
  "59": "6", "432": "20111118"})

```

#MARKET ON AUCTION orders

```

com.apama.oms.NewOrder("BMF:26000", "WDLF15", 0.0, "SELL", "MARKET", 100,
  "BMF-FIX", "", "", "BMF_TRANSPORT", "", "", {"Party0": "8,D,7",
  "Party1": "LIN,D,54", "Party2": "XPROG01,D,36", "1": "1002", "59": "A"})

```

#STOP LIMIT orders with validity GTC

```

com.apama.oms.NewOrder("BMF:31000", "WDLF15", 23.1, "BUY", "STOP LIMIT", 100,
  "BMF-FIX", "", "", "BMF_TRANSPORT", "", "", {"Party0": "8,D,7",
  "Party1": "LIN,D,54", "Party2": "XPROG01,D,36", "1": "1002", "59": "1", "99": "23.0"})

```

#STOP with PROTECTION orders with validity GTD

```

com.apama.oms.NewOrder("BMF:42000", "WDLF15", 0.0, "BUY", "STOP MARKET", 100,
  "BMF-FIX", "", "", "BMF_TRANSPORT", "", "", {"Party0": "8,D,7",
  "Party1": "LIN,D,54", "Party2": "XPROG01,D,36", "1": "1002", "59": "6",
  "432": "20111118", "99": "31.4"})

```

#Minimum Quantity Orders

```
com.apama.oms.NewOrder("BMF:60000", "WDLF15", 20.0, "BUY", "LIMIT", 200,
    "BMF-FIX", "", "", "BMF_TRANSPORT", "", "", {"Party0": "8,D,7",
    "Party1": "LIN,D,54", "Party2": "XPROG01,D,36", "1": "1002", "110": "100"})
```

#Disclosed Quantity Orders

```
com.apama.oms.NewOrder("BMF:70000", "WDLF15", 20.0, "BUY", "LIMIT", 200,
    "BMF-FIX", "", "", "BMF_TRANSPORT", "", "", {"Party0": "8,D,7",
    "Party1": "LIN,D,54", "Party2": "XPROG01,D,36", "1": "1002", "111": "100"})
```

#MARKET ON CLOSE orders

```
com.apama.oms.NewOrder("BMF:90000", "WDLF15", 0.0, "SELL", "MARKET", 100,
    "BMF-FIX", "", "", "BMF_TRANSPORT", "", "", {"Party0": "8,D,7",
    "Party1": "LIN,D,54", "Party2": "XPROG01,D,36", "1": "1002", "59": "7"})
```

Sample events

NewOrders

```
com.apama.oms.NewOrder("BMF:22", "DI1V12", 12.5, "SELL",
    "LIMIT", 100, "BMF-FIX", "", "", "BMF_TRANSPORT", "",
    "", {"207": "XBMF", "Party0": "PROGRESS03,D,36",
    "Party1": "BMF,D,54", "Party2": "58,D,7"})
```

AmendOrder

```
com.apama.oms.AmendOrder("BMF:22", "BMF-FIX", "DI1V12", 12.5,
    "SELL", "LIMIT", 200, {"207": "XBMF", "453": "[
    com.apama.fix.ExecutionReport_Party(\"PROGRESS03\", \"D\", \"36\", [], {}),
    com.apama.fix.ExecutionReport_Party(\"BMF\", \"D\", \"54\", [], {}),
    com.apama.fix.ExecutionReport_Party(\"58\", \"D\", \"7\", [], {})]"})
```

CancelOrder

```
com.apama.oms.CancelOrder("BMF:22", "BMF-FIX", {"207": "XBMF", "453": "[
    com.apama.fix.ExecutionReport_Party(\"PROGRESS03\", \"D\", \"36\", [], {}),
    com.apama.fix.ExecutionReport_Party(\"BMF\", \"D\", \"54\", [], {}),
    com.apama.fix.ExecutionReport_Party(\"58\", \"D\", \"7\", [], {})]"})
```

New EntryPoint system

For connecting to EntryPoint Server, the following configuration has to be followed:

Session configuration

EntryPoint has only order sessions. For example,

```
com.apama.fix.bmf.SessionConfiguration("BMF_TRANSPORT", "BMF_TRANSPORT", {})
```

```
com.apama.fix.SessionConfiguration("BMF_TRANSPORT", {"FixChannel": "BMF_FIX",
    "OrderManager.IgnoreStatusOnOrderCancelReject": "true",
    "OrderManager.GenerateNewStyleOrderIds": "true"})
```

9 BNPP FX FIX Adapter

■ Prerequisites	158
■ IAF configuration	158
■ Service monitor extensions	158
■ Service monitor injection order	158
■ Service ID configuration	159
■ BNPP subscriptions and unsubscriptions (Quote)	159
■ BNPP order management	160

The BNPP group FX eCommerce FIX system deviates from standard FIX 4.4 in a number of ways which have made it necessary to provide a number of extensions to the FIX adapter. The extensions are into two areas: IAF configuration and service monitors.

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

IAF configuration

When acting as a client to BNPP FX, the IAF should be configured with the BNPP specific configuration file, the distribution version of which can be found here:

```
${APAMA_HOME}/adapters/config/FIX-BNPP.xml.dist
```

Note that the requirement for a BNPP-specific configuration file means that an IAF using this configuration file must be used for BNPP FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for BNPP operation. If you need to connect to other FIX servers as well as BNPP, the non-BNPP sessions should be configured in a separate IAF instance. However, BNPP and non-BNPP session can co-exist safely within the same correlator instance.

As well as the standard host, port, sender and target CompID parameters, the following parameters must also be supplied for the session:

```
<property name="StaticField.body.A.554" value="@BNPP_PASSWORD@" />  
<property name="StaticField.body.D.1" value="@BNPP_FIX_ACCOUNT@" />
```

Service monitor extensions

The file `FIX_BNPP_Support.mon` provides mass quote support and execution report acknowledgements to BNPP.

Therefore, the following EPL files must be injected into the correlator, after the standard FIX adapter service monitors:

- `FIX_BNPP_Events.mon`
- `FIX_BNPP_Support.mon`

Service monitor injection order

Inject the monitors listed in [“FIX order management session” on page 82](#) and [“FIX Legacy Market data session” on page 83](#), followed by:

- `${APAMA_HOME}/adapters/monitors/FIX_BNPP_Events.mon`

- `#{APAMA_HOME}/adapters/monitors/FIX_BNPP_Support.mon`

Service ID configuration

The Service Id to be used is "BNPP-FIX".

Session configuration

BNPP supports streaming prices and trading over the same session. The session must be configured as the following examples:

The session facilitates inbound "Mass Quote" messages and outbound "New Order Single" messages (which are requests to trade on the BNPP) as well as Execution Reports (conformation from both client and BNPP).

For example: for most BNPP client installations the following session configuration could be used:

```
com.apama.fix.SessionConfiguration("BNPP_MARKETDATA", {"FixChannel":"XXXX",
  "SERVICEID":"BNPP-FIX","SubscriptionManager.RequireSessionStatusOpen":"true"})
com.apama.fix.SessionConfiguration("BNPP_TRADING", {"FixChannel":"XXXX",
  "SERVICEID":"BNPP-FIX","OrderManager.RequireSessionStatusOpen":"true",
  "OrderManager.GenerateNewStyleOrderIds":"true"})
```

Where "XXXX" is the custom value for fix channel provided by the user.

SessionConfig parameters

SERVICEID: Service ID to be used is "BNPP-FIX". This parameters configures the BNPP specific monitors to handle BNPP specific events or special handling of the events.

SubscriptionManager.RequireSessionStatusOpen : To make sure the session status is open.

OrderManager.GenerateNewStyleOrderIds : This parameter configures the BNPP specific monitors to generate unique OrderIDs during placing orders.

SubscriptionManager.SuppressZeroQuantities : This parameter configures the BNPP specific monitors to support indicative prices.

BNPP subscriptions and unsubscriptions (Quote)

Sample subscription

```
com.apama.marketdata.SubscribeDepth("BNPP-FIX", "BNPP_MARKETDATA",
  "USD/JPY", {"Quote":"Y", "TradingSessionSubId":"ORANGE"})
com.apama.marketdata.SubscribeDepth("BNPP-FIX", "BNPP_MARKETDATA",
  "AUD/NZD", {"Quote":"Y", "TradingSessionSubId":"BLUE"})
```

BNPP has an optional `TradingSessionSubId`. If there is an agreement with the client, an additional extra parameter (e.g. `"TradingSessionSubId":"BLUE"`) should be provided in the subscription request. Pass this if the same is set at BNPP exchange. This parameter allows us to have different massquote streams, on which you can place order for which you are interested.

ORANGE and BLUE feeds only vary in the liquidity provided, ORANGE will give deeper liquidity when compared to BLUE.

Sample unsubscription

```
com.apama.marketdata.UnsubscribeDepth("BNPP-FIX", "BNPP_MARKETDATA",
    "USD/JPY", {"Quote": "Y", "TradingSessionSubId": "BLUE"})
```

Note:

BNPP may periodically supply indicative prices or report unavailable prices.

If indicative quotes are received (i.e tags 134 and 135 are 0) then user will notice empty depths if the sessionconfig parameter `suppressZeroQuantities` is set to true. For example:

```
com.apama.marketdata.Depth("AUD/NZD", [], [], [], [], [{"35": "i",
    "52": "20110907-10:50:10.062", "Market": "BNPP_MARKETDATA",
    "Quote": "Y", "QuoteID": "110907AUD/NZD6934", "QuoteReqID": "1",
    "QuoteSetID": "110907AUD/NZD6934", "SERVICE_NAME": "BNPP-FIX",
    "TotNoQuoteEntries": "2"}])
```

When an unavailable price is reported by BNPP the adapter sends a Depth event with a single depth level with all price and size fields set to zero and a quote id of 'X' as provided by BNPP.

```
com.apama.marketdata.Depth("USD/JPY", [0], [0], [0], [0], [{"35": "i",
    "52": "20110128-09:11:16.298", "ASK1_QuoteEntryID": "X",
    "BID1_QuoteEntryID": "X", "LEVEL1_QuoteEntryID": "X",
    "Market": "BNPP_MARKETDATA", "Quote": "Y", "QuoteID": "110128EUR/SEK4984",
    "QuoteReqID": "9", "SERVICE_NAME": "BNPP-FIX", "TotNoQuoteEntries": "1"}])
```

BNPP order management

Depth event now has `BID<n>_QuoteEntryID` and `ASK<n>_QuoteEntryID` published in the extra parameters. Use `BID<n>_QuoteEntryID` and `ASK<n>_QuoteEntryID` to retrieve the `QuoteID` for the corresponding side and level. Note that, publishing `LEVEL<n>_QuoteEntryID` in extra parameters of depth event has been removed.

Supply following parameter values

QuoteID (Tag 117)	Supply the QuoteEntryID (from the BID<n>_QuoteEntryID or ASK<n>_QuoteEntryID present in the extra params of the depth event)
Currency (Tag 15)	Supply the currency used for price
FutSettDate (Tag 64)	Supply the settlement date provided in the quote for required level

Sample new order

```
com.apama.oms.NewOrder("6", "USD/SGD", 1.2093, "SELL", "PREVIOUSLY QUOTED",
    700000, "BNPP-FIX", "", "", "BNPP_TRADING", "", "",
    {"1": "XXXX", "117": "6126_88988.1", "15": "USD", "64": "20110909"})
com.apama.oms.NewOrder("8", "USD/SGD", 1.2097, "BUY", "PREVIOUSLY QUOTED",
    1000000, "BNPP-FIX", "", "", "BNPP_TRADING", "", "",
    {"1": "XXXX", "117": "6126_89177.1", "15": "USD", "64": "20110909"})
```

where "XXXX" in above NewOrder refers to account provided by BNPP.

BNPP Execution Reports

Once an execution report is received by the adapter an acknowledgement execution report (confirmation) is automatically sent back to BNPP. The outbound message is the same as the inbound execution report but with the ExecType field (tag 150) set to "I".

It is BNPP's requirement that if the client does not receive a fill or reject ExecutionReport within 5 seconds of submitting an order that they are contacted by email at ecsg@bnpparibas.com. This can be implemented at the application level.

In FIX_BNPP_Support.mon, the AutomaticAlert event is routed in case of not getting fill or reject message(executionreport) with in 5 seconds. Application can listen on AutomaticAlert for responding back with proper action to BNPP. As per specification definition of the AutomaticAlert is as follows

```
event AutomaticAlert{
  string TRANSPORT;
  string orderId;
  string Instrument; //refers to the symbol in new order
  string Currency;
  string ClOrdID;
  string Account;
  string SettleDate;
  integer Quantity;
  dictionary<string, string> __extraParams;
}
```

The above event has been defined in package `com.apama.fix.bnpp`.

10 Citigroup FIX Adapter

■ Citigroup SpotESP FIX Adapter	164
■ Citigroup Colo SpotESP FIX Adapter	170

Citigroup SpotESP FIX Adapter

The Citi group FX eCommerce FIX system deviates from standard FIX 4.4 in a number of ways which have made it necessary to provide a number of extensions to the FIX adapter. The extensions are into two areas: IAF configuration and service monitors.

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

Legacy Mode

IAF Legacy configuration

When acting as a client to Citi FX in Legacy mode, the IAF should be configured with the Citi specific configuration file, the distribution version of which can be found here:

```
${APAMA_HOME}/adapters/config/FIX-CITI.xml.dist
```

Note that the requirement for a Citi-specific configuration file means that an IAF using this configuration file must be used for Citi FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for Citi operation. If you need to connect to other FIX servers as well as Citi, the non-Citi sessions should be configured in a separate IAF instance. However, Citi and non-Citi session can co-exist safely within the same correlator instance.

As well as the standard host, port, sender and target CompID parameters, the following parameters must also be supplied for Quote session:

```
<property name="StaticField.body.A.554" value="@PASSWORD@"/>
<property name="StaticField.header.*.57" value="@FIX_TARGET_SUBID@"/>
```

The following parameters must be supplied for Trade session:

```
<property name="StaticField.body.A.554" value="@PASSWORD@"/>
<property name="StaticField.header.*.57" value="@FIX_TARGET_SUBID2@"/>
<property name="StaticField.body.D.1" value="@FIX_ACCOUNT@"/>
<property name="StaticField.body.AJ.1" value="@FIX_ACCOUNT@"/>
<property name="StaticField.body.D.21" value="1"/>
```

Service monitor extensions

The file "FIX_CITI_Support.mon" provides email ping reply support.

Therefore, the following EPL files must be injected into the correlator, after the standard FIX adapter service monitors:

- `FIX_CITI_Events.mon`
- `FIX_CITI_Support.mon`

Service monitor injection order

Inject the monitors listed in “[FIX order management session](#)” on page 82 and “[FIX Legacy Market data session](#)” on page 83, followed by:

- `${APAMA_HOME}/adapters/monitors/FIX_CITI_Events.mon`
- `${APAMA_HOME}/adapters/monitors/FIX_CITI_Support.mon`

Service ID configuration

The Service Id to be used is "CITI-FIX".

Session configuration

Citi sessions must be configured with the following configuration parameters:

Quote Stream Sessions

`FixChannel = FIX` : The Quote session will be dedicated to inbound requests for Executable Streaming Prices and the actual outbound ESPs in response to these requests.

Trading Sessions

`FixChannel = FIX`: The Trade session facilitates inbound "Quote Response" and "New Order" messages (which are requests to trade on the ESPs), and outbound Execution Reports (confirmations).

For example: For most Citi client installations the following session configuration could be used:

- For Citi Quote streaming Session:

```
com.apama.fix.SessionConfiguration("CITI-QUOTE", {"FixChannel":"FIX",
"SERVICEID":"CITI-FIX",
"SubscriptionManager.IgnoreQuoteCancelOnUnsubscription":"true",
"SubscriptionManager.SuppressZeroQuantities":"true",
"SubscriptionManager.AcceptNonPositivePrices":"true"})
```

- For Citi Trading Session:

```
com.apama.fix.SessionConfiguration("CITI-TRADE", {"FixChannel":"FIX",
"SERVICEID":"CITI-FIX"})
```

To send the trading session request as part of Login, use the following configuration parameters:

```
com.apama.fix.SessionConfiguration("CITI-TRADE", {"FixChannel":"FIX",
"SERVICEID":"CITI-FIX","SubscriptionManager.MarketDataFullRefresh":"true",
"SendSessionStatusRequest":"true"})
```

Session configuration parameters

Parameter	Description
SERVICEID	Service ID to be used is "CITI-FIX". This parameters configures the CITI specific monitors to handle CITI specific events or special handling of the events.
SubscriptionManager. IgnoreQuoteCancelOnUnsubscription	This parameter will handle the QuoteCancel messages received as the response to the Quote Unsubscription.
SubscriptionManager. MarketDataFullRefresh	Whether to request full refreshes (snapshots) rather than snapshot + updates. This parameter ensures that in the TradingSessionStatusRequest the tag 263 is set to "0".
SendSessionStatusRequest	Whether to send the TradingSessionStatusRequest on Logon.
SubscriptionManager. SuppressZeroQuantities	Enable to suppress the zero quantities in the com.apama.marketdata.Depth event. The Depth event will not publish those entries which has zeros quantities.
SubscriptionManager. AcceptNonPositivePrice	Overwrites old prices, if the new price is Zero.

Citi subscriptions or unsubscriptions (quote)

Sample subscription

```
com.apama.marketdata.SubscribeDepth("CITI-FIX", "CITI-QUOTE",
  "USDJPY", {"Quote": "Y", "40": "D", "38": "2", "537": "2", "303": "2",
  "Currency": "USD"})
```

Sample unsubscription

```
com.apama.marketdata.UnsubscribeDepth("CITI-FIX", "CITI-QUOTE",
  "USDJPY", {"Quote": "Y", "40": "D", "38": "2", "537": "2", "303": "2",
  "Currency": "USD"})
```

Citi QuoteCancel

When CITI FIX adapter receives a QuoteCancel message, based on the value of QuoteId it decides whether to resubscribe or not. If QuoteId is set to "UNSUBSCRIBE", the adapter will resubscribe automatically.

Notes

When a subscription is sent and quotes stream starts with indicative quotes (i.e tag 537 is 0) then user will notice empty depths. For example:

```
com.apama.marketdata.Depth("USDJPY", [], [], [], [], [{"15": "USD", "167": "FOR",
"35": "S", "52": "20091027-09:33:13", "537": "0", "60": "20091027-09:33:13.394",
"64": "20091029", "762": "SPOT", "Currency": "USD", "Market": "CITI-QUOTE",
"OrdType": "D", "OrderQty": "2", "Quote": "Y",
"QuoteID": "1##643USDJPY0S--27ktwyqzm.7ob9hR", "QuoteReqID": "1",
"QuoteRequestType": "2", "QuoteType": "2", "SERVICE_NAME": "CITI-FIX"}])
```

And in other scenario's quantity will be shown zero. For example: (i.e starts with tradable quotes and then turns to indicative and so on ...)

```
com.apama.marketdata.Depth("USDJPY", [.6128], [.6129], [.61285], [0], [0], [{"15": "USD",
"167": "FOR", "35": "S", "52": "20091027-09:33:13", "537": "0", "60": "20091027-09:33:13.394",
"64": "20091029", "762": "SPOT", "Currency": "USD", "Market": "CITI-QUOTE", "OrdType": "D",
"OrderQty": "2", "Quote": "Y", "QuoteID": "1##643USDJPY0S--27ktwyqzm.7ob9hR",
"QuoteReqID": "1", "QuoteRequestType": "2", "QuoteType": "2", "SERVICE_NAME": "CITI-FIX"}])
```

Multi tier price for a particular currency pair, using the Mass Quote message

To subscribe multi-tier price quotes, OrderQty (tag 38) must be -1. For example:

```
com.apama.marketdata.SubscribeDepth("CITI-FIX", "CITI-QUOTE",
"USDJPY", {"Quote": "Y", "40": "D", "38": "-1", "537": "2", "303": "2", "15": "USD"})
```

For this request, Citi send "Mass Quote" messages (35=i) which are converted to com.apama.market.Depth events. For example:

```
com.apama.marketdata.Depth("USDJPY", [92.18, 92.18, 92.18], [92.22, 92.22, 92.22],
[92.2, 92.2, 92.2], [1000000, 5000000, 10000000], [1000000, 5000000, 10000000],
{"35": "i", "52": "20100112-04:51:12", "537": "2", "LEVEL1_BidSpotRate": "92.18",
"LEVEL1_Currency": "USD", "LEVEL1_FutSettDate": "20100114", "LEVEL1_OfferSpotRate": "92.22",
"LEVEL1_QuoteEntryID": "1##643USDJPY-1S--1Lm7va7G8.1t0pvd--0",
"LEVEL1_TransactTime": "20100112-04:51:12.015", "LEVEL2_BidSpotRate": "92.18",
"LEVEL2_Currency": "USD", "LEVEL2_FutSettDate": "20100114",
"LEVEL2_OfferSpotRate": "92.22",
"LEVEL2_QuoteEntryID": "1##643USDJPY-1S--1Lm7va7G8.1t0pvd--1",
"LEVEL2_TransactTime": "20100112-04:51:12.015", "LEVEL3_BidSpotRate": "92.18",
"LEVEL3_Currency": "USD", "LEVEL3_FutSettDate": "20100114",
"LEVEL3_OfferSpotRate": "92.22",
"LEVEL3_QuoteEntryID": "1##643USDJPY-1S--1Lm7va7G8.1t0pvd--2",
"LEVEL3_TransactTime": "20100112-04:51:12.015", "Market": "CITI-QUOTE",
"Quote": "Y", "QuoteID": "1##643USDJPY-1S--1Lm7va7G8.1t0pvd",
"QuoteReqID": "1", "QuoteSetID": "1##643USDJPY-1S--1Lm7va7G8.1t0pvd--1Lm7va7G8.1t0pvd",
"SERVICE_NAME": "CITI-FIX", "TotNoQuoteEntries": "3"})
```

Parameters corresponding to each tier are visible in the depth extraparams as LEVEL1_BidSpotRate, LEVEL2_BidSpotRate and so on.

Citi order management

Citi supports quote deal with NewOrderSingle message.

Sending orders using OMS interface NewOrder(NewOrderSingle)

Specify the following parameter values:

- QuoteID (Tag 117): Supply the QuoteID
- Supply the QuoteID Currency (Tag 15): Supply the currency used for price.

For more information, see [“Placing an order” on page 38](#).

Sample new order:

```
com.apama.oms.NewOrder("CitiFx:141", "USDGBP",0.0, "BUY",  
"PREVIOUSLY QUOTED", 1000000, "CITI-FIX", "", "",  
"CITI-TRADE", "", "", {"15":"USD",  
"117": "1##643USDGBP1S--nsbhjhQy.8QxMob"})
```

Notes

- For Placing new orders QuoteID(tag 117) is must and received in com.apama.marketdata.Depth() event.
- Tag 693 to take the value of QuoteReqID(tag 131). QuoteReqID is recieved in com.apama.marketdata.Depth() event.
- The configuration parameter ValidateDepthSubErrorOnQuoteCancel has been deprecated. The DepthSubscriptionError will have 'fault' set to false by default.
- For Placing new orders QuoteID(tag 117) is must and received in Extra Param events.
- Tag 693 to take the value of QuoteReqID(tag 131). QuoteReqID is recieved in Extra Param events.

MDA Mode

IAF configuration

When acting as a client to Citi FX in MDA Mode, the IAF should be configured with the Citi specific configuration file, the distribution version of which can be found here:

```
${APAMA_HOME}/adapters/config/FIX-CITI_MDA.xml.dist
```

Note that the requirement for a Citi-specific configuration file means that an IAF using this configuration file must be used for Citi FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for Citi operation. If you need to connect to other FIX servers as well as Citi, the non-Citi sessions should be configured in a separate

IAF instance. However, Citi and non-Citi session can co-exist safely within the same correlator instance.

As well as the standard host, port, sender and target CompID parameters, the following parameters must also be supplied for Quote session:

```
<property name="StaticField.body.A.554" value="@PASSWORD@" />
  <property name="StaticField.header.*.57" value="@FIX_TARGET_SUBID@" />
```

The following parameters must be supplied for Trade session:

```
<property name="StaticField.body.A.554" value="@PASSWORD@" />
  <property name="StaticField.header.*.57" value="@FIX_TARGET_SUBID2@" />
  <property name="StaticField.body.D.1" value="@FIX_ACCOUNT@" />
  <property name="StaticField.body.AJ.1" value="@FIX_ACCOUNT@" />
  <property name="StaticField.body.D.21" value="1" />
```

Service monitor injection order

Inject the monitors to connect to MDA Session to subscribe for Quotes. See [“FIX MDA session” on page 82](#) and [“FIX order management session” on page 82](#).

Session configuration

CITI uses Legacy FIX for placing orders and MDA for Quotes. Citi Trading sessions must be configured with the following configuration parameters:

```
FixChannel = FIX
```

The Trade session facilitates inbound "Quote Response" and "New Order" messages (which are requests to trade on the ESPs), and outbound Execution Reports (confirmations). For example:

```
com.apama.fix.SessionConfiguration("CITI-TRADE", {"FixChannel":"FIX",
  "SERVICEID":"CITI-FIX"})
```

To send the trading session request as part of Login use the following configuration parameters:

```
com.apama.fix.SessionConfiguration("CITI-TRADE", {"FixChannel":"FIX",
  "SERVICEID":"CITI-FIX","SubscriptionManager.MarketDataFullRefresh":"true",
  "SendSessionStatusRequest":"true"})
```

Subscribing and unsubscribing

The CITI MDA adapter uses the MDA QuoteBook Stream to disseminate the Quote DATA.

- Streams Supported: QuoteBook

Subscription

- string symbol: The symbol to connect to. For example, USDJPY
- controlParams: The following dictionary of control parameters needs to be passed:

Parameter	Description
QuoteRequestType	Must be "2" (Manual)
QuoteType	Must be "2" for ESP ("0" : Indicative, "1" : Tradable)
OrderQty	<ul style="list-style-type: none"> ■ "1" for Stack Mass Quote (if the setup on Citi side is Stack Mass Quote) ■ "1" or "2" etc. for Tier 1 or Tier 2 (if setup on Citi side is tieredSingle) ■ "-1" for Tiered Mass Quote ■ "2300000" or "3200000" etc. for Mass Quotes by Amount

Note:

Other parameters like Currency or SettleDate needs to be passed the same way.

For example:

```
string symbol := "USDJPY"
com.apama.session.CtrlParams controlParams := new com.apama.session.CtrlParams;
controlParams.addParam("QuoteRequestType","2");
controlParams.addParam("QuoteType","2");
controlParams.addParam("OrderQty","-1");
controlParams.addParam("Currency","USD");
```

Processing QuoteResponse

```
action receiveQuotes(CurrentQuotebookInterface cqbiFace) {

    sequence<com.apama.md.client.QuoteEntry> bidEntries := cqbiFace.getRawBids();
    sequence<com.apama.md.client.QuoteEntry> askEntries := cqbiFace.getRawAsks();
    com.apama.md.EP ep:=cqbiFace.getEP();

    //Retrieve QuoteID and Price from Each Entry
    com.apama.md.client.QuoteEntry entry;
    for entry in bidEntries {
        log entry.getQuoteId() at INFO;
        log entry.getPrice().toString() at INFO;
        log entry.getQuantity().toString() at INFO;
        log entry.getEPValuesInterface() at INFO;
    }
}
```

Citigroup Colo SpotESP FIX Adapter

Citi Colo adapter can be configured to run either in Legacy mode or MDA mode. MDA configuration makes use of the Market Data API of the Apama CMF. Order handling is performed using the Legacy mode.

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

IAF configuration

When acting as a client to Citi Colo FX, the IAF adapter should be configured with the Citi Colo specific configuration file, the distribution version of which can be found here:

- Legacy mode

```
${APAMA_HOME}/adapters/config/FIX-CITI-Colo.xml.dist
```

- MDA mode

```
${APAMA_HOME}/adapters/config/FIX-CITI-Colo_MDA.xml.dist
```

For Citi Colo FIX sessions, you must use the IAF adapter that is configured with the Citi Colo specific configuration file. The FIX adapter does not interoperate with other FIX servers once it has been configured for Citi Colo sessions. If you need to connect to other FIX servers as well as Citi Colo sessions, you must configure the sessions other than Citi Colo in a separate IAF instance.

Citi Colo uses a custom data dictionary (FIX44-CITI-Colo.xml). This file must be referenced by the QuickFIX.DataDictionary transport property. For example:

```
<property name="QuickFIX.DataDictionary"
  value="${APAMA_HOME}/adapters/config/FIX44-CITI-Colo.xml"/>
```

The IAF configuration file should contain appropriate host, port, sender and target CompID parameters.

For the MDA Mode the path of the correct adapter configuration file should also be provided using the property AdapterConfiguration.

```
<property name="AdapterConfiguration"
  value="${APAMA_HOME}/adapters/config/FIXTransport-BaseFIXLibrary-Citi-COLO.xml"/>
```

Parameters from Citi

Tag 1408 is verified for CSAT_7.0.16.

PASSWORD is an optional feature and you must confirm with Citi if this is enabled.

```
<property name="StaticField.header.*.50" value="@FIX_SENDER_SUBID2@"/>
<property name="StaticField.body.A.1408" value="CSAT_7.0.16"/>
<property name="StaticField.body.A.554" value="@PASSWORD@"/>
<property name="StaticField.body.D.1" value="@FIX_ACCOUNT@"/>
```

Service monitor injection order

For Legacy session

Inject the monitors listed in [“FIX order management session” on page 82](#) and [“FIX Legacy Market data session” on page 83](#), followed by:

- `${APAMA_FOUNDATION_HOME}/ASB/bundles/New Market Data API.bnd`
- `${APAMA_HOME}/adapters/monitors/FIX_CITI_Events.mon`
- `${APAMA_HOME}/adapters/monitors/FIX_CITI_Support.mon`
- `${APAMA_HOME}/adapters/monitors/FIX_CITI_PasswordManager.mon`

For MDA session

Inject the monitors listed in [“FIX MDA session” on page 82](#) and [“FIX order management session” on page 82](#), followed by:

- `${APAMA_HOME}/adapters/monitors/FIX_CITI_PasswordManager.mon`

Session configuration

Citi Colo sessions must be configured with the following configuration parameters:

Trading sessions

- `FixChannel = FIX`
- `SERVICEID=CITI-FIX`

Citi Colo adapters has been tested for 'Previously Quoted' and 'Market Order' only.

```
com.apama.fix.SessionConfiguration("CITI-COLO-TRADE",  
    {"FixChannel":"FIX","SERVICEID":"CITI-FIX"})
```

For more information, see [“Placing an order” on page 38](#).

Market data sessions for Legacy mode

- `FixChannel = FIX`
- `SERVICEID = CITI-FIX`

Market Data session uses `SubscribeDepth` interface to make subscriptions for Quotes to Citi. This requires `"Quote": "Y"` to be sent in extra parameters. Citi does not allow to send Order Type tag 40 when performing Subscription, these will be rejected.

```
com.apama.fix.SessionConfiguration("CITI-COLO-QUOTE",  
    {"FixChannel":"FIX","SERVICEID":"CITI-FIX"})
```

Order management

Sample new order

```
com.apama.oms.NewOrder("CitiFx:141", "USDGBP",0.0,"BUY",
  "PREVIOUSLY QUOTED", 1000000, "CITI-FIX", "", "",
  "CITI-COLO-TRADE", "", "", {"15":"USD","117":"4DHPK3GQC6A1GJ000A9PT"})
```

Marketdata management

Sample subscription events

```
com.apama.marketdata.SubscribeDepth("CITI-FIX",
  "CITI-COLO-QUOTE", "USDJPY", {"Quote":"Y", "537":"2",
  "303":"2", "Currency":"USD"})
```

Sample unsubscription events

```
com.apama.marketdata.UnsubscribeDepth("CITI-FIX",
  "CITI-COLO-QUOTE", "USDJPY", {"Quote":"Y", "537":"2", "303":"2",
  "Currency":"USD"})
```

Depth update (Quote)

```
com.apama.marketdata.Depth("USDJPY", [.6128], [.6129],
  [.61285], [0], [0], {"15":"USD", "167":"FOR", "35":"S",
  "52":"20091027-09:33:13", "537":"0", "60":"20091027-09:33:13.394",
  "64":"20091029", "762":"SPOT", "Currency":"USD",
  "Market":"CITI-QUOTE", "OrdType":"D", "OrderQty":"2",
  "Quote":"Y", "QuoteID":"1#643USDJPY0S--27ktwyqzm.7ob9hR",
  "QuoteReqID":"1", "QuoteRequestType":"2", "QuoteType":"2",
  "SERVICE_NAME":"CITI-FIX"})
```

Market data session for MDA mode

- FixChannel = FIX
- SERVICEID = CITI-COLO
- QuoteRequestType = Must be "2" (Manual)
- QuoteType = Must be "2" for ESP ("0" : Indicative, "1" : Tradable)

For example, for most Citi Colo client installations the following session configuration could be used:

```
string symbol := "USDJPY"
com.apama.session.CtrlParams controlParams := new com.apama.session.CtrlParams;
controlParams.addParam("QuoteRequestType", "2");
controlParams.addParam("QuoteType", "2");
controlParams.addParam("Currency", "USD");
```

```
quotebSubscriber := (new QuotebookSubscriberFactory).create(sessionHandler);
quotebSubscriber.setParams(controlParams);
quotebSubscriber.subscribeCb(symbol , onQuotebook);
action onQuotebook(com.apama.md.client.CurrentQuotebookInterface cqbIFace) {
    sequence<com.apama.md.client.QuoteEntry> bidEntries := cqbIFace.getRawBids();
    sequence<com.apama.md.client.QuoteEntry> askEntries := cqbIFace.getRawAsks();
    com.apama.md.EP ep:=cqbIFace.getEPValuesInterface().getRaw();
    log bidEntries.toString() at INFO;
    log askEntries.toString() at INFO;
    log ep.toString() at INFO;
}
```

Password management

CITI Colo service provides an option to change the password by FIX messages. To use this feature, you have to use `com.apama.fix.PasswordChangeRequest` event (refer `FIX_Events.mon` for known event definition) and should listen to `com.apama.fix.PasswordChangeResponse` event to receive the acknowledgement.

For example:

```
com.apama.fix.PasswordChangeRequest("CITI-COLO-QUOTE", "1",
    "USER", "OLDPASSWORD", "NEWPASSWORD", {})
```

If the password is changed successfully, you will receive the following response:

```
com.apama.fix.PasswordChangeResponse("CITI-COLO-QUOTE", "1",
    "USER", true, {"text": "Password Changed Successfully"});
```

The `text` field in `PasswordChangeResponse` carries the status of the password status.

11 CME iLink FIX Adapter

■ Prerequisites	176
■ IAF configuration	176
■ Service monitor extensions	177
■ Service monitor injection order	177
■ Session configuration	178
■ SenderSubID (header tag 50) Configuration	179
■ Audit logger configuration	179
■ Drop Copy session support	183

This file describes the design, implementation and usage of the FIX iLink extensions.

The iLink FIX system deviates from standard FIX in a number of ways which have made it necessary to provide a number of extensions to the FIX adapter. The extensions fall into two areas: IAF configuration and service monitor extensions.

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3.0.1

IAF configuration

When acting as a client to iLink, the IAF must be configured with the iLink specific configuration file. It can be found at `${APAMA_HOME}/adapters/config/FIXiLink.xml.dist`.

Note that the requirement for an iLink-specific configuration file means that an IAF using this configuration file must be used for iLink FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for iLink operation. If you need to connect to other FIX servers as well as iLink, the non-iLink sessions should be configured in a separate IAF instance. This configuration file differs from the standard file in that it defines a number of new events. In addition, it provides the following static FIX tags that are necessary for interaction with the iLink server:

```
<!-- Account -->
<property name="StaticField.body.D.1" value="<Account>"/>
<property name="StaticField.body.F.1" value="<Account>"/>
<property name="StaticField.body.G.1" value="<Account>"/>
<!-- HandlInst -->
<property name="StaticField.body.D.21" value="<HandlInst>"/>
<property name="StaticField.body.G.21" value="<HandlInst>"/>
<!-- CustomerOrFirm -->
<property name="StaticField.body.D.204" value="<CustomerOrFirm>"/>
<property name="StaticField.body.G.204" value="<CustomerOrFirm>"/>
<!-- CitiCode -->
<property name="StaticField.body.D.9702" value="<CitiCode>"/>
<property name="StaticField.body.G.9702" value="<CitiCode>"/>
<!-- SenderSubID -->
<property name="StaticField.header.*.50" value="<SenderSubID>"/>
<!-- SenderSubID -->
<property name="StaticField.header.*.50" value="<SenderSubID>"/>
<!-- TargetSubID -->
<property name="StaticField.header.*.57" value="<TargetSubID>"/>
<!-- SenderLocationID -->
<property name="StaticField.header.*.142" value="<SenderLocationID>"/>
<!-- Enabling use of Secure Login -->
<property name="iLinkSecureLogin" value="Y" />
<!-- Tag list to be used to generate HMAC code for secure login -->
<property name="OrderedTagList" value="header:34, header:49,
header:50, header:52, header:57, body:108, header:142,
body:369, header:1603, header:1604, header:1605"/>
```

```

<!-- Access Key ID - used to sign Logon request to iLink or Drop Copy. Mandatory field.
-->
<property name="AccessKeyId" value="@ACCESS_KEY_ID@"/>
<!-- Secret Key - used to create HMAC signature. Mandatory field. -->
<property name="SecretKey" value="@SECRET_KEY@" />
<!-- Resend request limit handling -->
<property name="FatalRejectMessage" value="Range of messages to
  resend is greater than maximum allowed [2500]."/>
<!-- ManualOrderIndicator (Y=manual, N=automated), it can be overridden
  for individual Orders -->
<property name="StaticField.body.D.1028" value="@ILINK_MANUAL_ORDER_INDICATOR@"/>
<property name="StaticField.body.F.1028" value="@ILINK_MANUAL_ORDER_INDICATOR@"/>
<property name="StaticField.body.G.1028" value="@ILINK_MANUAL_ORDER_INDICATOR@"/>

```

These values must be present and will have to be substituted with the correct values for the account to be used.

Following are standard values defined for CME iLink and should not be changed.

```

<!-- ApplicationSystemName -->
<property name="StaticField.header.A.1603" value="CME iLink FIX Adapter"/>
<!-- TradingSystemVersion -->
<property name="StaticField.header.A.1604" value="CME iLink FIX Adapter version"/>
<!-- ApplicationSystemVendor -->
<property name="StaticField.header.A.1605" value="SoftwareAG"/>

```

The events defined in the iLink configuration file fall into two packages and therefore must be placed into two separate event files which must both be injected in the order shown below:

- FIX_Events.mon
- FIX_iLink_Events.mon

Service monitor extensions

The file `FIX_iLink_Support.mon` provides an order preprocessor that must be used when executing orders against the iLink exchange. The pre-processor is used by routing a new order to the service id 'ILINK-FIX' rather than the usual 'FIX'. The preprocessor will then modify the order to make it suitable for the iLink market before forwarding it on to the usual order manager service and listening for responses (executions etc).

In addition, the pre-processor listens for iLink execution reports and cancel rejects which differ from the FIX standard and translates them before forwarding them onto the order manager. For the orders which are getting rejected immediately, the values of `legsSettled` and `legsFinal` are always zero. Therefore, the following EPL files must be injected into the correlator, after the standard FIX adapter service monitors in this order:

- FIX_EventViewer.mon
- FIX_iLink_Support.mon

Service monitor injection order

Inject the monitors listed in [“FIX order management session”](#) on page 82, followed by:

- `${APAMA_HOME}/monitors/data_storage/MemoryStore.mon`
- `${APAMA_HOME}/adapters/monitors/FIX_iLink_Events.mon`
- `${APAMA_HOME}/adapters/monitors/FIX_iLink_Support.mon`

Session configuration

`com.apama.fix.SessionConfiguration` event might have the following boolean flags specified:

Configuration property	Description
<code>iLinkUseLegacyMode</code>	<p>Type: Boolean</p> <p>If this field's value is "false", then a number of changes to the iLink adapter to work around differences between their protocol and standard fix will be enabled. Software AG recommends that this flag is set to false in all new deployments.</p> <p>The default value is true.</p> <p>Note: The default behavior of the <code>iLinkUseLegacyMode</code> will be changed to false in future releases.</p>
<code>ID_RANGE</code>	<p>Type: String</p> <p>Allocate these many order IDs and persists the ID, maintain pooled IDs for later requests. Once the order IDs usage reaches half of the range, the next range of order ID generation request is made and IDs get pooled.</p> <p>The default value is 100.</p>
<code>MAX_ID_NUM</code>	<p>Type: String</p> <p>Maximum order ID generated. IDs get wrapped after this limit is reached.</p> <p>The default value is 9999999.</p>
<code>iLink_OrderManagerSupport.LogLatency</code>	<p>Type: Boolean</p> <p>Adds support for latency framework in <code>iLink_OrderManagerSupport</code> monitor scope.</p> <p>The default value is false.</p>

Sample

```
com.apama.fix.SessionConfiguration("ILINK_TRADING", {
  "ilinkUseLegacyMode":"false",
  "ilinkSwap55and107Tags":"true",
  "OrderManager.OrderIDServiceName":"ILINK-FIX",
  "FixChannel":"ILINK_FIX",
  "MAX_ID_NUM":"999999",
  "ID_RANGE":"1000",
  "OrderManager.KillOrdersOnSessionDown":"false",
  "SenderCompID":"V06004N",
  "SenderSubID":"Apama",
  "TargetSubID":"G",
  "Account":"606060",
  "CustomerOrFirm":"1",
  "CTICode":"4",
  "SenderLocationID":"23001",
  "ManualOrderIndicator":"N",
  "Audit.FileName":"AuditTestLog.csv","Audit.FileTruncate":"true",
  "EnableAuditLog":"true",
  "Audit.Style":"iLink",
  "OrderManager.useMillisecondTransactTimes":"true",
  "OrderManager.waitForMarketOrderId":"true"})
```

SenderSubID (header tag 50) Configuration

The CME requires that all iLink transactions set the FIX SenderSubID header field (tag 50) to a value that uniquely identifies the trader or automated system that initiated the transaction. Typically this will be the login ID of the trader responsible. Applications using the FIX adapter with iLink must therefore set the "ownerId" field of all `com.apama.oms.NewOrder` events sent to the adapter to an appropriate value. This value will be used as the SenderSubID for all outgoing messages related to a given order, and recorded in the audit log. NewOrder events without a valid "ownerId" field will be rejected by the iLink preprocessor service monitor.

Audit logger configuration

An audit logger is provided with the adapter that meets the requirements of the iLink exchange.

IAF configuration

`EnableMessageSendAcknowledgements` : this is a mandatory properties to an audit log.

Example: `<property name="EnableMessageSendAcknowledgements" value="true"/>`

`MessageAckFieldIdentifiers` : this is an optional properties. it forwards the acknowledgements of upstream messages to service monitor with FIXTag as a key . The value to be given as string with elements are `MessageType:FIXTag` and separated by comma.

Example : `<property name="MessageAckFieldIdentifiers" value="D:11,G:11,F:11"/>`

Session configuration

- `EnableAuditLog`. Type boolean.
Adds support for audit logger (false by default).
- `Audit.Style`. Type string.
Use iLink style audit logging and the value must be "iLink".
- `Audit.Filename`. Type string.
Provide the audit file name.
- `Audit.FileTruncate`. Type boolean.
By enabling this, the existing audit log file will be truncated (false by default)

The following session parameters are mandatory to know the values of some audit fields/columns.

- `SenderCompID`. Used to know the value of Session ID field(i.e Identifier of the iLink Order Entry session.) and Executing Firm ID field(i.e Identifier of the Executing Trading Firm submitting the messages to CME Globex) this value should be the same as `@ILINK_FIX_SENDER_COMPID@` properties in the IAF configuration file.
- `SenderSubID`. The value of Operator ID field (i.e Identifier of the operator, who submitted the message or is responsible for its submission) this value should be the same as `@ILINK_SENDER_SUB_ID@` properties in the IAF configuration file.
- `TargetSubID`. The value of Market Segment ID field (57 tag value) this value should be the same as `@ILINK_TARGET_SUB_ID@` properties in the IAF configuration file.
- `Account`. The value of Trading account number field. this value should be the same as `@ILINK_ACCOUNT@` properties in the IAF configuration file.
- `CustomerOrFirm`. The value of Origin field(customer or Firm). The valid values are: 0 or 1 this value should be the same as `@ILINK_CUSTOMER_OR_FIRM@` properties in the IAF configuration file.
- `CTICode`. Customer Type Indicator. The valid value are : 1, 2, 3 or 4 this value should be the same as `@ILINK_CITI_CODE@` properties in the IAF configuration file.
- `SenderLocationID`. Country of Origin(i.e Location of the operator) this value should be the same as `@ILINK_SENDER_LOCATION_ID@` properties in the IAF configuration file.
- `ManualOrderIndicator`. The value of Manual Order Identifier field (i.e Identifies if the captured message was submitted manually or automatically) The valid value are: 'Y' or 'N'. this value should be the same as `@ILINK_MANUAL_ORDER_INDICATOR@` properties in the IAF configuration file.

Note:

Values for the above fields will be the same as those used in populating the IAF configuration file.

Sample SessionConfiguration events for Audit Logger :

```
com.apama.fix.SessionConfiguration("ILINK_TRADING",
  {"ilinkUseLegacyMode":"false","ilinkSwap55and107Tags":"true",
  "OrderManager.OrderIDServiceName":"ILINK-FIX",
  "FixChannel":"ILINK_FIX", "MAX_ID_NUM":"999999","ID_RANGE":"1000",
  "OrderManager.KillOrdersOnSessionDown":"false",
  "OrderManager.waitForMarketOrderId":"true","SenderCompID":"F06004N",
  "SenderSubID":"Apama","TargetSubID":"G","Account":"66600",
  "CustomerOrFirm":"0","CTICode":"3","SenderLocationID":"IN",
  "ManualOrderIndicator":"Y","Audit.FileName":"AuditTestLog.csv",
  "Audit.FileTruncate":"true","EnableAuditLog":"true",
  "Audit.Style":"iLink","OrderManager.useMillisecondTransactTimes":"true"})
```

Note:

Audit messages are not logged in below cases:

- When IAF restarts.
- If you are not receiving acceptance for the order, you are not logging "TO CME"
- If you are not logging "TO CLIENT" for Order cancel Reject/Order cancel replace reject if order is already FILLED/REJECTED.

Support Custom/Additional Headers

The following session parameter parameters can be used to add additional columns to the Audit Log. These columns will get added to the end of the columns.

Configuration parameter	Description
<code>AuditLogger.CustomHeaders</code>	<p>type string</p> <p>Using this parameter the user is able to log new fields/columns in the audit log. The value to be given as string with new column names separated by comma. This field is mandatory if new custom header is to be added. Repetition of the headers will be ignored. The column names are case-sensitive.</p> <p>Example: <code>"AuditLogger.CustomHeaders":"NewColumn1, NewColumn2"</code> where <code>NewColumn1</code> and <code>NewColumn2</code> are the new columns to be added in the audit log</p>
<code>AuditLogger.CustomFieldMapping</code>	<p>type dictionary<string, dictionary<string, string>></p> <p>Using this parameter, the user is able to map a column to a message field for a given message type. All the NEW custom header names mentioned in the parameter should be present in <code>AuditLogger.CustomHeaders</code> too, otherwise the field</p>

Configuration parameter	Description
	<p>will be ignored. But to define the mapping for an existing header, it is not necessary to add the header in <code>AuditLogger.CustomHeaders</code>.</p> <p>Supported message types are:</p> <ul style="list-style-type: none"> ■ NEW ORDER, ■ AMEND ORDER, ■ CANCEL ORDER, ■ ORDER UPDATE, ■ D that is, FIX New Order Single, ■ G that is, FIX Order Cancel Replace Request, ■ F that is, FIX Order Cancel Request, ■ 8 that is, FIX Execution Report, ■ 9 that is, FIX Order Cancel Reject, ■ j that is, Business Message Reject <p>Example:</p> <pre>"AuditLogger.CustomFieldMapping": "{\ "NEW ORDER\ ": {\ "NewColumn1\ ": \ "price\ " } ,\ "8\ ": {\ "NewColumn2\ ": \ "CumQty\ " ,\ "NewColumn1\ ": \ "CLOrdID\ " } }"</pre> <p>that is, <code>NewColumn1</code> will store price for New Order but <code>CLOrdID</code> of FIX Execution report. <code>NewColumn2</code> will store <code>CumQty</code> of FIX Execution report.</p>
<code>AuditLogger.OverrideFieldMappings</code>	<p>type boolean</p> <p>By enabling this parameter, the mapping of a column to a message field can be re-configured. Use <code>AuditLogger.CustomFieldMapping</code> parameter to redefine the mapping of the header to a new field (message field or EP field). This parameter is optional, by default it is false.</p> <p>Example:</p> <pre>"AuditLogger.OverrideFieldMappings": "true"</pre>

Sample `SessionConfiguration` events for Audit Logger to add additional columns to it:

```
com.apama.fix.SessionConfiguration("ILINK_TRADING", {"ilinkUseLegacyMode":
"false", "ilinkSwap55and107Tags": "true",
"OrderManager.OrderIDServiceName": "ILINK-FIX",
"FixChannel": "ILINK_FIX", "MAX_ID_NUM": "999999", "ID_RANGE": "1000",
```

```
"OrderManager.KillOrdersOnSessionDown":"false",
"OrderManager.waitForMarketOrderId":"true","SenderCompID":"F06004N",
"SenderSubID":"Apama","TargetSubID":"G",
"Account":"66600","CustomerOrFirm":"0","CTICode":"3",
"SenderLocationID":"IN","ManualOrderIndicator":"Y",
"Audit.FileName":"AuditTestLog.csv","Audit.FileTruncate":"true",
"EnableAuditLog":"true","Audit.Style":"iLink",
"OrderManager.useMillisecondTransactTimes":"true",
"AuditLogger.CustomHeaders":"NewColumn1,NewColumn2",
"AuditLogger.CustomFieldMapping":{"\NEW ORDER\":{"NewColumn1\":"price\"},
\8\":{"NewColumn2\":"CumQty\"},
\NewColumn1\":"ClOrdID\"}}","AuditLogger.OverrideFieldMappings":"true"})
```

Note that all systems connecting to the iLink exchange are required by the CME to produce an approved audit log, so the audit log must not be disabled, the log files must be retained for a sufficient period as specified by the CME and in case any file access issue make sure that fix those issue before process further.

Note:

It is recommended that you do not use the defaults, as this behaviour breaks the standard Apama abstraction. Use this configuration event:

```
com.apama.fix.SessionConfiguration("ILINK_TRADING",
{"iLinkUseLegacyMode":"false",
"iLinkSwap55and107Tags":"true","OrderManager.OrderIDServiceName":"ILINK-FIX",
"FixChannel":"ILINK_FIX", "MAX_ID_NUM":"999999","ID_RANGE":"1000",
"OrderManager.KillOrdersOnSessionDown":"false",
"OrderManager.waitForMarketOrderId":"true",
"OrderManager.useMillisecondTransactTimes":"true"})
```

The configuration parameters `iLinkUseExecReportAvgPrice` and `iLinkUseExecReportCumQty` should be set to "false" if the user plans to use the adapter with the standard OMS block.

Sample events:

Placing Order:

```
com.apama.oms.NewOrder("1","6RH0",32640.0,"SELL","LIMIT",5,
"ILINK-FIX","","","ILINK_TRADING","","apama_admin",{})
```

Drop Copy session support

The Drop Copy service allows users to receive real-time copies of CME Globex Execution Report and Acknowledgment messages as they are sent over iLink order entry system sessions. Drop Copy aggregates iLink messages, enabling users to aggregate positions and monitor orders for sessions guaranteed by one or more clearing firms upon approval from these clearing firms.

Service monitor extensions

The file `FIX_iLink_DropCopySupport.mon` handles drop copy session of the iLink exchange. The pre-processor reject the new order request and update to user by `OrderUpdate` with the service id 'ILINK-DROP-COPY'.

In addition, the pre-processor routes Order Updates mapping orderID to corresponding correlationID and service name to "ILINK-DROP-COPY", With this correlationID user can identify the order flow of the messages. Therefore, the following EPL files must be injected into the correlator, after the standard FIX adapter service monitors:

- FIX_DropCopyManager.mon
- FIX_iLink_DropCopySupport.mon

Injection order for supporting Drop Copy session

Inject the monitors listed in ["FIX order management session"](#) on page 82, followed by:

- \${APAMA_HOME}/adapters/monitors/FIX_DropCopyManager.mon
- \${APAMA_HOME}/adapters/monitors/FIX_iLink_DropCopySupport.mon

You can configure the ilink adapter to support Drop Copy session through below session configuration. Sample session configuration:

```
com.apama.fix.DropCopyConfiguration("ILINK_DROPCOPY",
{"SERVICEID":"ILINK-DROP-COPY","FixChannel":"ILINK_FIX_DROPCOPY"})
```

IAF configuration

When acting as a client to iLink Drop Copy service, IAF must be configured with the iLink specific configuration file located at \${APAMA_HOME}/adapters/config/FIX-iLink_DropCopy.xml.dist.

Note that the requirement for an iLink-specific configuration file means that an IAF using this configuration file must be used for iLink Drop Copy FIX sessions only.

Note:

Order Update which is routed from drop copy session will contain correlationID as Order ID

Sample OrderUpdate routed from Drop Copy Session :

```
com.apama.oms.OrderUpdate("000000000", "6E", 11245, "SELL", "MARKET",
6, false, false, false, false, false, false, false, "62523108722",
6, 0, 1, 11265, 11265, "Filled:", "1028": "N", "1057": "Y", "107": "6EM6",
"167": "FUT", "337": "TRADE", "35": "8", "375": "CME000A", "432": "20160322",
"442": "1", "48": "166067", "52": "20160322-05:16:16.045",
"527": "6252310872220160322766", "59": "0", "60": "20160322-05:16:16.044",
"75": "20160322", "9717": "000000000", "Account": "606060",
"ExecID": "62257:M:72877TN0000766", "ExecType": "2",
"Market": "ILINK_DROPCOPY", "OrdStatus": "2", "SERVICE_NAME": "ILINK-DROP-COPY",
"__APAMA_ORDER_STATE": "settled"})
```

12 CME Simple Binary Encoding Adapter

■ Prerequisites	186
■ Configuring correlator	187
■ Configuring IAF	187
■ Using SecurityDefinition query interface	192
■ Subscribing and unsubscribing to an event	193
■ Troubleshooting	193

The CME SBE adapter uses the Apama Integration Adapter Framework (IAF) to connect to a CME FIX/SBE market data feed. The CME SBE adapter works with the following functionality of the CME FIX/SBE market data feed:

- Market depth
- Trade reports
- Market statistics
- Instrument static data (security definitions)

The adapter supports the following data stream types:

- BBA
- DEPTH
- TRADE
- EP

For more information on the data stream types and their usage, see MDA documentation.

The adapter works with all the FIX/SBE message types used by the CME feed, although it uses only the following message types directly:

- Market Data Snapshot Full Refresh (35=W)
- Market Data Snapshot Incremental Refresh (35=X)
- Security Definition (35=d)
- Security Status Message (35=f)

The adapter ignores the following message types:

- News (35=B)
- Quote Request (35=R)

All other CME FIX/SBE message types are only used internally by the IAF transport.

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

Configuring correlator

Service monitor extensions

CME_SBE_IAFEvents.mon has event definitions used by the adapter, and CME_SBE_SecDef_QueryInterface.mon has the support for querying the security definitions received from the exchange.

Service monitor injection order

Inject the monitors to connect to MDA Session listed in [“FIX MDA session” on page 82](#), followed by:

- `${APAMA_HOME}/adapters/monitors/CME_SBE_IAFEvents.mon`
- `${APAMA_HOME}/adapters/monitors/CME_SBE_SecDef_QueryInterface.mon`

Setting runtime properties

Once the session is created user can start sending connect requests to the adapter. Following session control parameters are supported by the adapter.

Property name	Description
DISABLE_EP	Type: Boolean If true, disable the publishing of extra parameters
TRANSFER_MODE	Type: String The data transmission mode between adapter and the correlator. Allowable values: SNAPSHOT_ONLY, COMPOUND_DELTA.

Note:

TRANSFER_MODE is an essential property and must be set to one of the above mentioned values by the user for adapter to function properly.

Configuring IAF

An IAF process can support multiple instances of the CME_SBE transport, which can in turn provide data from multiple CME market data channels.

To configure the transport, make a copy of the CME_SBE.xml.dist template configuration file (for example, copy to "CME_SBE_PROD.xml") and edit the copy as described below. All user-configurable parameters are marked with @PARAMETER_NAME@ text in the template configuration. No other part of the transport configuration file should need to be changed.

Transport name and correlator configuration

Replace @CME_SBE_TRANSPORT@ in the <transport> element with the name of this transport instance, for example, "CME_PROD". The same name should also be substituted into the "channels" attribute of the <source> element at the end of the configuration file. Replace @CORRELATOR_HOST@ and @CORRELATOR_PORT@ with the hostname and listening port of your correlator instance.

Message template file configuration

The FIX/SBE decoder uses a template file supplied by CME to describe all of the message structures on the FIX/SBE feed. The most recent template file for each environment (production, certification or new-release certification) can be downloaded from ftp.cmegroup.com. Replace @CME_SBE_TEMPLATES_FILE@ in the transport properties with the full path to the template file.

Channel definitions file configuration

The transport uses a configuration file supplied by the CME to define all of the hosts, ports and multicast groups used by the available market data channels. The most recent channel configuration file for the feed (production, certification or new-release certification) can be downloaded from ftp.cmegroup.com. Replace @CME_SBE_CHANNEL_FILE@ in the transport properties with the full path to this file.

IAF channel name

Replace @CME_SBE_CHANNEL@ with the channel name on which the IAF process will be running. The IAF will listen to the correlator from this channel.

Channels to configure

The List of channels that the transport will connect and configure should be specified. Replace the @CONFIGURE_CHANNELS@ parameter with a comma separated list of channels. For example to configure channels 9 and channel 11, Replace @CONFIGURE_CHANNELS@ with "11, 9".

RefreshNow

The RefreshNow parameter is to tell the adapter whether to request an immediate refresh of security definitions on the configured channels. The default value is set to true.

Additional transport configuration

- **CorrelatorHBTimeout.** A natural number telling the transport how many seconds it should wait for a heartbeat from the service monitors before deciding that the connection had been lost. If not specified in the configuration file it defaults to 15 seconds.
- **MulticastInterface.** IP address (as a dotted quad, for example, "1.2.3.4") of the network interface on the machine running the adapter that should be used to subscribe to UDP multicast traffic. If not specified, all interfaces will be used. For example:

```
<property name="MulticastInterface" value="a.b.c.d"/>
```

You can also configure `MulticastInterface` for different feeds (primary and secondary). For Example:

```
<property name="MulticastInterface.A" value="a.b.c.d"/>
<property name="MulticastInterface.B" value="w.x.y.z"/>
```

- `IncludeExtraParam`. Using this property, adapter can publish requested fields/tag for any market data as extra parameters. These tags must be separated by comma. For example:

```
<property name="IncludeExtraParam" value="60,75"/>
```

Note:

Only top level fields can be published using this property.

- `SecurityDefinitionCacheFolder`. Location Of Security Definition Cache Folder (as a String). The cache file is created inside this folder by the following name `Channel_<channel name>.dat`.

For example, the security definition cache file for channel no 11 should be named as `Channel_11.dat`.

- `DisableSecurityDefinitionCache`. Parameter to tell the adapter whether to load/save the security definition from/to the security definition cache file. If set to false, the transport will try to load the security definition from the `SecurityDefinitionCacheFolder`. The name of the cache file should be `Channel_<channel name>.dat`. If the file is not present cache will not be loaded. If `RefreshNow` configuration parameter was set to true and if the `DisableSecurityDefinitionCache` is set to false, adapter will save the security definition in the cache file in `SecurityDefinitionCacheFolder`. For every configured channel one cache file will be created. By default it is set to false.
- `LogMessages`. A comma-separated list of FIX message types that should be logged when they are received or sent by the adapter. The message contents will be logged at INFO level. The following message types are supported:
 - Heartbeat
 - Logout
 - Logon
 - News
 - SecurityDefenition
 - SecurityStatus
 - QuoteRequest
 - MarketDataRequest
 - MarketDataFullRefresh
 - MarketDataIncrementalRefresh

- `MarketDataRequestReject`
- `Unknown`
- `LogFIXMessages`. A Boolean property to tell the transport whether to save the Fix Messages it receives in a file. By default the file will be saved in the folder named "CME_FIX_LOG". The filename will be "Channel_<channel name>_<feed type>.log". For example, all the snapshots received from channel 11 will be saved in the file `Channel_11_S.log`.
- `FIXLogFolder`. Location of the Fix Log Folder (as a String). If specified the transport will save the fix messages inside this folder.
- `LogDebug`. A comma separated list of extra debug logging categories that should be enabled. The additional logging produced by an enabled category will be logged at the INFO level unless otherwise stated. Currently the following categories are supported:
 - `UpstreamEvents` enable logging of all events sent to the adapter by the correlator ("upstream" events). These will include subscription and unsubscription requests, commands to enable and disable the feed of security definitions, and heartbeat events.
- `DropCrossedBooks`. A Boolean property to tell the adapter whether to prevent all Depth events having best bid price \geq best offer price from being published. By default it's false.
- `PublishDepthSourceKeys`. A Boolean property, if set to "true", the `com.apama.marketdata.Depth` events generated by the adapter will contain `extraParams` keys identifying the "source" (that is, the outright or implied book) of each price level. The default value is false. For each Bid Level I, the `extraParam Bid-i` can take several different values:
 - "A" is the price at this bid level includes actual orders
 - "I" is the price at this bid level includes implied orders
 - "AI" is the price at this bid level includes actual and implied orders
 - "Ask1"- "AskN" are the As for the corresponding "BidN" keys
- `PublishUpdateInfoKeys`. A Boolean property, if set to "true", `com.apama.markertdata.Depth` events will include `extraParams` keys identifying the "source" of each update which are the channel, update type, and message sequence numbers Channel - FIX/SBE channel number.
 - `UpdateType` - "S" (Snapshot) or "I" (Incremental)
 - `SeqNum` - Per-channel message sequence number
 - `RptSeq` - Highest per-instrument update sequence number that contributed to this event
 - `RptSeq0` - Lowest per-instrument update sequence number that contributed to this event
 - `LastMsgSeq` - (Snapshots only) highest incremental message sequence number that contributed to the contents of the snapshot messageIts default Value is false.
- `PublishExtraFIXKeys`. A Boolean property, If set to "true", `com.apama.marketdata.Depth` events will include additional FIX tags from the FIX/SBE messages used to generate the events, in the

extraParams dictionary. The set of extra tags that will be included is subject to change. The default value of this key is "false".

- PublishTradeSummary. A Boolean property to tell the adapter to send trade summary details for corresponding Trade update's. By default it's set to false.
- PublisherPoolSize. This property is used to configure the size of the publisher thread pool. The default size is 2.
- DecoderPoolSize. This property is used to configure the size of the decoder thread pool. The default size is 1.

Note that LogMessages, LogFixMessages and LogDebug can produce extreme amounts of output and should be used with caution in a production environment.

Clients wishing to minimize the size of Depth events by eliminating extraParams that are not required for a given application should keep the PublishDepthSourceKeys, PublishUpdateInfoKeys, and PublishExtraFIXKeys session parameters to "false" as required.

Note:

If RefreshNow is set to false, and DisableSecurityDefinitionCache is set to true (that is, secdef cache loading from the disk is disabled), adapter will automatically set RefreshNow to true.

After setting the parameters, your application.dist file must look similar to the sample below:

```
<property name="ChannelConfigFile" value=" config.xml" />
<property name="TemplatesFile" value=" templates.xml" />
<property name="LogDebug" value="" />
<property name="LogMessages" value="" />
<property name="LogFIXMessages" value="true"/>
<property name="ConfigureChannels" value="4, 9, 11" />
<property name="RefreshNow" value="true"/>

<!-- Channel on which the IAF is running -->
<property name="IAF_CHANNEL" value="CME_CHANNEL"/>
<!-- Correlator Heartbeat Timeout -->
<property name="CorrelatorHBTimeout" value="30"/>
<property name="DropCrossedBooks" value="true"/>
<property name="MulticastInterface" value="0.0.0.0" />
```

The Security definition Cache file

As mentioned above the name of the security definition cache file should be Channel_<channel name>.dat. The data in the cache file is in the following format:

- Each line corresponds to one security definition update
- Each line has the structure: <template-id > <fixMessage>

For example the security definition cache file for channel 11 may look like:

```
140 <SecurityDefinition Message>
140 <SecurtiyDefinition Message>
151 <SecurityDefintion Message>
```

Using SecurityDefinition query interface

To Query the SecurityDefinition Interface, following event has to be sent first.

```
(defined under com.apama.cme package)
event SessionConfiguration {
  string connection;
  dictionary<string,string> configuration;
}
```

The "connection" is the same as for the channel configuration. The channel on which the IAF is running should be specified in the configuration dictionary as the key value pair "channel":<channel name> . If channel name is not specified, the channel name specified in the transport configuration property IAF_CHANNEL is used. The session name should be specified in the configuration dictionary as the key value pair "sessionName":<session-name>. If session name is not specified default session name "CME" is used.

After sending the SessionConfiguration event, following event can be used to Query the SecurityDefinition Interface.

```
(defined under com.apama.cme package)
event SecurityDefinitionRequest {
  string connection;
  integer channel;
  string symbol;
  dictionary<string, string> extraParams;
}
```

The "connection" parameter is the same as for the channel configuration. The "channel" field holds the channel no that has the queried symbol. If the channel no is not known, then "channel" field should be set to -1. The "symbol" field holds the symbol decryption (FIX tag 107) for the security. The "extraParams" field holds any additional data needed for the lookup, although currently no keys are defined for this event.

For example to get the SecurityDefinition for the symbol "6EB3" on channel 11 following events has to be sent.

```
com.apama.cme.SessionConfiguration("CME_SBE_TRANSPORT",
  {"channel":"CME_CHANNEL"})
com.apama.cme.SecurityDefinitionRequest
  ("CME_SBE_TRANSPORT",11,"6BM3",{})
```

Each lookup request will route a com.apama.cme.SecurityDefinitionResponse event in reply:

```
event SecurityDefinitionResponse {
  string connection;
  integer channel;
  string symbol;
  boolean found;
  SecurityDefinition record;
  dictionary<string, string> extraParams;
}
```

The fields of this event will be identical to the request except that the "record" field - holding the returned security definition - is added, and the "found" field is set to "true" if the requested record

was found. If the record was *not* found, "found" will be "false" and the "record" field should be ignored.

Subscribing and unsubscribing to an event

Follow the MDA documentation on how to subscribe and unsubscribe to an event symbol.

Troubleshooting

Make sure that either the IAF configuration property RefreshNow is set to true or the DisableSecurityDefinitionCache is false and the SecurityDefinitionCacheFolder has the security definition cache files for the configured channels. Adapter session will only start only when it has the SecurityDefinitions available.

The template and the channel configuration file provided by the CME should be updated. Client systems should download updates according to the schedule specified.

The FTP site is <ftp://ftp.cmegroup.com/>.

Environment*	Service	Directory Location	Client System Update Schedule
Certification	Template	/Cert/Templates	Sunday prior to market open
Certification	Configuration	/Cert/Configuration	Daily
Certification AutoCert+	Template	/CertAutoCertPlus/Templates	Sunday prior to market open
Certification AutoCert+	Configuration	/CertAutoCertPlus/Configuration	Daily
New Release Certification	Template	/NRCert/Templates	Sunday prior to market open
New Release Certification	Configuration	/NRCert/Configuration	Daily
New Release Certification AutoCert+	Template	/NRAutoCertPlus/Templates	Sunday prior to market open
New Release Certification AutoCert+	Configuration	/NRAutoCertPlus/Configuration	Daily
Production	Template	/Production/Templates	Sunday prior to market open
Production	Configuration	/Production/Configuration	Daily
Production	Flat File	/Production	8:30pm Sunday -Thursday

13 Credit Suisse SER FX FIX Adapter

■ Prerequisites	196
■ IAF configuration	196
■ Service monitor extensions	196
■ Service monitor injection order	196
■ Service configuration	197
■ CS subscriptions and unsubscriptions (Quote)	198
■ CS order management	198

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

IAF configuration

When acting as a client to Credit Suisse FX, the IAF should be configured with the Credit Suisse(CS) specific configuration file, the distribution version of which can be found here:

```
${APAMA_HOME}/adapters/config/FIX-CS.xml.dist
```

Note that the requirement for a CS-specific configuration file means that an IAF using this configuration file must be used for CS FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for CS operation. If you need to connect to other FIX servers as well as CS, the non-CS sessions should be configured in a separate IAF instance. However, CS and non-CS session can co-exist safely within the same correlator instance.

As well as the standard host, port, sender and target CompID parameters, the following parameters must also be supplied for CS Quote and Trade session:

```
<property name="StaticField.body.D.1" value="@FIX_ACCOUNT@" />  
<property name="StaticField.body.D.109" value="@FIX_CLIENT_ID@" />  
<property name="StaticField.body.D.21" value="1" />
```

Note:

Credit Suisse can support multiple sessions by issuing multiple SenderCompID (tag 49) to the Market Taker. This allows, for example, client applications to have separate pricing and trading processes. So the above parameters are applicable for trading session only and need not be going in pricing session.

Service monitor extensions

The file "FIX_CS_Support.mon" provides email ping reply support.

Therefore, the following EPL files must be injected into the correlator, after the standard FIX adapter service monitors:

```
FIX_CS_Support.mon
```

Service monitor injection order

Inject the monitors listed in [“FIX order management session” on page 82](#) and [“FIX Legacy Market data session” on page 83](#), followed by:

```
${APAMA_HOME}/adapters/monitors/FIX_CS_Support.mon
```

Service configuration

CS sessions must be configured with the following configuration parameters:

The Service ID to be used is "CS-FIX".

QUOTE STREAM/TRADING SESSION: - FixChannel = FIX

For example, for most CS client installations, the following session configuration will work:

```
com.apama.fix.SessionConfiguration("CS-QUOTE", {"FixChannel":"FIX",
"SERVICEID":"CS-FIX","SubscriptionManager.delayBeforeResubscription":"1.0",
"SubscriptionManager.IgnoreQuoteCancelOnUnsubscription":"true",
"SendSessionStatusRequest":"true",
"SubscriptionManager.EnableMassQuoteAcknowledgement":"true",
"SubscriptionManager.RequireSessionStatusOpen":"true"})
```

To send unique TradingSession Id, set the configuration parameter

SessionManager.SendUniqTradingSessionId to "true"

```
com.apama.fix.SessionConfiguration("CS-QUOTE", {"FixChannel":"FIX",
"SERVICEID":"CS-FIX","SubscriptionManager.delayBeforeResubscription":"1.0",
"SubscriptionManager.IgnoreQuoteCancelOnUnsubscription":"true",
"SendSessionStatusRequest":"true",
"SubscriptionManager.EnableMassQuoteAcknowledgement":"true",
"SessionManager.SendUniqTradingSessionId":"true",
"SubscriptionManager.RequireSessionStatusOpen":"true"})
```

SessionConfig parameters

SERVICEID	Service ID to be used is "CS-FIX". This parameters configures the CS specific monitors to handle CS specific events or special handling of the events.
SubscriptionManager. delayBeforeResubscription	Delay before sending the re-subscription requests. Takes the float value.
SubscriptionManager. IgnoreQuoteCancelOnUnsubscription	This parameter will handle the QuoteCancel messages received as the response to the Quote Unsubscription.
SubscriptionManager. MarketDataFullRefresh	Whether to request full refreshes (snapshots) rather than snapshot + updates. This parameter ensures that in the TradingSessionStatusRequest the tag 263 is set to "0".
SendSessionStatusRequest	Whether to send the TradingSessionStatusRequest on Logon.
SubscriptionManager. EnableMassQuoteAcknowledgement	Whether to enable the Quote Acks.
SubscriptionManager.	Whether to re-subscribe to Quotes on Logon.

ResubscribeOnLogon

SessionManager. SendUniqTradingSessionId	Whether to send the unique sessionID (Include the time) for Trading session Status request.
---	--

CS subscriptions and unsubscriptions (Quote)

Sample subscription

```
com.apama.marketdata.SubscribeDepth("CS-FIX", "CS-QUOTE",  
  "USD/CAD", {"Quote": "Y", "167": "FOR", "38": "1000", "15": "USD", "OrdType": "C"})
```

Sample unsubscription

```
com.apama.marketdata.UnsubscribeDepth("CS-FIX", "CS-QUOTE",  
  "USD/CAD", {"Quote": "Y", "167": "FOR", "38": "1000", "15": "USD", "OrdType": "C"})
```

CS order management

Sending orders using OMS interface NewOrder(NewOrderSingle)

Supply the following parameters:

SettlDate(Tag 64) Supply the settlement date

QuoteID (Tag 117) Supply the QuoteID

Currency (Tag 15) Supply the currency used for price

Sample new order

```
com.apama.oms.NewOrder("CSFX:1", "USD/CAD", 1.06878, "BUY",  
  "PREVIOUSLY QUOTED", 1000, "CS-FIX", "", "", "CS-QUOTE", "",  
  "", {"117": "83339354", "15": "USD", "64": "20091028"})
```

Note:

All the required tags for NewOrder are received in the event `com.apama.marketdata.Depth()`.

14 Currenex FIX Adapter

■ Prerequisites	200
■ IAF adapter configuration	200
■ Session configuration	200
■ Service monitor extensions	201
■ Service monitor injection order	201
■ Password management	201
■ Marketdata subscriptions	202
■ Order management	202
■ Currenex FIX MDA adapter	203

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

IAF adapter configuration

Currenex uses a custom data dictionary FIX42-CNX.xml. This file must be referenced by the QuickFIX.DataDictionary transport property. For example,

```
<property name="QuickFIX.DataDictionary"
value="${APAMA_HOME}/adapters/config/FIX42-CNX.xml"/>.
```

In addition, the following properties are required for Currenex connections:

```
<!-- Password -->
<property name="StaticField.body.A.554" value="<CURRENEX_PASSWORD>"/>
```

Where CURRENEX_PASSWORD is the password supplied to you by Currenex.

The supplied sample configuration file FIX-CNX-Legacy.xml.dist contains placeholders for the necessary configuration parameters. This configuration file defines two connections (CURRENEX_TRADING and CURRENEX_MARKET_DATA) using the event channel "CURRENEX_FIX" for communication with the correlator.

Session configuration

Currenex sessions must be configured with the following configuration parameters:

```
Trading sessions:
OrderManager.RequireSessionsStatusOpen = true
OrderManager.IgnoreStatusOnOrderCancelReject = true
OrderManager.HandleSuspendedExecType = true
OrderManager.UseCurrencyFromSymbol = "BASE" or "TERM"

OrderManager.CustomOrdTypes = "TRAILING STOP:V,OCO:W,IFD OCO:Y"
This is used to set order types specific to currenex
Market data sessions:
SubscriptionManager.RequireSessionStatusOpen = true
SubscriptionManager.FieldOmissions = Currency
```

Sample session configuration events, compatible with the distributed configuration file:

```
// Currenex trading session
com.apama.fix.SessionConfiguration("CURRENEX_TRADING",
  {"FixChannel":"CURRENEX_FIX", "OrderManager.RequireSessionStatusOpen":"true",
    "OrderManager.IgnoreStatusOnOrderCancelReject":"true",
    "OrderManager.HandleSuspendedExecType":"true",
    "OrderManager.UseCurrencyFromSymbol":"BASE",
    "OrderManager.CustomOrdTypes":"TRAILING STOP:V,OCO:W,IFD OCO:Y"})
// Currenex market data session
com.apama.fix.SessionConfiguration("CURRENEX_MARKET_DATA",
```

```
{"FixChannel":"CURRENEX_FIX",
  "SubscriptionManager.RequireSessionStatusOpen":"true",
  "SubscriptionManager.FieldOmissions":"Currency"}
```

Service monitor extensions

FIX_CNX_Support.mon file has the extensions for supporting password management specifically for connection to Currenex.

Service monitor injection order

Inject the monitors listed in [“FIX order management session” on page 82](#) and [“FIX Legacy Market data session” on page 83](#), followed by:

```
${APAMA_HOME}/adapters/monitors/FIX_CNX_Support.mon
```

Password management

The FIX_CNX_Support.mon service monitor can be used to reset the Currenex session password. This monitor file must be injected after all of the standard service monitors, but before any Currenex sessions are configured.

To reset the password for a Currenex session, send the following event to the correlator:

```
com.apama.fix.cnx.PasswordChangeRequest
(<transport>, <requestId>, <userId>, <oldPassword>, <newPassword>)
```

Where,

- <transport> is the session/transport name (for example, "CURRENEX_TRADING")
- <requestId> is a user-supplied key to identify replies to this request
- <userId> is the SenderCompId for this session
- <oldPassword> is the existing password for the session
- <newPassword> is the desired new password for the session

Each request should result in a com.apama.fix.cnx.PasswordChangeResponse() event being routed and logged by the monitor:

```
com.apama.fix.cnx.PasswordChangeResponse
(<transport>, <requestId>, <userId>, <success>, <status>, <message>)
```

Where,

- <transport> is the session/transport name
- <requestId> is the user-supplied identifier from the request
- <userId> is the SenderCompId from the request
- <success> is a boolean value indicating whether the password was changed

- <status> is one of the RESPONSE_* constants listed in the FIX_CNX_Support monitor. Typical values are RESPONSE_PASSWORD_CHANGED ("5") or RESPONSE_USER_NOT_RECOGNISED ("3") which Currenex uses as a catch-all failure code.
- <message> is any error message returned by the server

A password reset can be performed at any time after the session has been successfully logged on and a Trading Session Status of "Open" has been sent (this will be logged by the FIX Session Manager). A password reset is required if the server sends a Trading Session Status of "Halted" immediately after a successful logon (this will be logged at WARN level by the FIX CNX Password Manager). Note that you cannot successfully log on with an expired password, so perform a password reset as soon as the "Halted" session status is seen. The session will be locked after too many login attempts with an expired password. The session can be unlocked only by calling Currenex technical support.

Marketdata subscriptions

Examples:

```
//market data request (full book non-aggregated)
com.apama.marketdata.SubscribeDepth
("FIX", "CURRENEX_MARKET_DATA", "EUR/USD", {"264": "0", "266": "N", "7560": "Y"})
//market data request (full book aggregated)
com.apama.marketdata.SubscribeDepth
("FIX", "CURRENEX_MARKET_DATA", "EUR/USD", {"264": "0", "266": "Y"})
//market data request (top of book)
com.apama.marketdata.SubscribeDepth
("FIX", "CURRENEX_MARKET_DATA", "EUR/USD", {"264": "1"})
//Tick request
com.apama.marketdata.SubscribeTick
("FIX", "CURRENEX_MARKET_DATA", "EUR/USD", {"263": "T", "264": "1"})
```

Order management

Examples

```
//Limit: Buy Base
com.apama.oms.NewOrder("Currenex:1", "EUR/USD", 1.2899, "BUY", "FOREX LIMIT",
  40000, "FIX", "", "", "CURRENEX_TRADING", "", "",
  {"15": "EUR", "460": "4", "21": "1", "60": "20141002-00:16:40", "59": "1", "1": "xxx"})
//Market: Buy Base
com.apama.oms.NewOrder("Currenex:3", "GBP/USD", 0.0, "BUY", "FOREX MARKET",
  20000, "FIX", "", "", "CURRENEX_TRADING", "", "",
  {"15": "GBP", "460": "4", "21": "1", "60": "20141002-00:16:40", "59": "1", "1": "progress"})
//Stop Loss: Sell Terms - Stop Side (Bid)
com.apama.oms.NewOrder("Currenex:17", "EUR/USD", 0.0, "SELL", "STOP MARKET",
  5000000, "FIX", "", "", "CURRENEX_TRADING", "", "",
  {"15": "USD", "7534": "1", "99": "1.2653", "460": "4", "59": "1"})
//Stop Limit: Buy Base - Stop Side (Bid)
com.apama.oms.NewOrder("Currenex:17", "EUR/USD", 1.2670, "BUY", "STOP LIMIT",
  1000000, "FIX", "", "", "CURRENEX_TRADING", "", "",
  {"15": "EUR", "7534": "1", "99": "1.2650", "460": "4", "59": "1"})
//Trailing Stop Sell - Trail the Bid by 2 pips
com.apama.oms.NewOrder("Currenex:17", "EUR/JPY", 0.0, "SELL", "TRAILING STOP",
  30000, "FIX", "", "", "CURRENEX_TRADING", "", "",
```

```

{"15":"EUR","7534":"1","7587":"0.02","460":"4","59":"1"})
//OCO: Leg 1 is Limit; Leg 2 is Stop
com.apama.oms.NewOrder("Currenex:26","EUR/USD",0.0,"BUY","LIMIT IF TOUCHED",
  3000000,"FIX","","","CURRENEX_TRADING","","",
  {"15":"EUR","7534":"1","59":"0","167":"FOR","460":"4","1":"xxx","7540":"1.0482",
    "7541":"3","7542":"1.0482","7543":"1"})
//IFDOCO order for 2000000 EUR/USD
com.apama.oms.NewOrder("Currenex:32","EUR/USD",0.0,"BUY","BEST LIMIT IF
TOUCHED",2000000,
"FIX","","","CURRENEX_TRADING","","",
{"15":"EUR","59":"0","167":"FOR","460":"4","1":"progress","7535":"3","7536":"2",
  "7537":"1.10382","7539":"1","7569":"3","7570":"1.10382"})
//ICEBERG Order with hidden quantity. Tag 210 (MaxShow)
indicates quantity to be displayed. The Quantity field of NewOrder holds
the total quantity including the hidden amount.
com.apama.oms.NewOrder("Currenex:17","EUR/USD",1.27378,"BUY","FOREX ICEBERG",500000,
"FIX","","","CURRENEX_TRADING","","",
{"15":"EUR","59":"0","210":"200000","460":"4"})

```

Amending an order

```

//Limit order rate is modified
com.apama.oms.NewOrder("Currenex:11","EUR/USD",1.0378,"BUY","FOREX LIMIT",
  500000,"FIX","","","CURRENEX_TRADING","","",
  {"15":"EUR","460":"4","21":"1","60":"20141002-00:16:40","59":"1","1":"xxx"})
com.apama.oms.AmendOrder("Currenex:11","FIX","EUR/USD",1.27374,"BUY","FOREX LIMIT",
  500000,{"15":"EUR","460":"4","21":"1","60":"20141002-00:16:40","59":"1","1":"xxx"})

```

Cancelling an order

```

com.apama.oms.NewOrder("Currenex:12","EUR/USD",1.0378,"BUY","FOREX LIMIT",
  400000,"FIX","","","CURRENEX_TRADING","","",
  {"15":"EUR","460":"4","21":"1","60":"20141002-00:16:40","59":"1","1":"progress"})
com.apama.oms.CancelOrder("Currenex:12","FIX",
  {"15":"EUR","460":"4","21":"1","60":"20141002-00:16:40","59":"1","1":"progress"})

```

Currenex FIX MDA adapter

The Currenex connectivity chain uses new connectivity plug-in architecture to connect to Currenex-FIX compliant systems. For more information on connectivity plug-in, see Apama documentation.

IAF configuration

When acting as a client to Currenex in MDA Mode, the IAF should be configured with the Currenex specific configuration file:

```

${APAMA_HOME}/adapters/config/FIX-CNX.xml.dist

```

Note that the requirement for a Currenex-specific configuration file means that an IAF using this configuration file must be used for Currenex FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for Currenex operation. If you need to connect to other FIX servers as well as Currenex, the non-Currenex sessions should be configured in a separate IAF instance. However, Currenex and non-Currenex sessions can co-exist safely within the same correlator instance.

Connectivity plug-in configuration

The YAML configuration file `FIX-CNX-Connectivity.yaml` describes the chain and its components. It contains place holders for the necessary configuration parameters to connect Currenex ESP Service.

Connection-related configuration are specified in the `fixSession` stanza on the "FixConnectivityTransport" instance. It requires custom data dictionary `FIX44-CNX.xml`. This file must be referenced by the element `DataDictionary` of `fixSession`. You must update the `fixSession` stanza with relevant values.

Service monitor injection order

Inject the monitors to connect to MDA Session listed in ["FIX MDA session" on page 82](#) and ["Connectivity plug-in order management session" on page 84](#).

Subscription management

Currenex FIX Adapter uses the CMF MDA MBP (`com.apama.md.D`) and BBA (`com.apama.md.BBA`) streams to publish marketData, and supports SPOT subscription only.

Note:

Currenex enforces a constraint on the client to send only one subscription request per symbol. To get both BBA and Depth data for same symbol, you can subscribe to Depth stream and use CMF API to get BBA updates.

15 Deutsche Bank Autobahn FIX Adapter

■ Prerequisites	206
■ IAF configuration	206
■ Service monitor injection order	207
■ Platforms supported	207

This file describes the design, implementation and usage of the FIX AUTOBAHN extensions. Autobahn supports two type of FIX APIs. The AutobahnFX RAPID FIX platform offers Spot contracts only and ABFX FIX API which offers Spots, forwards and Swaps. This version of the Autobahn adapter confirms to following version of AUTOBAHN API's:

- Version 2.4 of the AutobahnFX FIX API Integration Guide Rules Document.
- Version 1.14 of AutobahnFX RAPID FIX Integration Guide.

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

IAF configuration

Autobahn Adapter can be configured to run in two modes - Legacy and/or MDA. MDA Configuration makes use of the Market Data API of the Apama CMF and Order handling is done using the Legacy Mode.

When acting as a client to AUTOBAHN(ABFX), the IAF should be configured with the Autobahn specific configuration file, the distribution version of which can be found here:

For Autobahn Rapid platform: `${APAMA_HOME}/adapters/config/FIX-Autobahn-Rapid.xml.dist`

For Autobahn FIX platform: `${APAMA_HOME}/adapters/config/FIX-Autobahn-FX.xml.dist`

Autobahn uses a custom data dictionary . This file must be referenced by the QuickFIX.DataDictionary transport property.

For example, fFor Autobahn Rapid platform:

```
<property name="QuickFIX.DataDictionary"
value="${APAMA_HOME}/adapters/config/FIX44-Autobahn-Rapid.xml"/>
```

For Autobahn FIX platform:

```
<property name="QuickFIX.DataDictionary"
value="${APAMA_HOME}/adapters/config/FIX44-Autobahn.xml"/>
```

The IAF configuration file should contain appropriate host, port, sender and target CompID parameters.

For the MDA Mode the path of the correct adapter configuration file should also be provided using the property -

For Autobahn Rapid platform:

```
<property name="AdapterConfiguration"
value="${APAMA_HOME}/adapters/config/FIXTransport-BaseFIXLibrary-Autobahn-Rapid.xml"
/>
```

For Autobahn FIX platform:

```
<property
name="AdapterConfiguration"
value="\${APAMA_HOME}/adapters/config/FIXTransport-BaseFIXLibrary-Autobahn.xml"
/>
```

Service monitor injection order

Inject the monitors to connect to both MDA Session listed in [“FIX MDA session” on page 82](#) and [“FIX order management session” on page 82](#).

Platforms supported

Autobahn Rapid platform:

Session configuration:

MDA adapter configuration and subscription

The MDA adapter configuration file `FIXTransport-BaseFIXLibrary-Autobahn-Rapid.xml` contains properties which are used for sending subscription and unsubscription Requests. Autobahn(RAPID) supports only SPOT type subscriptions.

TRADING SESSIONS:

- - `OrderManager.GenerateNewStyleOrderIds = true`
- - `FixChannel = AUTOBAHN_FIX`

Example:

```
com.apama.fix.SessionConfiguration("AUTOBAHN_RAPID_FIX_TRADING",
{"FixChannel":"AUTOBAHN_FIX",
"OrderManager.GenerateNewStyleOrderIds":"true"})
```

Order management

For example:

```
com.apama.oms.NewOrder("order1", "EUR/USD", 1.28647,
"BUY", "MARKET", 100000, "FIX", "", "",
"AUTOBAHN_RAPID_FIX_TRADING", "", "", {"15":"EUR"})
```

Autobahn FIX platform:

Session configuration:

MDA adapter configuration and subscription

The MDA adapter configuration file `'FIXTransport-BaseFIXLibrary-Autobahn.xml'` contains properties which are used for sending subscription and unsubscription Requests. Autobahn supports SPOT,Outright and SAWP type subscriptions.

For SPOT subscription the following fields are required ,these values should send in com.apama.session.CtrlParams parms while subscription.

- 5232 Currency String (3)
- 5233 OrderQty Qty (20)
- 6215 TenorValue (n) refer the Spec of Autobahn for details.

TRADING SESSIONS:

- OrderManager.GenerateNewStyleOrderIds = true
- OrderManager.UseTransactTimeOfOrder=true
- FixChannel = AUTOBAHN_FIX

Optional configuration parameters:

OrderManager.UseTransactTimeOfOrder is a parameter of type boolean. By enabling this parameter, forwards received timestamp of Market Data message in TransactTime(60) field of a New Order Single.

Example:

```
com.apama.fix.SessionConfiguration("AUTOBAHN_FIX_TRADING",
  {"FixChannel":"AUTOBAHN_FIX",
   "OrderManager.GenerateNewStyleOrderIds":"true",
   "OrderManager.UseTransactTimeOfOrder":"true"})
```

Order management

To submit a trade, the value from MDEntryDate (272) from the Market Data should be copied into the FuttSettDate (64) field of the New Order Single message (35 = D). The user should save the current timestamp when the Market Data message is received, to be able to populate it in the TransactTime (60) field of a further New Order Single message (35 = D).

For e.x.:

```
com.apama.oms.NewOrder("order1", "EUR/USD", 1.28647, "BUY",
"MARKET", 100000, "FIX", "", "", "AUTOBAHN_FIX_TRADING", "", "",
{"15":"EUR", "64":"20140922", "59":"4", "60":"20141002-00:16:40"})
```

16 EBS Spot Ai FIX Adapter

■ Prerequisites	210
■ Connecting to EBS Ai FIX	210
■ Service monitor extensions	210
■ Service monitor injection order	210
■ Session configuration	211
■ UserLogonRequest	212
■ UserLogoutRequest	212
■ Password Management	212
■ Querying Login Response Fields	213
■ EBS Spot Ai Depth of MV	214
■ Market data subscription	214
■ Order management support	216
■ Connecting to EBS FIX 4.4	218
■ Sample Messages and events	221

This chapter describes the design, implementation and usage of the FIX EBS-FIX extensions.

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

Connecting to EBS AI FIX

IAF configuration

When acting as a client to EBS-FIX, the IAF should be configured with the EBS-FIX specific configuration file, the distribution version of which can be found here: `${APAMA_HOME}/adapters/config/FIX-EBS-FIX.xml.dist`

Note that the requirement for a EBS-FIX-specific configuration file means that an IAF using this configuration file must be used for EBS-FIX FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for EBS-FIX operation. If you need to connect to other FIX servers as well as EBS-FIX, the non-EBS-FIX sessions should be configured in a separate IAF instance. However, EBS-FIX and non-EBS-FIX session can co-exist safely within the same correlator instance.

As well as the standard host, port, sender and target CompID parameters, the following parameters must also be supplied for EBS-FIX session:

```
<property name="StaticField.body.BE.1129" value="1.7"/>
<property name="StaticField.body.A.1137" value="7"/>
```

Note:

The adapter is not backwards compatible. It will not work with version EBS AI 6.5 and prior versions

Service monitor extensions

The file `FIX_EBS_Support.mon` provides password handling and market data conversion. Therefore, the following EPL files must be injected into the correlator, after the standard FIX adapter service monitors.

- `FIX_EBS_Events.mon`
- `FIX_EBS_Support.mon`

Service monitor injection order

Inject the monitors listed in [“FIX order management session” on page 82](#) and [“FIX Legacy Market data session” on page 83](#), followed by:

- `${APAMA_HOME}/adapters/monitors/FIX_EBS_Events.mon`
- `${APAMA_HOME}/adapters/monitors/FIX_EBS_Support.mon`

Session configuration

EBS FIX Session can be configured with the following optional configuration parameters:

- `"FixChannel":"FIX"`
The channel to emit upstream events to. It defaults to FIX.
- `"EBSFIX.SERVICEID":"EBS-FIX"`
It should not be "FIX" and it should be different from "SERVICEID" if mentioned. All new/amend/cancel orders must use this service id rather than the usual "SERVICEID". It defaults to "EBS-FIX".
- `"OrderManager.OrderIDServiceName":"EBS-FIX"`
It should not be "FIX" and it should be different from "SERVICEID" if mentioned. It allows customization of clOrdId for orders. Specifying a value here will allow the EBS service to override the default behavior of order id generation in the FIX service. It defaults to "EBS-FIX".
- `"EBSFIX.EnablePriceRounding":"false"`
Setting it to "true" enables the rounding of price to symbol PIP for Orders. It defaults to "false".
- `"EBSFIX.LogonRetryCount":"2"`
Maximum user logon retry count. It defaults to 2.
- `"EBSFIX.DisableAutoPasswdGeneration":"false"`
If enabled adapter will not autogenerate new password when it receives a password expire response. Adapter will wait for a "PasswordChangeRequest" event from the user.
- `"EBSFIX.EBSOrderTypes":`
Currently, our adapter supports order types 'LIMIT', 'FIXING'. Here 'LIMIT' orders can buy/sell at a price using tag 44. The other two order types trade at a fixed price hence the prices for these orders should not be submitted. In future, if EBS introduces any new order type which support price or not then same can be configured as, "string:boolean, string:boolean, string:boolean"

For example: `"EBSFIX.EBSOrderTypes":"orderTyp1:true, orderTyp2:false, orderTyp3:true"`
- `"EBSFIX.SubKeyTags":`
A dictionary of tags:default that will be used to uniquely identify a subscription. By default the subscription key is a combination of Symbol, SettleType(63), CFICode(461). This can be modified by providing the dictionary of parameter and default values that user wants.

For example: `"EBSFIX.SubKeyTags": "{ \"64\": \"1\", \"121\": \"A\" }"`.

The subscription key will now be a combination of Symbol:64:121. If the SubscribeDepth/Tick event doesn't have these tags, the default values of these tags (1, A) will be used the subscription key.

Sample configuration event

```
com.apama.fix.ebs.SessionConfiguration("Connection Name",
{"FixChannel":"FIX", "OrderManager.OrderIDServiceName":"EBS-FIX",
"EBSFIX.SERVICEID":"EBS-FIX", "EBSFIX.EnablePriceRounding":"true",
"EBSFIX.LogonRetryCount":"2" })
```

The above configuration is equivalent to

```
com.apama.fix.ebs.SessionConfiguration("Connection Name", {})
```

UserLogonRequest

After sending SessionConfiguration and the session has started, you must send application login:

```
com.apama.fix.ebs.UserLogonRequest("Connection Name", "RequestID",
"Username", "Password", [com.apama.fix.ebs.UserData("AutoCancelDuplSession","Y",{}),
com.apama.fix.ebs.UserData("PriceCheck","N",{})], {})
```

UserData like AutoCancelDuplSession, PriceCheck, AggregationProvider, ClientType etc. can also be specified with the Logon Request as shown in above sample.

Sample user login request to get NDF data:

```
com.apama.fix.ebs.UserLogonRequest("TRANSPORT", "123", "USERNAME", "PASSWORD",
[com.apama.fix.ebs.UserData("AllowNDFSwapInfo","Y",{}),
com.apama.fix.ebs.UserData("AllowFixPointsInfo","Y",{})], {})
```

UserLogoutRequest

User can do send a application logout using:

```
com.apama.fix.ebs.UserLogoutRequest("Connection Name", "RequestID",
"Username")
```

Note:

EBS user session gets logged of but FIX session will not be disconnected.

Password Management

You can change the password using the interface:

```
PasswordChangeRequest("Connection Name", "RequestID",
"Username", "Old Pass", "New Pass");
```

If the "Password Expired" UserResponse is sent by the EBS server, then a PasswordChangeRequest is sent automatically by the by the FIX_EBS_Support.mon.

The FIX_EBS_Support.mon monitor routes a GetNewPassword("Connection Name", "Unique ID", "Old Pass") event. You can capture this event and generate a new password and send it back as:

```
NewPassword{"Connection Name", "Unique ID", "Old Pass", "New Pass"}
```

If you do not handle it, a password will be automatically generated which will be printed in the correlator log.

As monitor doesn't store password in a particular file, user needs to setup separate listeners on `com.apama.fix.ebs.PasswordChangeResponse`. A sample monitor file is provided `FIX_EBS_Support_Password.mon` which will write changed password a file whose name is mentioned in `extraParams` of event `com.apama.fix.ebs.UserLogonRequest`. These fields include:

- `AntPropertyFile` : Name of file in which changed password is written
- `AntPWProperty` : Password is written in file in format `"$Key:$password"`. This field should carry a name which can substituted in place of 'Key'.

Note:

This sample monitor is dependent on `FileEvents.mon` of File adapter.

Querying Login Response Fields

User can request for login fields in login response through `com.apama.fix.ebs.GetUserSessionParams` interface. Requested fields are exposed by `com.apama.fix.ebs.UserSessionParams` Interface.

```
com.apama.fix.ebs.GetUserSessionParams(string Transport,string
    symbol,sequence <string> fields,dictionary <string,string>
    extraParams)
```

Where,

- `Transport` is connection name
- `symbol` is interested symbol to retrieve fields
- `fields` is sequence of comma separated interested fields
- `extraParams`. You can provide `CFICode` (defaults to `RCSXXX`) and `SettlType` (defaults to `"0"` (that is, `Regular`)) in `extraParams`

Note:

`Symbol` can be empty(`""`) to retrieve fields of `UserData` in `UserResponse` Interface.

To retrieve `TradingSessions` related information, user expects to send field as `"TradingSession"`.

```
com.apama.fix.ebs.UserSessionParams(string Transport,string
    symbol,dictionary<string,string> values)
```

`Transport` is connection name, `symbol` is interested symbol to retrieve fields and .

Where,

- `Transport` is connection name
- `symbol` is interested symbol to retrieve fields
- `values` is dictionary of field versus value

Sample event without extraParams:

```
com.apama.fix.ebs.GetUserSessionParams("Connection","XAG/USD",
    ["TradingSession","SettlDate","IcebergRandomTimeIncrement"],{})
```

Sample response

```
com.apama.fix.ebs.UserSessionParams("Connection","XAG/USD",
    {"IcebergRandomTimeIncrement":"100","MarketSegmentID_0":"Standard",
    "SettlDate":"20140318","TradingSessionID_0":"56128"})
```

Sample event with extraParams:

```
com.apama.fix.ebs.GetUserSessionParams("Connection","EUR/USD",["SettlDate",
    "541","SettlType","CFIcode"],{"CFIcode":"FFCNO","SettlType":"M1"})
```

Sample response

```
com.apama.fix.ebs.UserSessionParams("Connection","USD/CNS",{"541":"20170808",
    "CFIcode":"FFCNO","SettlDate":"20170810","SettlType":"M1"})
```

EBS Spot Ai Depth of MV

```
com.apama.fix.ebs.SetMVEntry(string TRANSPORT, string serviceId, string symbol, integer
value)
```

Set or override the "Ai Depth of MV" value for a symbol. The Depth of MV value is used to determine when rates have moved too far from the current best price and should be removed from the Price Depth View. If this value is set incorrectly for a symbol, there is a risk that stale rates could appear in the published market data. Depth of MV values for each symbol are supplied by the Ai server in logon response and you use them while processing Price Depth view. The adapter is pre-configured with the Depth of MV values that were valid at the time of release. If any Depth of MV value is changed by EBS it may be necessary for users to override the pre-configured value until a new release is available. SetMVEntry events should be sent into the correlator *after* the SessionConfig event for a session but *before* any subscriptions are issued for that session.

If user wants to set MVEntry for a specific tenor the symbol name should be in the following format: SetMVEntry.symbol = Symbol:CFIcode:Tenor.

If just the symbol name is provided, then adapter will assume it to be FX SPOT (tenor as 0, and CFIcode as RCSXXX)

Market data subscription

After getting a UserLogin Success response, you can send MarketData subscriptions:

Tick subscription

Tick subscription support is enabled on "EBS-FIX" or value of "EBSFIX.SERVICEID" as specified in session configuration.

```
com.apama.marketdata.SubscribeTick("EBS-FIX","Connection
Name","EUR/USD",{"461":"RCSXXX","63":"0"})
com.apama.marketdata.UnsubscribeTick("EBS-FIX","Connection
Name","EUR/USD",{"461":"RCSXXX","63":"0"})
```

Depth subscription

- PRICE VIEW subscription/unsubscription: Price Depth View support is enabled on "EBS-FIX" or value of "EBSFIX.SERVICEID" as specified in session configuration.

```
com.apama.marketdata.SubscribeDepth("EBS-FIX","Connection
Name","EUR/USD",{ "461":"RCSXXX","63":"0"})
com.apama.marketdata.UnsubscribeDepth("EBS-FIX","Connection
Name","EUR/USD",{ "461":"RCSXXX","63":"0"}) The resultant depth due
to this subscription will have SERVICE_NAME of EBS-FIX
```

- SPREAD VIEW subscription/unsubscription:

```
com.apama.marketdata.SubscribeDepth("EBS-SPREAD","Connection
Name","EUR/USD",{ "461":"RCSXXX","63":"0"})
com.apama.marketdata.UnsubscribeDepth("EBS-SPREAD","Connection
Name","EUR/USD",{ "461":"RCSXXX","63":"0"})
```

The resultant depth due to this subscription will have SERVICE_NAME of EBS-SPREAD

- AMOUNT VIEW subscription/unsubscription:

```
com.apama.marketdata.SubscribeDepth("EBS-AMOUNT","Connection
Name","EUR/USD",{ "461":"RCSXXX","63":"0"})
com.apama.marketdata.UnsubscribeDepth("EBS-AMOUNT","Connection
Name","EUR/USD",{ "461":"RCSXXX","63":"0"})
```

The resultant depth due to this subscription will have SERVICE_NAME of EBS-AMOUNT

- FULL AMOUNT VIEW subscription/unsubscription:

```
com.apama.marketdata.SubscribeDepth("EBS-FULL-AMOUNT","Connection
Name","EUR/USD",{ "461":"RCSXXX","63":"0"})
com.apama.marketdata.UnsubscribeDepth("EBS-FULL-AMOUNT","Connection
Name","EUR/USD",{ "461":"RCSXXX","63":"0"})
```

The resultant depth due to this subscription will have SERVICE_NAME of EBS-FULL-AMOUNT

- EBS Best Prices. EBS best prices enabled for all Views. These prices will be forwarded as EBSBestBid and EBSBestOffer in extraParams of depth event.

Sample depth event:

```
com.apama.marketdata.Depth("EUR/USD", [1.3891], [1.3892], [1.3891499999999999],
[4000000], [4000000], {"1021":"2", "35":"W", "461":"RCSXXX",
"52":"20140314-11:02:52.044", "63":"0", "EBSBestBid":"1.3891",
"EBSBestOffer":"1.3892", "Market":"Connection", "SERVICE_NAME":"EBS-FIX"})
```

- Reference rate. Reference rate is enabled for all Views. These prices will be forwarded as ReferenceRate in extraParams of depth event.

Sample depth event:

```
com.apama.marketdata.Depth("EUR/USD", [1.3891], [1.3892], [1.3891499999999999],
[4000000], [4000000], {"1021":"2", "35":"W", "461":"RCSXXX",
"52":"20140314-11:02:52.044", "63":"0", "EBSBestBid":"1.3891",
"EBSBestOffer":"1.3892", "Market":"Connection", "ReferenceRate":"1.3446",
"SERVICE_NAME":"EBS-FIX"})
```

Order management support

In order to submit and manage orders to the EBS exchange, it has been necessary to define a custom pre-processor module that works in conjunction with the standard FIX order manager. The pre-processor is used by routing a new order to the service id defined in session configuration using `EBSFIX.SERVICEID` parameter. The preprocessor will modify the order to make it suitable for the EBS market before forwarding it on to the usual order manager service and listening for responses (executions etc).

Order placing

Order management support is enabled on "EBS-FIX" or value of `EBSFIX.SERVICEID` as specified in the session configuration. Orders must be placed only after subscribing to the Instrument. Time in Force(59) values can be - Good Till Cancelled (59=1,Quote) , Immediate or Cancel (59=3,Hit) and (59=4) Fill or Kill (Full Amount trading for Direct pairs).

```
com.apama.oms.NewOrder("10","EUR/USD",1.2675,"BUY","LIMIT",20000000,"EBS-FIX",
    "", "", "ConnectionName", "", "", {"461":"RCSXXX", "63":"0", "59":"1"})
```

According to the specification CancelOrder should have Order Type, so

1. For limit orders, add "40": "2" to the extraParams.

```
com.apama.oms.CancelOrder("10", "EBS-FIX",
    {"461":"RCSXXX", "63":"0", "59":"1", "40":"2"})
```

2. For Fixing orders, add "40": "Z" to the extraParams.

```
com.apama.oms.CancelOrder("10", "EBS-FIX",
    {"461":"RCSXXX", "63":"0", "59":"1", "40":"Z"})
```

Iceberg enabled order

Iceberg Enabled order should contain `DisplayQty(1138)` in extraParams. You can also specify `Display method(1084)` and `IcebergHigh RandomTime(20112)`. If you don't specify it will take default values. To amend iceberg enabled order, you should send the display quantity in amend order. Sample events:

```
com.apama.oms.NewOrder("order4","EUR/USD",1.3762,"SELL","LIMIT",20000000,"EBS-FIX",
    "", "", "Connection", "", "", {"461":"RCSXXX", "59":"1", "63":"0", "1138":"20000000", "1084":"1",
    "20112":"100"})
com.apama.oms.AmendOrder("order4","EBS-FIX","EUR/USD",1.3785,"SELL","LIMIT",22000000,
    {"461":"RCSXXX", "59":"1", "63":"0", "1138":"10000000"})
```

FIXING Order

If a user is interested in placing 'FIXING' orders then you should request the same in user login request by adding parameter 'AllowFixingInfo' in UserData as:

```
com.apama.fix.ebs.UserLogonRequest("Connection", "123", "SAG",
    "APAMA_123", [com.apama.fix.ebs.UserData("AllowFixingInfo", "Y", {})], {})
```

Once logged in successfully, you will get a trading session status message which provides information on the status of the market. This message carries trading ID, its status, ValuationDateTime etc. To place a fixing order, EBS expects us to send mandatory tags - TimeInForce (Tag 59) MarketSegmentID (Tag 1300) TradingSessionID (Tag 336) ValuationDateTime (Tag 20107)

Client just needs to send an extra parameter 'TradingSessionID' with a value in the NewOrder event and adapter will map remaining tags from its cache.

For example:

```
com.apama.oms.NewOrder("1","USD/JPY",101.91,"SELL","FIXING",3000000,"EBS-PRICE",
    "", "", "Connection", "", "", {"461":"RCSXXX", "59":"1", "63":"0", "TradingSessionID":"6"})
```

For amending a fixing order, you don't need these mandatory tags except TimeInForce which will be taken care by adapter.

Party Info

PartyInfo which is received as part of TradeCaptureReport, will be published in OrderUpdate as follows

```
Sample Order Update:
com.apama.oms.OrderUpdate("order1","EUR/USD",1.41125,"BUY","LIMIT",5000000,
    true,true,true,false,false,false,false,"xyz",1437,4998563,1437,147,147,
    "PartiallyFilled":{"35":"8","461":"RCSXXX","52":"20110711-10:54:07.501",
    "59":"1","63":"0","Account":"","ExecID":"123","ExecType":"F",
    "Market":"Connection","OrdStatus":"1","Party0":"Progress,BIC,ExecutingBroker",
    "SERVICE_NAME":"EBS-PRICE"})
```

The PartyInfo in the above example is "Party0":"Progress,BIC,Executing Broker" Where, "Party_x":"PartyID, PartyIDSource, PartyRole"

How does our adapter/monitor handle TCR and ER?

In EBS support monitor, on receiving an execution report with a partial fill or fill, you will open a listener for corresponding TCR with same matching Id. This execution report will be sent to base FIX Ordermanager after receiving matching TCR. In some cases though you receive an ER with filled state but still you need to receive corresponding TCR. Meanwhile user tries to cancel order but gets rejected by exchange. To handle such scenarios, use `__APAMA_ORDER_STATE` extra parameter to handle order completion as stated below,

1. On receiving final execution report, EBS support monitor will add an extra parameter `__APAMA_ORDER_STATE : 'final'`

This parameter says that the order has reached done state but still need to receive TCRs which carry settlement details.

2. On receiving all TCRs, support monitor will route an orderupdate with `__APAMA_ORDER_STATE` as 'settled' and kill its thread.

In EBS, an OrderUpdate with `__APAMA_ORDER_STATE: 'settled'` will be the last after which order can be considered done.

Connecting to EBS FIX 4.4

EBS adapter supports EBS Gateway version of EBS. Each connection to this platform will require two separate FIX sessions per client; one dedicated to market data messages, and the other dedicated to order management related messages (Order). So, Market data can be accessed via CMF MDA and orders can be placed using `com.apama.oms` interface.

IAF configuration

When acting as a client to EBS Gateway(EBS FIX 4.4), the IAF should be configured with the EBS FIX specific configuration file, the distribution version of which can be found here: `${APAMA_HOME}/adapters/config/FIX-EBS-MDA.xml.dist`

For Market data the path of the correct adapter configuration file should also be provided using the property

```
<property name="AdapterConfiguration"
value="${APAMA_HOME}/adapters/config/FIXTransport-BaseFIXLibrary-EBS.xml" />
```

Note that the requirement for a EBS-FIX specific configuration file means that an IAF using this configuration file must be used for EBS FIX 4.4 sessions only. The FIX adapter will not inter-operate correctly with other FIX servers once it has been configured for EBS FIX 4.4 operation. If you need to connect to other FIX servers as well as EBS FIX 4.4, the non-EBS FIX 4.4 sessions should be configured in a separate IAF instance. However, EBS FIX 4.4 and non-EBS FIX 4.4 session can co-exist safely within the same correlator instance. As well as the standard host, port, sender and target CompID parameters, the following parameters must also be supplied for marketData session:

```
<!-- Username (supplied
by EBSGW) --> <property name="StaticField.body.A.553"
value="@USERNAME@" /> <!-- Password (supplied by EBSGW) -->
<property name="StaticField.body.A.554" value="@PASSWORD@" />
<property name="StaticField.body.A.1408" value="1.1"/>
```

Service monitor injection order

Inject the monitors to connect to both MDA Session and legacy finance support interfaces listed in [“FIX MDA session” on page 82](#) and [“FIX order management session” on page 82](#), followed by:

- `${APAMA_HOME}/adapters/monitors/FIX_EBS_Support.mon`
- `${APAMA_HOME}/adapters/monitors/FIX_EBS_Events.mon`

Session configuration

EBS FIX 4.4 uses `com.apama.oms` interface for placing orders and CMF MDA for marketData.

Adapter configuration

The Adapter Configuration file `FIXTransport-BaseFIXLibrary-EBS.xml` contains properties which are used for sending subscription and unsubscription Requests.

EBS-FIX Trading sessions must be configured with the following configuration parameters:

- `FixChannel = FIX`
- `OrderManager.updateKeyParamsOnStateChange = true`

Subscribing and Unsubscribing

You should subscribe for the symbol before placing order.

Sample configuration:

```
com.apama.fix.SessionConfiguration("EBSGW_TRADE_DATA",
  {"FixChannel":"FIX_CHANNEL","OrderManager.updateKeyParamsOnStateChange":"true"})
```

The EBS FIX 4.4 adapter uses the CMF MDA MBP(`com.apama.md.D`), MBO(`com.apama.md.O`) and BBA(`com.apama.md.BBA`) streams to publish the EBS marketData . For EBS "PriceDepth Book" updates, you must be subscribed to MBP(`com.apama.md.D`) stream. For "MultiLevel PriceDepth Book" updates, you must be subscribed to MBO(`com.apama.md.O`) stream. With BBA stream, you will receive the "top of book" updates of EBS "PriceDepth Book".

You can also do the subscription for specific quantity by giving the quantity value in control parameter "QUANTITY_RUNGS", and it is supported only for EBS "MultiLevel PriceDepth Book".

Example:

```
subscription - EUR/USD(symbol) with 15000000 quantity rungs,
com.apama.session.CtrlParams controlParams := new
com.apama.session.CtrlParams;
controlParams.addParam("QUANTITY_RUNGS","15000000")
```

if customer is interested to subscribe non-aggregated PriceDepth Book then they must use MBO stream with below control parameters.

```
com.apama.session.CtrlParams controlParams := new
com.apama.session.CtrlParams; controlParams.addParam("1300","D")
controlParams.addParam("265","1") for Incremental updates. If not
specified it will always get a Snapshot update.
```

EBS Book vs CMF MDA streams

EBS Book vs CMF MDA streams and default control parameters (which are added by adapter for subscriptions)

EBS Book Type	MDA Streams	Default tags
PriceDepth	MBP(com.apama.md.D)	1300(MarketSegmentID) = D 1021(MDBookType) = 2(PriceDepth) 167(SecurityType) = "FXSPOT" 264(MarketDepth) = 0(Full Book) 265(MDUpdateType) = 0(Snapshot)
MultiLevelPriceDepth	MBO(com.apama.md.O)	1300(MarketSegmentID) = DF 1021(MDBookType) = 1104(MultiLevelPriceDepth) 167(SecurityType) = "FXSPOT" 264(MarketDepth) = 0(Full Book) 265(MDUpdateType) = 0(Snapshot)
PriceDepth (Top of Book	BBA(com.apama.md.BBA)	1300(MarketSegmentID) = D 1021(MDBookType) = 2(PriceDepth) 167(SecurityType) = "FXSPOT" 264(MarketDepth) = 1(Top of Book) 265(MDUpdateType) = 0(Snapshot)
PriceDepth (Non-aggregated)	MBO(com.apama.md.O)	1021(MDBookType) = 2(PriceDepth) 167(SecurityType) = "FXSPOT" 264(MarketDepth) = 0(Full Book) 265(MDUpdateType) = 0(Snapshot)
MultiLevelPriceDepth with Quantity Rung	MBO(com.apama.md.O)	1300(MarketSegmentID) = DF 1021(MDBookType) = 1104(MultiLevelPriceDepth) 167(SecurityType) = "FXSPOT" 264(MarketDepth) = 0(Full Book) 265(MDUpdateType) = 0(Snapshot) 12002(MDRungQty) = the value of "QUANTITY_RUNGS" controlParams which is given by user.

EBS FIX 4.4 order management

As EBS FIX 4.4 adapter uses OMS interface for an order management, ServiceName should be "FIX".

Sending orders using OMS interface NewOrder(NewOrderSingle)

Supply following parameter values:

- MarketSegmentID (Tag 1300) : EBS Market Segment of the instrument
- Currency (Tag 15) :- Supply the currency used for price.
- Account (Tag 1) :- Trader's floor code.
- SecurityType (Tag 167) .

Sample new order:

```
com.apama.oms.NewOrder("6","EUR/USD",1.12905,"BUY","LIMIT",9000000,"FIX","",
    "", "EBSGW_TRADE_DATA", "", "", {"15": "EUR", "1": "GWC7", "1300": "DF", "167": "FXSPOT"})
```

To place order for specific LP, you need to provide LP details in "Header:128". For example:

```
com.apama.oms.NewOrder("10","EUR/USD",1.12678,"BUY","LIMIT",5000000,
    "FIX","", "", "EBSGW_TRADE_DATA", "", "", {"15": "EUR", "1": "GWC7",
    "1300": "D", "167": "FXSPOT", "63": "0", "Header:128": "5A0B,6A0B", "59": "0"})
```

To place GTD/GTA orders , you need to pass related tags in extraParams. For Example:

```
com.apama.oms.NewOrder("9","EUR/USD",1.12678,"BUY","LIMIT",9000000,
    "FIX","", "", "EBSGW_TRADE_DATA", "", "", {"15": "EUR", "1": "GWC7", "1300": "D",
    "167": "FXSPOT", "63": "0", "59": "6", "126": "20142121-00:05:00.000"})
```

Note:

Make sure that no extraParams are provided in cancel order.

Sample cancel order:

```
com.apama.oms.CancelOrder("6","FIX",{})
```

Sample Messages and events

EBS Ai Sample Market Views

Currency Pair = EUR/USD

Order Book	Depth View		
	Level	Price	Size
<hr/>			

Order Book			Depth View			
#	BID	Size	Best	1.37732	2M	
1	1.37732	2M	+1	1.37731	1M	
2	1.37731	1M	+2	1.37729	3M	None at 1.37730
3	1.37730	0M	+3	1.37728	1M	
4	1.37729	3M	+4	1.37727	1M	
5	1.37728	1M	+5	1.37726	1M	
6	1.37727	1M	+6	1.37725	1M	
7	1.37726	1M	+7	1.37724	2M	
8	1.37725	2M	+8	1.37723	2M	
9	1.37724	2M	+9	1.37719	3M	
10	1.37723	3M	+10	1.37718	1M	
11	1.37722	0M				
12	1.37721	0M	Spread View			
13	1.37720	0M	Level	Price	Size	
14	1.37719	1M	Best	1.37732	2M	
15	1.37718	1M	-0.5pips	1.37727	6M	sum of inventory 1.37731 to 1.37727
16	1.37717	1M	-1.0pips	1.37722	8M	sum of inventory 1.37726 to 1.37722
17	1.37716	1M	-1.5pips	1.37717	3M	&c.
18	1.37715	0M	-2.0pips	1.37712	3M	&c.
19	1.37714	0M				
20	1.37713	1M	Amount View			
21	1.37712	1M	Level	Price	Size	
22	1.37711	0M	Best	1.37732	2M	
23	1.37710	1M	5M	1.37729	n/a	
24	1.37709	1M	10M	1.37725	n/a	
25	1.37708	1M	15M	1.37723	n/a	

Order Book			Depth View		
26	1.37707	1M	20M	1.37716	n/a
27	1.37706	1M	25M	1.37708	n/a
28	1.37705	3M	30M	1.37705	n/a

FIX based Messages

Market Data Snapshot – Price Depth view

FIX message

```
35=W^262=123^1021=2^55=EUR/SD^461=RCSXXX^63=0^268=11^269=0^271
=2^270=1.37732^269=0^271=1^270=1.37731^269=0^271=3^270=1.37729^269
=0^271=1^270=1.37728^269=0^271=1^270=1.37727^269=0^271
=1^270=1.37726^269=0^271=2^270=1.37725^269=0^271=2^270=1.37724^269
=0^271=3^270=1.37723^269=0^271=1^270=1.37719^269=0^271=1^270=1.37718^
```

Tag name	Tag number	Value	Description
Header	35	W	Snapshot Full Refresh
MDReqID	262	123	My Request ID
MDBookType	1021	2	PriceDepth
Symbol	55	EUR/USD	Currency Pair
CFICode	461	RCSXXX	FX Spot
SettlType	63	0	Regular Spot Settlement
NoMDEntries	268	11	Number of repeating entries
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	2	Quantity
MDEntryPx	270	1.37732	Price -Dealable Best
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	1	Quantity
MDEntryPx	270	1.37731	Price

Tag name	Tag number	Value	Description
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	1	Quantity
MDEntryPx	270	1.37729	Price
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	1	Quantity
MDEntryPx	270	1.37728	Price
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	1	Quantity
MDEntryPx	270	1.37727	Price
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	1	Quantity
MDEntryPx	270	1.37726	Price
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	2	Quantity
MDEntryPx	270	1.37725	Price
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	2	Quantity
MDEntryPx	270	1.37724	Price
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	3	Quantity
MDEntryPx	270	1.37723	Price
MDEntryType	269	0	A Buy Side Snapshot

Tag name	Tag number	Value	Description
MDEntrySize	271	1	Quantity
MDEntryPx	270	1.37719	Price
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	1	Quantity
MDEntryPx	270	1.37718	Price

Price Depth view Constructed from FIX message

Price Depth View - BID

Price	Size
1,37732	2
1,37731	1
1,37729	3
1,37728	1
1,37727	1
1,37726	1
1,37725	2
1,37724	2
1,37723	3
1,37719	1
1,37718	1

```
com.apama.marketdata.Depth("EUR/USD ", [1.37732,1.37731,1.37729,1.37728,
1.37727,1.37726,1.37725,1.37724,1.37723,1.37719,1.37718], [], [-1,-1,-1,-1,-1],
[ 2,1,3,1,1,1,2,2,3,1,1], [],
{" 461=RCSXXX^63=0,"Market": "Transport", "SERVICE_NAME": "EBS-FIX"})
```

Market Data Snapshot – Spread View

The Spread view provides the view of the market at a price offset from the Deable Best.

FIX message

35=W^262=123^1021=1101^55=EUR/USD^461=RCSXXX^63=0^268=5^269=
 0^271=2^270=1.37732^269=0^271=6^270=1.37727^269=0^271=
 8^270=1.37722^269=0^271=3^270=1.37717^269=0^271=3^270= 1.37712^

Tag name	Tag number	Value	Description
Header	35	W	Snapshot Full Refresh
MDReqID	262	123	My Request ID
MDBookType	1021	2	PriceDepth
Symbol	55	EUR/USD	Currency Pair
CFICode	461	RCSXXX	FX Spot
SettlType	63	0	Regular Spot Settlement
NoMDEntries	268	5	Number of repeating entries
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	2	Quantity
MDEntryPx	270	1.37732	Price
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	6	Quantity
MDEntryPx	270	1.37727	Price
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	8	Quantity
MDEntryPx	270	1.37722	Price
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	3	Quantity
MDEntryPx	270	1.37717	Price
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	3	Quantity

Tag name	Tag number	Value	Description
MDEntryPx	270	1.37712	Price
Trailer			

Spread view Constructed from FIX message

Price Depth View - BID

Price	Size
1.37732	2
1.37727	6
1.37722	8
1.37717	3
1.37712	3

```
com.apama.marketdata.Depth("EUR/USD ",
[1.37732, 1.37727, 1.37722, 1.37717],
[], [-1, -1, -1, -1, -1], [ 2, 6, 8, 3, 3], [],
{" 461=RCSXXX^63=0,"Market": "Transport", "SERVICE_NAME": "EBS-SPREAD"})
```

Market Data Snapshot – Amount View

Amount view provides the maximum price for a given amount at which some of the orders will be filled.

Say for preconfigured amounts - 5, 10, 15, 20, 25, 30 and 35

FIX message

```
35=W^262=123^1021=1102^55=EUR/USD^461=RCSXXX^63=0^268=6^269=0^271=5^270=1.37729^269=0^271=10^270=1.37725^269=0^271=15^270=1.37723^269=0^271=20^270=1.37716^269=0^271=25^270=1.37708^269=0^271=30^270=1.37705^
```

Tag name	Tag number	Value	Description
Header	35	W	Snapshot Full Refresh
MDReqID	262	123	My Request ID
MDBookType	1021	2	PriceDepth
Symbol	55	EUR/USD	Currency Pair

Tag name	Tag number	Value	Description
CFICode	461	RCSXXX	FX Spot
SettlType	63	0	Regular Spot Settlement
NoMDEntries	268	6	Number of repeating entries
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	6	Quantity
MDEntryPx	270	1.37729	Price
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	10	Quantity
MDEntryPx	270	1.37725	Price
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	15	Quantity
MDEntryPx	270	1.37723	Price
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	20	Quantity
MDEntryPx	270	1.37716	Price
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	25	Quantity
MDEntryPx	270	1.37708	Price
MDEntryType	269	0	A Buy Side Snapshot
MDEntrySize	271	30	Quantity
MDEntryPx	270	1.37705	Price
Trailer			

Amount view Constructed from FIX message

Amount View - BID

Price	Size
1.37729	5
1.37725	10
1.37723	15
1.37716	20
1.37708	25
1.37705	30

```
com.apama.marketdata.Depth("EUR/USD ",
[1.37729, 1.37725, 1.37723, 1.37716, 1.37708, 1.37705],
[], [-1, -1, -1, -1, -1], [5, 10, 15, 20, 25, 30], [],
{"461=RCSXXX^63=0","Market":"Transport","SERVICE_NAME":"EBS-AMOUNT"}
}
```

EBS FIX handling of different views

Session configuration

EBS FIX Session can be configured with the following optional configuration parameters

- `"FixChannel":"FIX"`
The channel to emit upstream events to. It defaults to FIX.
- `"EBSFIX.SERVICEID":"EBS-FIX"`
It should not be "FIX" and it should be different from "SERVICEID" if mentioned. All new/amend/cancel orders must use this service id rather than the usual "SERVICEID". It defaults to "EBS-FIX".
- `"OrderManager.OrderIDServiceName":"EBS-FIX"`
It should not be "FIX" and it should be different from "SERVICEID" if mentioned. It allows customization of clOrdId for orders. Specifying a value here will allow the EBS service to override the default behavior of order id generation in the QFIX service. It defaults to "EBS-FIX".
- `"EBSFIX.EnablePriceRounding":"false"`
Setting it to "true" enables the rounding of price to symbol PIP for Orders. It defaults to "false".
- `"EBSFIX.LogonRetryCount":"2"`
Maximum user logon retry count. It defaults to 2.

Sample configuration event:

```
com.apama.fix.ebs.SessionConfiguration("Connection Name",
```

```
{“FixChannel”:"FIX",“OrderManager.OrderIDServiceName”:"EBS-FIX",
“EBSFIX.SERVICEID”:"EBS-FIX",
“EBSFIX.EnablePriceRounding”:"true”,
“EBSFIX.LogonRetryCount”:"2”})
```

The above configuration is equivalent to `com.apama.fix.ebs.SessionConfiguration(“Connection Name”, {})`

Tick subscription

Tick subscription support is enabled on "EBS-FIX" or value of "EBSFIX.SERVICEID" as specified in session configuration.

Subscribe Tick:

```
com.apama.marketdata.SubscribeTick(“EBS-FIX”, “Connection Name”, “EUR/USD”,
{“461”:"RCSXXX”, “63”:"0”})
```

Generated Tick information:

```
com.apama.marketdata.Tick(“AUD/JPY”, 84.055, 1, {“1021”:"2”, “461”:"RCSXXX”,
“5450”:"11”, “63”:"0”, “CFIcode”:"RCSXXX”, “Market”:" Connection Name”,
“SERVICE_NAME”:"EBS- FIX”, “SettlmntTyp”:"0”})
```

```
com.apama.marketdata.Tick(“AUD/JPY”, 83.055, 1, {“1021”:"2”, “461”:"RCSXXX”,
“5450”:"12”, “63”:"0”, “CFIcode”:"RCSXXX”, “Market”:" Connection Name”,
“SERVICE_NAME”:"EBS-FIX”, “SettlmntTyp”:"0”})
```

Un-Subscribe Tick:

```
com.apama.marketdata.UnsubscribeTick(“EBS-FIX”, “Connection Name”, “EUR/USD”,
{“461”:"RCSXXX”, “63”:"0”})
```

Depth subscription

Three different depth views can be subscribed as shown below

Price Depth View

Price Depth View support is enabled on "EBS-FIX" or value of "EBSFIX.SERVICEID" as specified in session configuration.

Subscribe Depth:

```
com.apama.marketdata.SubscribeDepth(“EBS-FIX”, “Connection Name”, “EUR/USD”,
{“461”:"RCSXXX”, “63”:"0”})
```

Generated Depth:

```
com.apama.marketdata.Depth("EUR/USD",
[1.3283,1.3282,1.3281,1.328,1.3279,1.3278,1.3277,1.3276,1.3275,1.3274],
[1.3285,1.3286,1.3287,1.3288,1.3289,1.329,1.3291,1.3292,1.3293,1.3294],
[1.3284,1.3284,1.3284,1.3284,1.3284,1.3284,1.3284,1.32839998,1.328399998,1.328399998],
[2000000,4000000,10000000,8000000,9000000,6000000,7000000,3000000,9000000,1000000],
[2000000,4000000,10000000,8000000,9000000,6000000,7000000,3000000,9000000,1000000],
```

```
{
  "1021": "2",
  "20203": "0",
  "35": "X",
  "461": "RCSXXX",
  "52": "20110105-07:56:04.475",
  "63": "0",
  "ASK10_CFICode": "RCSXXX",
  "ASK10_SettlmntTyp": "0",
  "ASK1_CFICode": "RCSXXX",
  "ASK1_SettlmntTyp": "0",
  "ASK2_CFICode": "RCSXXX",
  "ASK2_SettlmntTyp": "0",
  "ASK3_CFICode": "RCSXXX",
  "ASK3_SettlmntTyp": "0",
  "ASK4_CFICode": "RCSXXX",
  "ASK4_SettlmntTyp": "0",
  "ASK5_CFICode": "RCSXXX",
  "ASK5_SettlmntTyp": "0",
  "ASK6_CFICode": "RCSXXX",
  "ASK6_SettlmntTyp": "0",
  "ASK7_CFICode": "RCSXXX",
  "ASK7_SettlmntTyp": "0",
  "ASK8_CFICode": "RCSXXX",
  "ASK8_SettlmntTyp": "0",
  "ASK9_CFICode": "RCSXXX",
  "ASK9_SettlmntTyp": "0",
  "BID10_CFICode": "RCSXXX",
  "BID10_SettlmntTyp": "0",
  "BID1_CFICode": "RCSXXX",
  "BID1_SettlmntTyp": "0",
  "BID2_CFICode": "RCSXXX",
  "BID2_SettlmntTyp": "0",
  "BID3_CFICode": "RCSXXX",
  "BID3_SettlmntTyp": "0",
  "BID4_CFICode": "RCSXXX",
  "BID4_SettlmntTyp": "0",
  "BID5_CFICode": "RCSXXX",
  "BID5_SettlmntTyp": "0",
  "BID6_CFICode": "RCSXXX",
  "BID6_SettlmntTyp": "0",
  "BID7_CFICode": "RCSXXX",
  "BID7_SettlmntTyp": "0",
  "BID8_CFICode": "RCSXXX",
  "BID8_SettlmntTyp": "0",
  "BID9_CFICode": "RCSXXX",
  "BID9_SettlmntTyp": "0",
  "Market": "Connection",
  "SERVICE_NAME": "EBS-FIX"
}
```

In case of no market:

```
com.apama.marketdata.Depth("EUR/USD", [], [], [], [], [], {
  "1021": "2",
  "461": "RCSXXX",
  "63": "0",
  "Market": "Connection Name",
  "SERVICE_NAME": "EBS-FIX",
  "_ERROR": "Subscription Error: Session Connection has been logged-off"
})
```

Un-subscribe Depth:

```
com.apama.marketdata.UnsubscribeDepth("EBS-FIX", "Connection Name",
"EUR/USD", {
  "461": "RCSXXX",
  "63": "0"
})
```

Spread view

Subscribe Depth:

```
com.apama.marketdata.SubscribeDepth("EBS-SPREAD", "Connection Name",
"EUR/USD", {
  "461": "RCSXXX",
  "63": "0"
})
```

Generated Depth:

```
com.apama.marketdata.Depth("EUR/USD", [1.3266,1.3256,1.3246,1.3236,1.3226],
[1.3308,1.3318,1.3328,1.3338,1.3348], [1.3287,1.3287,1.3287,1.3287,1.3287],
[6580000000,6580000000,6580000000,6580000000,6580000000],
[6580000000,6580000000,6580000000,6580000000,6580000000],
{"1021": "1101", "20203": "0", "35": "X", "461": "RCSXXX", "52": "20110105-07:51:10.974",
"63": "0", "BID1_CFICode": "RCSXXX", "BID1_SettlmntTyp": "0", "BID2_CFICode": "RCSXXX",
"BID2_SettlmntTyp": "0", "BID3_CFICode": "RCSXXX", "BID3_SettlmntTyp": "0",
"BID4_CFICode": "RCSXXX", "BID4_SettlmntTyp": "0", "BID5_CFICode": "RCSXXX",
"BID5_SettlmntTyp": "0", "Market": "Connection", "SERVICE_NAME": "EBS-SPREAD"
})
```

In case of no market:

```
com.apama.marketdata.Depth("EUR/USD", [], [], [], [], [], {
  "1021": "2",
  "461": "RCSXXX",
  "63": "0",
  "Market": "Connection Name",
  "SERVICE_NAME": "EBS-SPREAD",
  "_ERROR": "Subscription Error: Session Connection has been logged-off"
})
```

Un-subscribe Depth:

```
com.apama.marketdata.UnsubscribeDepth("EBS-SPREAD", "Connection Name",
"EUR/USD", {
  "461": "RCSXXX",
  "63": "0"
})
```

Amount view

Subscribe Depth:

```
com.apama.marketdata.SubscribeDepth("EBS-AMOUNT", "Connection Name",
"EUR/USD", {"461":"RCSXXX", "63":"0"})
```

Generated Depth:

```
com.apama.marketdata.Depth("EUR/USD", [1.3272,1.3271,1.327], [1.3289,1.329,1.3291],
[1.32805,1.32805,1.32805], [50000000,150000000,250000000],
[50000000,150000000,250000000],
{"1021":"1102", "20203":"1", "35":"X", "461":"RCSXXX", "52":"20110105-07:46:37.472",
"63":"0",
"ASK1_CFICode":"RCSXXX", "ASK1_SettlmntTyp":"0",
"ASK2_CFICode":"RCSXXX", "ASK2_SettlmntTyp":"0",
"ASK3_CFICode":"RCSXXX", "ASK3_SettlmntTyp":"0",
"ASK4_CFICode":"RCSXXX", "ASK4_SettlmntTyp":"0",
"ASK5_CFICode":"RCSXXX", "ASK5_SettlmntTyp":"0",
"BID1_CFICode":"RCSXXX", "BID1_SettlmntTyp":"0",
"BID2_CFICode":"RCSXXX", "BID2_SettlmntTyp":"0",
"BID3_CFICode":"RCSXXX", "BID3_SettlmntTyp":"0",
"BID4_CFICode":"RCSXXX", "BID4_SettlmntTyp":"0",
"BID5_CFICode":"RCSXXX", "BID5_SettlmntTyp":"0",
"Market":"Connection", "SERVICE_NAME":"EBS-AMOUNT"})
```

In case of no market:

```
com.apama.marketdata.Depth("EUR/USD", [], [], [], [], [],
{"1021":"1102", "461":"RCSXXX", "63":"0", "Market":"Connection Name",
"SERVICE_NAME":"EBS-AMOUNT",
"_ERROR":"Subscription Error: Session Connection has been logged-off"})
```

Un-subscribe Depth:

```
com.apama.marketdata.UnsubscribeDepth("EBS-AMOUNT", "Connection Name",
"EUR/USD", {"461":"RCSXXX", "63":"0"})
```

17 FXall TCPI Server Adapter

■ Prerequisites	234
■ Transport properties configuration	234
■ Service monitor injection order	235
■ Service monitor and session configuration	235
■ Transport events	237
■ Working of the subscription provider	238
■ Order management	240
■ Info Management	240
■ Sample events	241
■ Reject Codes for Deals and QuoteRequests	242
■ Status reporting events	243

The FXall TCPI offers Provider connectivity to the FXall system through TCPI protocol version 4.1.0.1.

Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

Transport properties configuration

FXall adapter needs the following details to connect to FXall Exchange:

1. Transport Name. Replace "@FXALL_TRANSPORT_NAME@" in the <transport> element with the name of this transport instance. For example, FXALL_TCPI_TRANSPORT. The parameter (@FXALL_TRANSPORT_NAME@) must be provided in the session parameter event (Refer to SAMPLE CONFIGURATION section for a Sample session configuration Event). Property "@FXALL_TCPI_URL@" in <transport> element for URL of the Exchange. to connect to Properties "@FXALL_USERNAME@" and "@FXALL_PASSWORD@" to give provider credentials example:

```
<property name="connectionURL" value="ssl:64.209.231.116:443" />
<property name="username" value="abcd"/>
<property name="password" value="abcd123"/>
```

2. Correlator configuration. Replace "@CORRELATOR_HOST@" and "@CORRELATOR_PORT@" with the hostname and listening port of your correlator instance. Replace @FXALL_CHANNEL_NAME@ with the service monitors by default assume the name of the channel to be same as transport name.
3. Java configuration. Replace "@SSL_CA_CERTIFICATES_DIR@" in Java element with the SSL certificate directory will be provided by FXall Replace "@ADAPTERS_JARDIR@" is the location for the Transport library it is generally Apama installation directory \${APAMA_HOME}/adapters/lib. Replace "@TCPI_JARDIR@" with the directory of the TCPI jar provided by the FXall. For example,

```
<classpath path="C:/Program
Files/SoftwareAG/Apama/adapters/lib/FXall_TCPItransport.jar"/>
<classpath path="C:/FXall/TCPI/4.1.0.1/tcpi-4.1.0.1.jar" />
<classpath path="C:/Program Files/SoftwareAG/Apama/adapters/lib/JNullCodec.jar" />
<property name="SSL_CA_CERTIFICATES_DIR"
value="C:/FXall/TCPI/4.1.0.1/utilities/TCPIcerts" />
```

4. <property name="worker.thread.count" value="5"/>. This parameter is used to specify the number of threads for processing quotes in Upstream.
5. <property name="request.thread.count" value="5"/>. This parameter is used to specify the number of threads that handle requests from the Exchange in downstream.

6. `<property name="ExtractOrderValuesOnKeys" value="MIC_CODE,REG_VENUE,CUSTOMER_FED"/>`. This parameter is used to retrieve the venue, order and account information from Quote/Deal requests based on the defined keys.

The value must be in the string format with a comma separator. For example:

```
<property name="ExtractOrderValuesOnKeys" value="MIC_CODE,REG_VENUE,CUSTOMER_FED"/>
```

where keys are:

- MIC_CODE - to get static TR Fxall RFQ value of "TRAL"
- REG_VENUE - the venue of the trade-MTF/SEF
- CUSTOMER_FED - customer financial entity definition value

Note:

For more information on supported keys, see FXall TCPI document.

Service monitor injection order

The following monitors need to be injected into the correlator (in the following order) to support the adapter. All paths are relative to the Apama installation directory:

- `${APAMA_HOME}/adapters/monitors/StatusSupport.mon`
- `${APAMA_HOME}/adapters/monitors/IAFStatusManager.mon`
- `${APAMA_FOUNDATION_HOME}/ASB/projects/Legacy Finance Support/Legacy Finance Support.cdp`
- `${APAMA_HOME}/adapters/monitors/FXall_TCPI_ConfigEvents.mon`
- `${APAMA_HOME}/adapters/monitors/FXall_TCPI_IAFEvents.mon`
- `${APAMA_HOME}/adapters/monitors/FXall_TCPI_MultiContextSupport.mon`
- `${APAMA_HOME}/adapters/monitors/FXall_TCPI_RejectionCodes.mon`
- `${APAMA_HOME}/adapters/monitors/FXall_TCPI_StatusManager.mon`
- `${APAMA_HOME}/adapters/monitors/FXall_TCPI_StatusPublisher.mon`
- `${APAMA_HOME}/adapters/monitors/FXall_TCPI_SessionManager.mon`
- `${APAMA_HOME}/adapters/monitors/FXall_TCPI_QuoteManager.mon`
- `${APAMA_HOME}/adapters/monitors/FXall_TCPI_DealManager.mon`

Service monitor and session configuration

The FXall_TCPI Adapter must be configured by routing the following event after all of the monitors are injected:

```
event com.apama.fxall.tdpi.SessionConfiguration
```

```
{
  string transportName;
  dictionary<string,string> configuration;
}
```

The "transportName" field gives the name of the FXALL_TCPI "session" this must match the transport name in the IAF configuration file.

The "configuration" dictionary specifies configuration settings for the session. Currently the following keywords are supported:

Parameter	Type	Description
channel	string Default: Transport Name	Channel on which the IAF is name configured to listen.
supportMultiContext	boolean Default: true	The Adapter Supporting the multicontext applies only to QuoteRequests.
OrdersUseMultiContext	boolean Default: false	To support orders in multicontext, it only works if support supportMultiContext is also set to true.
heartbeatInterval	float Default: 5.0	Rate at which the heartbeats have to be sent to the transport.
reconnectInterval	integer Default: 60	Time before which the adapter should try to reconnect to the FXall System if the connection fails.
connectAtStartup	boolean Default: true	If set to false, Adapter expect an external doConnect event from user.
InfoRequestLiveTime	float Default: 120	Time within which a Info Request is expecting response from the Application.

Example: `com.apama.fxall.tdpi.SessionConfiguration("FXALL_TCPI_TRANSPORT", {"OrdersUseMultiContext":"true"})`

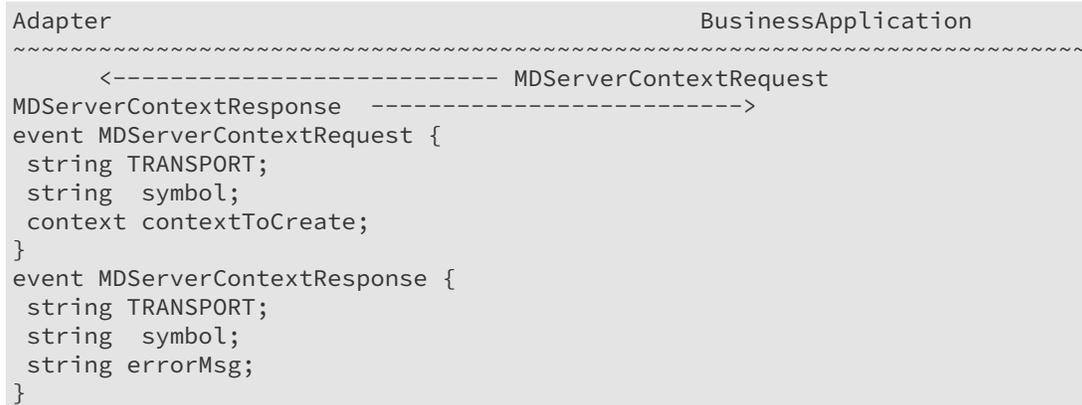
Setting up contexts

By enabling `supportMultiContext` session parameter, Adapter processes subscriptions in multi context mode. By enabling `OrdersUseMultiContext` orders can also be processed in multi context this can be used only after enabling "supportMultiContext". By using multi contexts the performance improvements can be seen in terms of throughput and decreased latency because the wait time will be reduced.

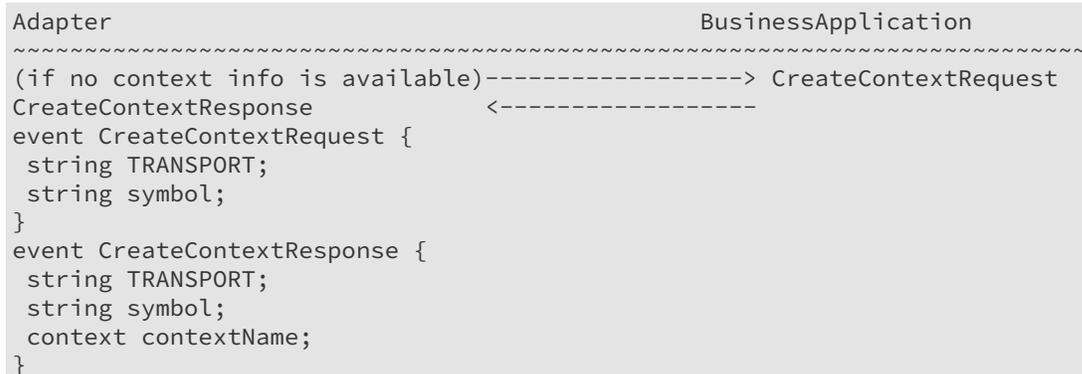
Context mapped to symbol can be done at startup or dynamically. At Startup only user (BA) can map symbol to context by `com.apama.mds.MDServerContextRequest` event. If the symbol is already mapped to one context, then adapter will reject that request and acknowledge to user with error message by `com.apama.mds.MDServerContextResponse` event.

If Symbol is not mapped to any context while handling subscriptions, then Adapter will send a request to user(BA) for create a new Context by `com.apama.mds.CreateContextRequest` event and expects response in `com.apama.mds.CreateContextResponse` event. If the user doesn't respond to request, then the subscription will be handled in the default context.

Work flow for context creation at startup



Work flow of content mapping dynamically



Transport events

The User is given the option of disconnecting from the exchange using following event:

```

event doDisconnect {
  string transportName;
  dictionary<string, string> extraParams;
}
To connect back to a Exchange the user have to send the event.
event doConnect {
  string transportName;
}

```

```
dictionary<string, string> extraParams;
}
```

Working of the subscription provider

Subscription Request. SubscribeDepth event is sent to the Business Application for requesting data (Quotes). The Application has to respond with MDServerDepth or a MarketDataRequestReject.

Message flow:

Business Application	Adapter
~~~~~	
<-----	
mds.MarketDataRequestReject	mds.SubscribeDepth
mds.MDServerDepth	MarketDataRequestReject
	----->

## Format of SubscribeDepth

Name	Description
serviceId	FXALL_TCPI
marketId	transportName
MDReqID	QuoteRequestID
symbol	Symbol

Extra parameter information of SubscribeDepth:

Name	Description
QuoteType	SPOT/SWAP/FORWARD
RequestKey	FxallID on which the orders will be placed.
customerId	
client	
OfferToDealType	
NumCompetitors	

Each leg has following values that will be added in the extra parameters of the SubscribeDepth with prefix Leg_1 and Leg_2 for swap and only leg_1 for SPOT and FORWARD:

Name	Description
Currency	Base currency of the leg.

Name	Description
Quantity	Quantity of the instrument quote.
Symbol	Symbol against which the quote is requested.
TenorValue	Tenor of the Leg
ValueDate	Value date of the leg

Each leg also has allocation's which are in the format:

Name	Description
Leg_n_Allocation_m	account name
Leg_n_AllocationQty_m	Quantity allocated for the account

where, n is the leg number and m is allocation number. Refer to sample events.

## Format of UnSubscribeDepth

Name	Description
serviceId	FXALL_TCPI
marketId	transportName
MDReqID	QuoteRequestId
symbol	Symbol

Refer to sample events.

## Format of MDServerDepth

MDServerDepth has a sequence of MDDepthEntry's, each giving leg information on askside or bidside. The ExtraParamameters of MDDepthEntry should contains the following keys:

Key	Value	Mandatory	Description
legside	1 or 2	yes	The leg number of the values.
Allin	float	no	AllinRate of quote of the corresponding leg.
FwdPts	float	no	Forward Points of the corresponding leg.

The extra parameter of the MDServerDepth should also contain "QuoteType" key with value SPOT/SWAP/FORWARD. (Refer to Sample events).

## Order management

Adapter places order using com.apama.oms.NewOrder event with Service ID "FXALL_TCPI". The Application has to Respond with an com.apama.oms.OrderUpdate. The adapter sends a DealResponseFromFXall to the provider only for a fill.

Message flow:

```

Business Application                                     Adapter
~~~~~
 <-----
oms.OrderUpdate ----->
 DealResponseFromFXall(only for fill)
event DealResponseFromFXall{
 string transportName;
 string orderId;
 boolean isTransactionCompleted;
 dictionary<integer,float> __timeStamps;
}

```

## Info Management

The server can send an Info Request on the status of a Deal. Adapter sends it to the Application as a TCPIInfoRequest and expects oms.OrderUpdate as a response.

Message flow:

```

Business Application Adapter
~~~~~
TCPIInfoRequest <----- InfoRequest
oms.OrderUpdate -----> InfoResponse
event TCPIInfoRequest{
  string transportName;
  string orderId;
  string symbol;
  string parameter;
  dictionary<integer,float> __timeStamps;
}
event InfoResponse{
  string transportName;
  string orderId;
  string symbol;
  boolean isAccepted;
  string rejectReason;
  dictionary<integer,float> __timestamps;
}

```

## Sample events

### SubscribeDepth

Example: SPOT

```
com.apama.mds.SubscribeDepth("FXALL_TCPI","FXALL_TCPI_TRANSPORT",
  "Quote_rq_1389267826126","EUR.USD",{},{},{ "InCompetition":"false",
  "Leg_1_AllocationEP_1":"{}",Leg_1_AllocationEP_2":"{}",
  "Leg_1_AllocationQty_1":"-1000","Leg_1_AllocationQty_2":"-1000",
  "Leg_1_Allocation_1":"ACCT1","Leg_1_Allocation_2":"ACCT2",
  "Leg_1_Currency":"EUR","Leg_1_LegEP":"{}","Leg_1_Quantity":"2000",
  "Leg_1_Symbol":"EUR.USD","Leg_1_TenorValue":"SPOT",
  "Leg_1_ValueDate":"20140113","MIC_CODE":"FXAL",
  "NumCompetitors":"0","OfferToDealType":"At Quote","QuoteType":"SPOT",
  "RequestKey":"17895250","__type":"QUOTE_REQUEST",
  "client":"user3@apamacust3","customerId":"apamacust3",
  "customerId":"false"})
```

Example: SWAP

```
com.apama.mds.SubscribeDepth("FXALL_TCPI","FXALL_TCPI_TRANSPORT",
  "Quote_rq_1389268042186","EUR.USD",{},{},{ "InCompetition":"false",
  "Leg_1_AllocationEP_1":"{}","Leg_1_AllocationQty_1":"1000",
  "Leg_1_Allocation_1":"ACCT1","Leg_1_Currency":"EUR",
  "Leg_1_LegEP":"{}","Leg_1_Quantity":"1000","Leg_1_Symbol":"EUR.USD",
  "Leg_1_TenorValue":"SPOT","Leg_1_ValueDate":"20140113",
  "Leg_2_AllocationEP_1":"{}","Leg_2_AllocationQty_1":"1000",
  "Leg_2_Allocation_1":"ACCT1","Leg_2_Currency":"EUR",
  "Leg_2_LegEP":"{}","Leg_2_Quantity":"1000","Leg_2_Symbol":"EUR.USD",
  "Leg_2_TenorValue":"1M","Leg_2_ValueDate":"20140213","MIC_CODE":"FXAL",
  "NumCompetitors":"0","OfferToDealType":"At Quote","QuoteType":"SWAP",
  "RequestKey":"17895259","__type":"QUOTE_REQUEST",
  "client":"user3@apamacust3","customerId":"apamacust3","isNDF":"false"})
```

### UnSubscribeDepth

Example:

```
com.apama.mds.UnsubscribeDepth("FXALL_TCPI","FXALL_TCPI_TRANSPORT",
  "Quote_rq_1389332769140","EUR.USD",{},{},{})
```

### MDServerDepth

Example: SPOT

```
com.apama.mds.MDServerDepth("FXALL_TCPI_TRANSPORT","Quote_rq_1387200864534","EUR.USD",
  [com.apama.mds.MDDepthEntry("", "", "0", 100, 1000,
    {"Allin":"2","FwdPts":"1","legside":"1"}),
  com.apama.mds.MDDepthEntry("", "", "1", 100.30000000000001, 1000,
    {"Allin":"2","FwdPts":"1","legside":"1"})],
  {}, {}, {"QuoteType":"SPOT"})
```

Example: SWAP

```
com.apama.mds.MDServerDepth("FXALL_TCPI_TRANSPORT","Quote_rq_1387200864534","EUR.USD",
[com.apama.mds.MDDepthEntry("", "", "0", 100, 1000,
  {"Allin": "2", "FwdPts": "1", "legside": "1"}),
com.apama.mds.MDDepthEntry("", "", "1", 100.300000000000001, 1000,
  {"Allin": "2", "FwdPts": "1", "legside": "1"}),
com.apama.mds.MDDepthEntry("", "", "0", 100, 1000,
  {"Allin": "2", "FwdPts": "1", "legside": "2"}),
com.apama.mds.MDDepthEntry("", "", "1", 100.300000000000001, 1000,
  {"Allin": "2", "FwdPts": "1", "legside": "2"})],
{},{}, {"QuoteType": "SWAP"})
```

## NewOrder

Example: SPOT

```
com.apama.oms.NewOrder("1", "EUR.USD", 100.100000000000001, "BUY",
"FOREX - PREVIOUSLY QUOTED", 1000, "FXALL_TCPI", "", "", "FXALL_TCPI_TRANSPORT", "",
"APAMATest", {"Leg_1_AllocationQty_1": "-1000", "Leg_1_AllocationQty_2": "-1000",
"Leg_1_Allocation_1": "ACCT1", "Leg_1_Allocation_2": "ACCT2", "Leg_1_Currency": "EUR.USD",
"Leg_1_ForwardPoints": "0", "Leg_1_FutSettDate": "06/01/2014",
"Leg_1_LastSpotRate": "100.100000000000001", "Leg_1_OrderQty": "2000",
"Leg_1_OutrightRate": "100.100000000000001", "Leg_1_Side": "BUY", "Leg_1_TenorValue": "SPOT",
"OrderType": "SPOT", "RequestKey": "1", "TradeDate": "20140106",
"TransactTime": "20140106-19:17:56.063", "__type": "DEAL_REQUEST",
"customerName": "APAMA", "serviceId": "FXALL_TCPI"})
```

Example: SWAP

```
com.apama.oms.NewOrder("1", "EUR.USD", 100.100000000000001, "BUY",
"FOREX - PREVIOUSLY QUOTED", 1000, "FXALL_TCPI", "", "", "FXALL_TCPI_TRANSPORT", "",
"APAMATest", {"Leg_1_AllocationQty_1": "1000", "Leg_1_Allocation_1": "ACCT1",
"Leg_1_Currency": "EUR.USD", "Leg_1_ForwardPoints": "0", "Leg_1_FutSettDate": "06/01/2014",
"Leg_1_LastSpotRate": "100.100000000000001", "Leg_1_OrderQty": "1000",
"Leg_1_OutrightRate": "100.100000000000001", "Leg_1_Side": "BUY", "Leg_1_TenorValue": "SPOT",
"Leg_2_Currency": "EUR.USD", "Leg_2_ForwardPoints": "0", "Leg_2_FutSettDate": "06/01/2014",
"Leg_2_LastSpotRate": "100.100000000000001", "Leg_2_OrderQty": "1000",
"Leg_2_OutrightRate": "100.100000000000001", "Leg_2_Side": "BUY", "Leg_2_TenorValue": "SPOT",
"OrderType": "SWAP", "RequestKey": "1", "TradeDate": "20140106",
"TransactTime": "20140106-19:17:56.063", "__type": "DEAL_REQUEST",
"customerName": "APAMA", "serviceId": "FXALL_TCPI"})
```

## Reject Codes for Deals and QuoteRequests

### Reject Codes for Quotes

This has to be sent using `com.apama.mds.MarketDataRequestReject` with `MDReqRejReason` having one of the following code. If any other non supported code is sent it will be mapped to 3009.

Code	Description
3001	This user does not have permission to receive quotes.

Code	Description
3002	The specified customer account does not have permissions to receive quotes.
3003	The requested currency pair is inverted from standard and not supported.
3004	The specific currency pair is not supported.
3005	Client credit check failed.
3006	Tenor not supported for this currency pair.
3007	Value date not recognised as valid business date.
3008	Trade type not supported (e.g. application does not support swaps).
3009	Provider cannot price - use this code for any technical problems.
3010	Minimum trade amount not met.
3014	Provider does not support multi-allocation (e.g. block) trades.

## Reject Codes for Deals

A Deal Reject has to be sent using `com.apama.oms.OrderUpdate` event with the extraparams should contain the key "RejectCode" with value one of the following:

Code	Description
4007	Quote has expired - either superceded or timed out.
4008	Decimal places on rates unsuitable.
4009	Trade amount not valid.
4010	Credit limit checks failed.
4011	Generic rejection code - use for miscellaneous failures.

## Status reporting events

Status reporting uses the standard `com.apama.statusreport` event interface, defined in the `StatusSupport.mon` file. Heartbeats and status messages between the service monitors and the transport/codecs are managed by the `IAFStatusManager.mon` service monitor.

Applications should subscribe to status by sending the following event to the correlator:

```
com.apama.statusreport.SubscribeStatus("FXALL_TCPI", <object>, "", <transportname>)
```

Where <transportname> is the transport name as used in the SessionConfiguration event, and <object> is one of the following:

#### Adapter

subscriptions where all fields are "wildcards" are also supported:

```
com.apama.statusreport.SubscribeStatus("", "", "", "")
```

In this case the subscriber will receive all status reports from the FXALL_TCPI adapter as well as every other status-reporting entity in the system.

A `com.apama.statusreport.UnsubscribeStatus` event with the same field values can be used to cancel a previous status subscription.

Status reports are routed as `com.apama.statusreport.Status` objects, with the "serviceId", "object" and "connection" fields set as above for subscriptions. The reported status is as follows:

## Adapter object

This object reports on the overall state of the connection between the FXall server, transport and service monitors. If the "available" flag is true, the adapter is "up", in contact with the server and ready to handle requests. The value "Connected" will appear in the "summaries" field in this case. Otherwise, the adapter is unavailable and clients should wait until it becomes available before trying to use it.

# 18 FXCMPPro FIX Adapter

---

■ Prerequisites .....	246
■ IAF configuration .....	246
■ Service monitor injection order .....	246
■ Session configuration .....	246
■ Sample events .....	247
■ Password Management .....	247

This file describes the steps that must be undertaken in addition to standard FIX configuration when connecting to the FXCMPro system. These notes also apply when connecting to FXCMPro via FXCMPro.

## Prerequisites

---

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

## IAF configuration

---

FXCMPro uses a custom data dictionary (FIX42-FXCMPro.xml). This file must be referenced by the QuickFIX.DataDictionary transport property. For example:

```
<property name="QuickFIX.DataDictionary"
  value="{APAMA_HOME}/adapters/config/FIX42-FXCMPro.xml"/>
In addition, the following properties are required for FXCMPro connections:
<!-- Password -->
<property name="StaticField.body.A.554" value="<FXCMPro_PASSWORD>"/>
```

Where <FXCMPro_PASSWORD> is the password supplied to you by FXCMPro.

The supplied sample configuration file (FIX-FXCMPro.xml.dist) contains placeholders for the necessary configuration parameters. This configuration file defines two connections (FXCMPro_TRADING and FXCMPro_MARKET_DATA) using the event channel FXCMPro_FIX for communication with the correlator.

## Service monitor injection order

---

Inject the monitors listed in [“FIX order management session” on page 82](#) and [“FIX Legacy Market data session” on page 83](#), followed by:

- {APAMA_HOME}/adapters/monitors/FIX_FXCMPro_Support.mon

## Session configuration

---

FXCMPro sessions should be configured with the following configuration parameters:

Trading sessions:

```
OrderManager.RequireSessionsStatusOpen = true
OrderManager.IgnoreStatusOnOrderCancelReject = true
OrderManager.HandleSuspendedExecType = true
```

Market data sessions:

```
SubscriptionManager.RequireSessionStatusOpen = true
SubscriptionManager.Field0missions = Currency
```

Sample session configuration events, compatible with the distributed configuration file, are shown below:

```
// FXCMPRO trading session
com.apama.fix.SessionConfiguration("FXCMPRO_TRADING",
  {"FixChannel":"FXCMPRO_FIX",
   "OrderManager.RequireSessionStatusOpen":"true",
   "OrderManager.IgnoreStatusOnOrderCancelReject":"true",
   "OrderManager.HandleSuspendedExecType":"true"})
// FXCMPRO market data session
com.apama.fix.SessionConfiguration("FXCMPRO_MARKET_DATA",
  {"FixChannel":"FXCMPRO_FIX",
   "SubscriptionManager.RequireSessionStatusOpen":"true",
   "SubscriptionManager.FieldOmissions":"Currency"})
```

## Sample events

### Subscription

```
com.apama.marketdata.SubscribeDepth("FIX", "FXCMPRO_MARKET_DATA",
  "EUR/GBP",{"264":"0","266":"N","7560":"Y"})
```

### Order Placing

```
com.apama.oms.NewOrder("order1", "EUR/USD",1.27378,"BUY",
  "FOREX LIMIT",500000,"FIX","","", "FXCMPRO_TRADING", "", "", {"15":"EUR","59":"0"})
```

ICEBERG Order with hidden quantity. Tag 210 indicates quantity displayed, The Quantity field of NewOrder holds the total quantity including the hidden amount.

```
com.apama.oms.NewOrder("order1", "EUR/USD",1.27378,"BUY",
  "FOREX ICEBERG",500000,"FIX","","", "FXCMPRO_TRADING", "", "",
  {"15":"EUR","59":"0","210":"200000"})
```

## Password Management

The supplied FIX_FXCMPRO_Support.mon service monitor can be used to reset the FXCMPRO session password. This monitor should be injected after all of the standard service monitors, but before any FXCMPRO sessions are configured.

To reset the password for a FXCMPRO session, sent the following event to the correlator:

```
com.apama.fix.fxcmpro.PasswordChangeRequest(<transport>, <requestId>,
  <userId>, <oldPassword>, <newPassword>)
```

Where:

<transport> is the session/transport name (e.g. "FXCMPRO_TRADING")

<requestId> is a user-supplied key to identify replies to this request

<userId> is the SenderCompId for this session

<oldPassword> is the existing password for the session

<newPassword> is the desired new password for the session

Each request should result in a `com.apama.fix.fxcmpro.PasswordChangeResponse()` event being routed and logged by the monitor:

```
com.apama.fix.fxcmpro.PasswordChangeResponse(<transport>,  
    <requestId>, <userId>, <success>, <status>, <message>)
```

Where:

<transport> is the session/transport name

<requestId> is the user-supplied identifier from the request

<userId> is the SenderCompId from the request

<success> is a boolean value indicating whether the password was changed

<status> is one of the `RESPONSE_*` constants listed in the `FIX_FXCMPRO_Support` monitor. Typical values are `RESPONSE_PASSWORD_CHANGED ("5")` or `RESPONSE_USER_NOT_RECOGNISED ("3")` which FXCMPro uses as a catch-all failure code.

<message> is any error message returned by the server

A password reset can be performed at any time after the session has been successfully logged on and a Trading Session Status of "Open" has been sent (this will be logged by the FIX Session Manager). A password reset is *required* if the server sends a Trading Session Status of "Halted" immediately after a successful logon (this will be logged at WARN level by the FIX FXCMPro Password Manager). Note that you cannot successfully log on with an expired password, so it is advised to perform a password reset as soon as the "Halted" session status is seen. The session will be locked after too many login attempts with an expired password, and can only be unlocked by calling FXCMPro technical support.

# 19 Goldman Sachs FX FIX Adapter

---

■ Prerequisites .....	250
■ IAF configuration .....	250
■ Service monitor extensions .....	250
■ Service monitor injection order .....	251
■ Service ID configuration .....	251
■ Session configuration .....	251
■ Session configuration parameters .....	251
■ GS Requirements or Recommendations .....	252
■ Quotes subscription extra parameters .....	252
■ Orders (Trade Requests) extra parameters .....	252
■ Goldman Sachs subscriptions and unsubscriptions (Quote) .....	252
■ Sending orders using OMS interface NewOrder(NewOrderSingle) .....	253

The Goldman Sachs FX system deviates from standard FIX 4.4 in a number of ways which have made it necessary to provide a number of extensions to the FIX adapter. The extensions are into two areas: IAF configuration and service monitors.

## Prerequisites

---

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

## IAF configuration

---

When acting as a client to Goldman Sachs FX, the IAF should be configured with the Goldman Sachs FX specific configuration file, the distribution version of which can be found here:

```
${APAMA_HOME}/adapters/config/FIX-GS.xml.dist
```

Note that the requirement for a GS-specific configuration file means that an IAF using this configuration file must be used for GS FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for GS operation. If you need to connect to other FIX servers as well as GS, the non-GS sessions should be configured in a separate IAF instance. However, GS and non-GS session can co-exist safely within the same correlator instance.

As well as the standard host, port, sender and target CompID the following parameters must be supplied for GS session:

- For market data session:

```
<property name="StaticField.body.A.553" value="Username" />
<property name="StaticField.body.A.554" value="Password" />
```

- For trading session:

```
<property name="StaticField.body.A.553" value="Username" />
<property name="StaticField.body.A.554" value="Password" />
<property name="StaticField.body.D.453" value="NoPartyIDs" />
<property name="StaticField.body.D.448" value="PartyID" />
<property name="StaticField.body.D.447" value="PartyIDSource" />
<property name="StaticField.body.D.452" value="PartyRole" />
```

## Service monitor extensions

---

The file `FIX_GS_Support.mon` provides Quote request message handling. GS required a special handling for sending QuoteRequest/QuoteCancel. Therefore, the following EPL files must be injected into the correlator, after the standard FIX adapter service monitors:

```
FIX_GS_Support.mon
```

## Service monitor injection order

Inject the monitors listed in “[FIX order management session](#)” on page 82 and “[FIX Legacy Market data session](#)” on page 83, followed by:

- `${APAMA_HOME}/adapters/monitors/FIX_GS_Support.mon`

## Service ID configuration

The Service Id to be used is "GS-FIX".

## Session configuration

### Market data session

```
com.apama.fix.SessionConfiguration("GS_MARKET_DATA", {"FixChannel":"GS_FIX",
"SERVICEID":"GS-FIX","SubscriptionManager.IncludeTimeInRequestID":"true",
"SubscriptionManager.SendQuoteAck":"true",
"SubscriptionManager.EnableMassQuoteAcknowledgement":"true"})
```

### Trading session

```
com.apama.fix.SessionConfiguration("GS_TRADING", {"FixChannel":"GS_FIX",
"SERVICEID":"GS-FIX", "OrderManager.GenerateNewStyleOrderIds":"true",
"SubscriptionManager.EnableMassQuoteAcknowledgement":"true"})
```

## Session configuration parameters

Parameter	Description
SERVICEID	Service ID to be used is "GS-FIX". This parameters configures the GS specific monitors to handle GS specific events or special handling of the events.
SubscriptionManager. SendQuoteAck	Whether to send the Quote acknowledgements on receiving of Quotes.
OrderManager. GenerateNewStyleOrderIds	Generates the id with the complete current (full length) time for uniqueness.
SubscriptionManager. IncludeTimeInRequestID	Whether to include timestamp in the requestID for QuoteRequest.
SubscriptionManager. SuppressZeroQuantities	Enable to suppress the zero quantities in the com.apama.marketdata.Depth event. The Depth event will not publish those entries which has zeros quantities.
SubscriptionManager.	Enable MassQuoteAcknowledgement.

Parameter	Description
EnableMassQuoteAcknowledgement	

## GS Requirements or Recommendations

- **Email Alerts:** In the case where a Trade Acknowledgement and accompanying Trade Fill or Reject message are not received within 5 seconds, Goldman Sachs expects the client to contact them immediately using the following support email address: efx@gs.com.

This support inbox is manned 24 hrs a day 5.5 days a week, providing global support at all hours. The email sent should contain adequate information to identify the trade.

- **Reconnect Interval:** GS recommends that the adapter reconnect interval is set between 15-20 seconds.

## Quotes subscription extra parameters

Parameter	Value
SPOT	Quote=Y
FORWARD	Quote=Y;FutSettDate=YYYYMMDD
NDF	Quote=Y;6363= (tenor ie. SN,B1,B3, W1, W2, M1, Y1, IMM1, IMM2)

## Orders (Trade Requests) extra parameters

Parameter	Value
SPOT	QuoteID=xxxxxxxxxxxxxxxxxx;Currency=XXX

For Market Orders specify Price as 0.0 and don't specify QuoteID.

## Goldman Sachs subscriptions and unsubscriptions (Quote)

### Sample subscriptions

- **SPOT Quote:**

```
com.apama.marketdata.SubscribeDepth("GS-FIX", "GS_MARKET_DATA",
  "EUR/GBP",{"Quote":"Y", "15":"EUR", "OrderQty":"100"})
```

- **FORWARD Quote:**

```
com.apama.marketdata.SubscribeDepth("GS-FIX", "GS_MARKET_DATA",
  "EUR/GBP",{"Quote":"Y", "15":"EUR", "OrderQty":"100",
```

```
"FutSettDate":"20091027"}})
```

#### ■ NDF Quote:

```
com.apama.marketdata.SubscribeDepth("GS-FIX", "GS_MARKET_DATA",
  "EUR/USD", {"Quote":"Y", "15":"EUR", "OrderQty":"100", "6363":"B1"})
```

### Sample Unsubscriptions

```
com.apama.marketdata.UnsubscribeDepth("GS-FIX", "GS_MARKET_DATA",
  "EUR/GBP", {"Quote":"Y", "15":"EUR", "OrderQty":"100"})
```

## Sending orders using OMS interface

### NewOrder(NewOrderSingle)

Specify the following parameter values:

```
BID_QuoteID/ASK_QuoteID (Tag 117) :- Supply the QuoteID
  Currency (Tag 15) :- Supply the currency used for price.
```

Sample new order:

```
com.apama.oms.NewOrder("order1", "EUR/GBP",0.0, "BUY", "MARKET", 100,
  "GS-FIX", "", "", "GS_TRADING", "", "", {"15":"EUR","117":"2.98295001288E11"})
```

#### Note:

For Market Orders specify Price as 0.0 and don't specify QuoteID For Placing new orders  
 BID_QuoteID/ASK_QuoteID(tag 117) may be required and can be picked from  
 com.apama.marketdata.Depth() event.



# 20 HotSpot FX FIX Adapter

---

■ Prerequisites .....	256
■ IAF configuration .....	256
■ HotSpot trading session configuration .....	256
■ Service monitor injection order .....	256
■ MarketData subscriptions .....	257
■ Order management .....	257

This chapter describes the design, implementation and usage of the FIX HotSpot extensions.

## Prerequisites

---

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

## IAF configuration

---

When acting as a client to HotSpot, the IAF should be configured with the HotSpot specific configuration file, the distribution version of which can be found here:

```
${APAMA_HOME}/adapters/config/FIX-HotSpot.xml.dist
```

HotSpot uses a custom data dictionary . This file must be referenced by the "QuickFIX.DataDictionary" transport property. For example, for HotSpot FIX platform:

```
<property name="QuickFIX.DataDictionary"  
value="${APAMA_HOME}/adapters/config/FIX42-HotSpot.xml"/>
```

The IAF configuration file should contain appropriate host, port, sender and target CompID parameters.

For the MDA Mode the path of the correct adapter configuration file should also be provided using the property, for HotSpot FIX platform:

```
<property name="AdapterConfiguration"  
value="${APAMA_HOME}/adapters/config/FIXtransport-BaseFIXLibrary-HotSpot.xml" />
```

The supplied sample configuration file (FIX-HotSpot.xml.dist) contains placeholders for the necessary configuration parameters. This configuration file defines two connections (HOTSPOT_FIX_MARKET_TRANSPORT and HOTSPOT_FIX_TRADING_TRANSPORT) using the event channel "HOTSPOT_FIX" for communication with the correlator.

## HotSpot trading session configuration

---

```
com.apama.fix.SessionConfiguration("HOTSPOT_TRADING",  
{"FixChannel":"HOTSPOT_FIX",  
"OrderManager.IgnoreStatusOnOrderCancelReject":"true",  
"OrderManager.GenerateNewStyleOrderIds":"true",  
"OrderManager.updateKeyParamsOnStateChange":"true"})
```

## Service monitor injection order

---

Inject the monitors to connect to MDA Session. See [“FIX MDA session” on page 82](#) and [“FIX order management session” on page 82](#).

## MarketData subscriptions

The MDA adapter configuration file 'FIXTransport-BaseFIXLibrary-HotSpot.xml' contains properties which are used for Sending subscription and unsubscription Requests. ServiceName to be used when starting the Market data session is FIX .

### Sample subscription:

```
...
// Create and connect the session handler
SessionHandler sessionHandler := (new SessionHandlerFactory).connect(mainContext,
    "FIX", "HOTSPOT_MARKETDATA_TRANSPORT");
// Create the trade subscriber factory and specify the session
// handler plus a symbol to subscribe
DepthSubscriber subscriber := (new DepthSubscriberFactory).subscribeCb(sessionHandler,
    "EUR/USD", onAllDepth);
...

action onAllDepth(com.apama.md.client.CurrentDepthInterface depth) {
    ...
}
```

## Order management

Supported Order types:1.MARKET, 2.LIMIT and 3.PEGGED

### Examples:

```
//New Order
com.apama.oms.NewOrder("12","EUR/USD",1.13765,"SELL","LIMIT",200000,
"FIX","","","HOTSPOT_TRADING","","",{ "59":"3"})
```

```
//Place New Order with MaxShow
MinQty com.apama.oms.NewOrder("12","EUR/USD",1.14048,"SELL","LIMIT",10000000,
"FIX","","","HOTSPOT_TRADING","","",{
"59":"0","15":"EUR","210":"30000000","110":"20000000"})
```

```
//Place Pegged Order
com.apama.oms.NewOrder("1","EUR/USD",1.14048,"BUY","PEGGED",10000000,
"FIX","","","HOTSPOT_TRADING","","",{
"59":"0","15":"EUR","210":"30000000","110":"20000000","211":"0.003","18":"R"})
```

```
//Cancel Order
com.apama.oms.CancelOrder("12", "FIX", { "59":"0" })
```

```
//Amend Order
com.apama.oms.AmendOrder("12","FIX","EUR/USD",1.13716,"BUY","LIMIT",800000,{ "59":"0" })
```

### Note:

No Support for Mass Order Cancel and Order Status. When the order gets replaced, OrdStatus is 0 in extraparams of com.apama.oms.OrderUpdate.



# 21 HSBC FX FIX Adapter

---

■ Prerequisites .....	260
■ IAF configuration .....	260
■ Service monitor injection order .....	260
■ Service ID configuration .....	260
■ Session configuration .....	260
■ Quote subscription or unsubscription .....	261
■ Order placing .....	261

This file describes the design, implementation and usage of the FIX HSBC extensions.

## Prerequisites

---

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

## IAF configuration

---

When acting as a client to HSBC FXtransact, the IAF should be configured with the HSBC specific configuration file, the distribution version of which can be found here:

```
${APAMA_HOME}/adapters/config/FIX-HSBC.xml.dist
```

Note that the requirement for a HSBC-specific configuration file means that an IAF using this configuration file must be used for HSBC FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for HSBC operation. If you need to connect to other FIX servers as well as HSBC, the non-HSBC sessions should be configured in a separate IAF instance. However, HSBC and non-HSBC session can co-exist safely within the same correlator instance.

HSBC provided host, port, sender and target CompID parameters must be supplied in the IAF parameter file.

## Service monitor injection order

---

Inject the monitors listed in [“FIX order management session” on page 82](#) and [“FIX Legacy Market data session” on page 83](#), followed by:

- `${APAMA_HOME}/adapters/monitors/FIX_HSBC_Support.mon`

## Service ID configuration

---

The Service Id to be used is "HSBC-FIX".

## Session configuration

---

The `com.apama.fix.SessionConfiguration` event has the following fields:

```
SubscriptionManager.UnsubscribeOnQuoteCancel = true
```

If "true", all subscriptions are removed when a QuoteCancel message is received from the server with `QuoteID(tag 117) = "*" .`

The default is false.

For example:

```
com.apama.fix.SessionConfiguration("HSBC-TEST",{
  "FixChannel":"HSBC_FIX",
  "SERVICEID":"HSBC-FIX",
  "SubscriptionManager.UnsubscribeOnQuoteCancel":"true"})
```

## Quote subscription or unsubscription

### > For Volume for Price(VFP):

```
com.apama.marketdata.SubscribeDepth("HSBC-FIX", "HSBC-TEST", "EUR/JPY",
  {"Quote":"Y"})
com.apama.marketdata.UnsubscribeDepth("HSBC-FIX", "HSBC-TEST", "EUR/JPY",
  {"Quote":"Y"})
```

### > For Price for Volume(PFV):

Here are volume bands and mentioned using the parameter HSBC.PFVbands.

```
com.apama.marketdata.SubscribeDepth("HSBC-FIX", "HSBC-TEST", "EUR/JPY",
  {"HSBC.PFVbands":"1000000,3000000,7000000", "Quote":"Y"})
com.apama.marketdata.UnsubscribeDepth("HSBC-FIX", "HSBC-TEST", "EUR/JPY",
  {"HSBC.PFVbands":"1000000,3000000,7000000", "Quote":"Y"})
```

## Order placing

Order should be place based on the latest depth received:

```
com.apama.marketdata.Depth("EUR/JPY", [103.084,103.083,103.082],
  [103.105,103.106,103.108], [103.09450000000001,103.0945,103.095],
  [1000000,3000000,7000000], [1000000,3000000,7000000],
  {"35":"W", "52":"20120213-08:46:16.370", "9063":"SP", "ASK1_9063":"SP",
  "ASK1_MDEntryDate":"20120215",
  "ASK1_QuoteEntryID":"1329122776023.1157442.P-1M",
  "ASK2_9063":"SP", "ASK2_MDEntryDate":"20120215",
  "ASK2_QuoteEntryID":"1329122776023.1157442.P-3M",
  "ASK3_9063":"SP", "ASK3_MDEntryDate":"20120215",
  "ASK3_QuoteEntryID":"1329122776023.1157442.P-7M",
  "BID1_9063":"SP", "BID1_MDEntryDate":"20120215",
  "BID1_QuoteEntryID":"1329122776023.1157442.P-1M",
  "BID2_9063":"SP", "BID2_MDEntryDate":"20120215",
  "BID2_QuoteEntryID":"1329122776023.1157442.P-3M",
  "BID3_9063":"SP", "BID3_MDEntryDate":"20120215",
  "BID3_QuoteEntryID":"1329122776023.1157442.P-7M",
  "HSBC.PFVbands":"1000000,3000000,7000000",
  "Market":"HSBC-TEST", "Quote":"Y",
  "SERVICE_NAME":"HSBC-FIX"})
```

From the depth the following things need to be chosen:

- Price: based on order qty the price corresponding volume band should be chosen.
- QuoteId: Quote Id corresponding to the price must be chosen. It is of the form "ASK2_QuoteEntryID" (where 2 signifies the 2nd volume band).

For example,

```
com.apama.oms.NewOrder("HSBC-TEST:1","EUR/JPY",103.112,"BUY",  
"FOREX - PREVIOUSLY QUOTED",1000000,"HSBC-FIX","", "",  
"HSBC-TEST","", "", {"117": "1329122775918.1157412.P-1M",  
"15": "EUR", "59": "4", "75": "20120202", "9063": "SP"})
```

### > Order with party information

Parties info should be provided in extraparams as: "Party0": "APAMA,D,3". Here  
PartyID(448)=APAMA, PartyIDSource(447)=D, PartyRole(452)="3"

For example,

```
com.apama.oms.NewOrder("HSBC-TEST:2","EUR/JPY",103.126,"BUY",  
"FOREX - PREVIOUSLY QUOTED",1000000,"HSBC-FIX","", "",  
"HSBC-TEST","", "", {"117": "1329123422020.1167519.P-1M",  
"15": "EUR", "59": "4", "75": "20120202",  
"9063": "SP", "Party0": "APAMA,D,3"})
```

# 22 JPMC FX FIX Adapter

---

■ Prerequisites .....	264
■ IAF configuration .....	264
■ Service monitor extensions .....	264
■ Service monitor injection order .....	264
■ Service ID configuration .....	265
■ Session configuration .....	265
■ JPMC requirements/recommendations .....	265
■ Market Data subscription EXTRA PARAMS .....	266
■ JPMorgan order management .....	267

The JP Morgan FX system deviates from standard FIX 4.2 in a number of ways which have made it necessary to provide a number of extensions to the FIX adapter. The extensions are into two areas: IAF configuration and service monitors.

## Prerequisites

---

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

## IAF configuration

---

When acting as a client to JP Morgan FX, the IAF should be configured with the JP Morgan FX specific configuration file, the distribution version of which can be found here:

```
${APAMA_HOME}/adapters/config/FIX-JPMC.xml.dist
```

Note that the requirement for a JPMC-specific configuration file means that an IAF using this configuration file must be used for JPMC FX FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for JPMC operation. If you need to connect to other FIX servers as well as JPMC, the non-JPMC sessions should be configured in a separate IAF instance. However, JPMC and non-JPMC session can co-exist safely within the same correlator instance.

As well as the standard host, port, sender and target CompID the following parameters must be supplied for JPMC session:

For market data session

```
<property name="StaticField.body.A.96" value="JPMC Password" />  
<property name="StaticField.body.A.95" value="JPMC Password length" />
```

For trading session

```
<property name="StaticField.body.A.96" value="JPMC Password" />  
<property name="StaticField.body.A.95" value="JPMC Password length" />
```

## Service monitor extensions

---

The file `FIX_JPMC_Support.mon` provides market data update message handling for JPMC.

Therefore, the following EPL files must be injected into the correlator, after the standard FIX adapter service monitors:

```
FIX_JPMC_Support.mon
```

## Service monitor injection order

---

Inject the monitors listed in [“FIX order management session” on page 82](#) and [“FIX Legacy Market data session” on page 83](#), followed by:

■ `#{APAMA_HOME}/adapters/monitors/FIX_JPMC_Support.mon`

## Service ID configuration

The Service Id for market data session should "JPMC-FIX".

The Service Id for trading session should be "FIX".

## Session configuration

Market data session

```
com.apama.fix.SessionConfiguration("JPMC_MARKET_DATA", {"SERVICEID":"JPMC-FIX",
  "SubscriptionManager.RepeatingGroupTags":"[146 55 107 64 9002]"})
```

### Trading session

```
com.apama.fix.SessionConfiguration("JPMC_TRADING", {})
```

### Notes: SessionConfig parameters

Parameters name	Description
SERVICEID	Service ID to be used is JPMC-FIX. This parameters configures the JPMC specific monitors to handle JPMC specific events or special handling of the events.

## JPMC requirements/recommendations

### SSL connection

JPMorgan FX eTrading Platform requires clients to be establish connections to JPMorgan server using SSL. Once the connection is established, the client should verify the identity of the server using the SSL certificate and then initiate the normal FIX logon process.

The SSL certificate will be provided by JPMorgan and the clients can use any application which provides secure tunneling to establish the connection. A good opensource example for such an application is stunnel.

### Order management

Currently only limit orders and market orders are supported with immediate execution and without partial fills (that is, Fill or Kill)

## Market Data subscription EXTRA PARAMS

### Streaming subscription

9001	Quantity Requested. If present, only 1 layer will be streamed.
107	SecurityDesc -> The stream type the request is for. supported types: SPOT, OUTRIGHT. (if not supplied, SPOT is assumed)
64	FutSettDate -> Required if SecurityDesc=OUTRIGHT and Tenor is not specified.
9002	Tenor -> Required if SecurityDesc=OUTRIGHT and FutSettDate is not specified.

### JPMorgan subscriptions/unsubscriptions (streaming subscription)

#### Sample subscriptions

##### SPOT subscription

```
com.apama.marketdata.SubscribeDepth("JPMC-FIX", "JPMC_MARKET_DATA", "EUR/USD", {"107": "SPOT"})
```

##### Subscription with value date

```
com.apama.marketdata.SubscribeDepth("JPMC-FIX", "JPMC_MARKET_DATA", "USD/CAD", {"9002": "T+1"})
```

##### Subscription for different price bands

```
com.apama.marketdata.SubscribeDepth("JPMC-FIX", "JPMC_MARKET_DATA", "EUR/USD", {"9001": "1000000"})
com.apama.marketdata.SubscribeDepth("JPMC-FIX", "JPMC_MARKET_DATA", "EUR/USD", {"9001": "5000000"})
```

##### Sample unsubscription

```
com.apama.marketdata.UnsubscribeDepth("JPMC-FIX", "JPMC_MARKET_DATA", "EUR/USD", {"107": "SPOT"})
```

### Orders (Trade Requests) EXTRA PARAMS:

Account	The identity of the settlement account
107	SecurityDesc -> The product type the order is for supported types: SPOR, OUTRIGHT, SWAP. (if not supplied, SPOT is assumed)
21	HandlInst -> 1 = Automated execution order, private, no Broker Intervention
167	SecurityType -> FOR = Foreign Exchange Contract
40	OrderType -> Only supported types are "FOREX LIMIT" and "FOREX MARKET"
15	Currency -> The dealt currency (ISO code)

---

59                      TimeInForce -> 4 = Fill or Kill (FOK)

---

#### Other constraints

44                      Price -> In the case of Market Orders (40 = C (FOREX MARKET)), Price should be set to 0

---

140                     PrevClosePx -> Required when SecurityDesc is OUTRIGHT or SWAP. The spot component of the order For Market Orders, this should be set to 0.

---

64                      FutSettDate -> Required if SecurityDesc = OUTRIGHT or SWAP.

---

## JPMorgan order management

---

### Sending orders using OMS interface NewOrder(NewOrderSingle)

Currency (Tag 15) :- Supply the currency used for price.

#### Sample new order

```
com.apama.oms.NewOrder("1", "EUR/USD", 0, "BUY", "FOREX MARKET", 1000,
  "FIX", "", "", "JPMC_TRADING", "", "", {"21":"1","167":"FOR",
  "TimeInForce":"4","Account":"ABCDEFGH","140":"0",
  "FutSettDate":"2011-06-13", "15":"EUR"})
```



# 23 kdb+ Adapter

---

- Prerequisites ..... 270
- Features Supported ..... 270
- Configuration ..... 270
- Service monitors and injection order ..... 272
- Starting the Adapter ..... 272
- Update API call ..... 274

This adapter provides support for the KX Systems kdb+ database. kdb+ is a relational database and architecture that has proven to deliver the requirements for the real time business in capital markets. It provides a unified approach to data to enable applications for real-time performance even in high volumes of data. So, wherever speed has been the essence of competitive advantage, including algorithmic trading, cross-asset trading, program trading, real-time risk management, real-time order book, pre and post trade risk, rapid back testing of trading strategies and compliance reporting, kdb+ has come out as a primary name.

To develop CEP applications on kdb+, this adapter provides connectivity between kdb+ and Apama correlator. It opens the database connection statically from the parameter file and processes the upstream requests/queries from the correlator.

kdb+ provides a Java implementation of the API for connecting and running various database requests. This adapter uses this API and provides interfacing between kdb+ requests and the Apama's Normalised events.

## Prerequisites

---

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

## Features Supported

---

Supported:

- Query
- Upsert
- Execute
- Delete

Unsupported:

- Tickerplant subscription

## Configuration

---

The kdbplus transport provides the following configuration properties:

Name	Description
dbProxy	Database proxy type, meaning type of database: file db or real kdb. Values could be FILE_DB, KDB_STUB, KDB_REAL. By default it would be KDB_REAL.
dbName	Logical name of the database. It should be specified.

Name	Description
ipaddr	IP address of the database. It should be specified.
port	Port to be used to connect to the database. Should be specified if db is listening on a port otherwise connection would fail
poolSize	Connection Pool size to be maintained in the adapter. Its kind of indication that how many queries or requests could be run in parallel with the database connection. By default it is 8.
dateFormat	date format to be used for queries or responses for the database. By default "Simple format" would be used. e.g. dd/MM/yyyy
timeFormat	time format for the database connection. By default Simple format would be used. e.g. H:m
dateTimeFormat	date & time format for the database connection. By default Simple format would be used.e.g. dd/MM/yyyy H:m
timeZone	time zone to be used for the database connection. Otherwise a default timezone would be used. e.g. GMT.
nThreads	Numbers of threads to be used by the Executor service to process upstream requests from the correlator. By default it would be 5.
decoderName	Decoder to be used to encode or decode messages between IAF & correlator. Currently it should be only JNullCodec.jar.
entityDefinition	It is an optional property for defining database related entity definitions.
updateAPICall	Update API call to be used for update/insert statements. By default it is ".u.upd".
logAllUpstreamMessages	To print all upstream messages at INFO level in the IAF log. By default it is false.
logAllDownstreamMessages	To print all downstream messages at INFO level in the IAF log. By default it is false.
logAPIRelatedMessages	To print all kdb+ query related information before and after the call to the database at INFO level. By default it is false .
logAPIRelatedMessagesInDetail	To print detailed information kdb+ query related information before and after the call to the database at INFO level. By default it is false.

## Service monitors and injection order

The following standard service monitors should be injected before any kdbplus adapter service monitors are injected:

```

${APAMA_HOME}/monitors/StatusSupport.mon
${APAMA_HOME}/adapters/monitors/IAFStatusManager.mon
${APAMA_HOME}/monitors/data_storage/MemoryStore.mon

```

The following service monitors are supplied with the kdbplus adapter and should be injected in this order:

Name	Description
DatabaseSupport.mon	Provides event definitions for Database blocks and services.
kdbplus_AdapterEvents.mon	Defines the events that are required
kdbplus_DatabaseSupport.mon	Extends the standard <code>com.apama.db</code> interface with KDB Execute and ExecuteResult events
kdbplus_ConfigEvents.mon	Session configuration Event
kdbplus_IAFEvents.mon	IAF related events
kdbplus_StatusPublisher.mon	Status Publisher
kdbplus_SessionManager.mon	Session Manager
kdbplus_BlockInterface.mon	Exposes the basic storage/retrieval functionality via the <code>com.apama.db.*</code> and <code>com.apamax.db.*</code> event interfaces.

## Starting the Adapter

Adapter can be started by following the standard steps and specifying the parameters in the configuration xml. SessionConfiguration event should be send before sending any other event. Following should be the format for SessionConfiguration event:

```
com.apama.db.kdb.SessionConfiguration("KdbTransport","KDB",{ });
```

where first parameter is the transport name and second parameter should always be "KDB"

logFile should be given fullpath to the file location

After that database related events could be send, e.g.

```

com.apama.db.Lookup(2,"KDB","","","KdbTransport","trade",[],""),"select from
trade",{ "Query": "Y" });
com.apama.db.Store(3,"KDB","","","KdbTransport","trade",{ },[]," `trade
insert(09:47:00.000; `IBM;10.9;680)",{"async": "Y" });

```

where trade is the table name already present in the database which is being created in the database by the following statement:

```
trade: ([]time:`time$();sym:`symbol$();price:`float$();size:`int$())
```

Transport name should be used in each event as database name since transport would be connecting to only one database.

Upsert statement can be run either by specifying the field with respective values or directly the query in the event interface. An update would be significant only in keyed tables otherwise a new row would be added with the same values in non-keyed tables.

An example of upsert event is:

```
com.apama.db.Store(5,"KDB","", "", "KdbTransport", "tradeKeyed", {"time":"05:17:00.000", "sym":"IBM", "price": "11.0", "size":"780" }, [], "", {});
```

where tradeKeyed is the table name keyed on time, so the record with that time would be updated. Table definition should be given in entityDefinition property of transport in the parameter file like:

```
<property name="entityDefinition" value="tradeKeyed:(time:`time$();sym:`varchar$();price:`float$();size:`int$())"/>
```

Otherwise Adapter would throw an error. So, all tables need to be defined where an upsert is required.

An upsert with query only, can be given like:

```
com.apama.db.Store(3,"KDB","", "", "tradeQ", "trade", {}, [], "`trade insert(09:47:00.000;`IBM;10.9;680)", {"async":"Y"});
```

If field values are null, then only it will consider the query field of the event, otherwise query would be neglected. In the above event, in extraParams, a field "async" is defined which tells that this upsert is asynchronous and the user doesn't want any response of the running, meaning "Run it and forget it". In the other case, where this field "async" is not specified, a response would be sent after running the upsert.

An example of Execute event is:

```
com.apamax.db.Execute(10,"KDB","", "", "KdbTransport", "f", {"x":"3"}, "", {})
```

where f is the function name already defined in the q database.

A function on q command prompt can be defined like:

```
f: {[x] x*x}
```

where x is the parameter and x*x is the result of the function. This function definition should be present in the entityDefinition property of the transport, otherwise function execution would be rejected from the transport. Entity can be defined in following way:

```
<property name="entityDefinition" value="f:(x:`int$())"/>
```

Since functions in q dont have a specific parameter type or return type, so while defining the entity, type should be given for whatever input this function is going to be called with. Like for same function, to call it with float values, entity can be defined like:

```
<property name="entityDefinition" value="f:(x:`float$())"/>
```

With the execute statement, tables etc. can also be created:

```
com.apamax.db.Execute(14,"KDB","", "", "KdbTransport", "", {}, "tradeCreate2: ([sym: `symbol$()] price:`float$();size:`int$())", {});
```

A delete can also be executed using execute statment:

```
com.apamax.db.Execute(12,"KDB","", "", "KdbTransport", "", {}, "trade:delete from trade where sym=`a", {});
```

Function execution can also be achieved by omitting the argument name and value pair dictionary and by directly giving q language syntax of function execution in the query field of the event:

```
com.apamax.db.Execute(10,"KDB","", "", "KdbTransport", "", {}, "f 5", {})
```

Different data types should be given in java format on the correlator side as mentioned on page 45 of Kdbplusversion3_0.pdf. For example, a data value in Minute format would be a complete integer value like 50,640 etc. but on q prompt it is given in ':' format like 00:40, 06:40 etc. So while inputting a value using an event from the correlator, a Minute type value should be complete integer. Same for Month and Second data types.

Bytes should not be given in hexadecimal format. Byte value should be given as an integer value in the byte range -128 to 127.

Real types value can be suffixed with 'e' but power should also be specified. Like '2.3e' is valid on q prompt but will not valid through the adapter. Instead of '2.3e' , valid format would be '2.3e0' which specifies the exponent power as zero.

Long and short should not suffixed (with 'j' or 'h' respectively) as on the q prompt. But an integer value should be specified in their respective range.

## Update API call

KDB+ database can have various customizations especially the tickerplants and trade & quote plants. These plants are set up with various configurations and different api calls could be supported on these plants which are not supported on normal kdb+ databases. Like ".u.upd" call for update/insert statements on normal kdb+ database will throw exception while it will work perfectly well on a tickerplant because ".u.upd" function has to be defined while setting up the tickerplant. On the other side, "upsert" api call on normal kdb+ will perform the update or insert of data in to the table of the kdb+ database while it doesnt work in this way for a tickerplant. Tickerplant has a complex kdb+ system and it has more than one database in it. These databases are in-memory databases and historical databases. So, it could be possible that "upsert" api call request for update/insert will be forwarded only to the historical database by the ticker while it is expected that it should update the in-memory database as well. So, if a customer is using a kdb+ system where ".u.upd" is being defined, it is recommended that ".u.upd" api call has to be used as update api call. If normal kdb+ database is used, "upsert" should be used.

".u.upd" can be defined for the normal database by executing the following command on the q prompt of the database:

```
.u.upd:upsert
```

Then .u.upd can also be used as update api call for normal databases but it will behave same as upsert.



# 24 Lava FX FIX Adapter

---

■ Prerequisites .....	278
■ IAF configuration .....	278
■ Service monitor injection order .....	279
■ Session configuration .....	279
■ Sample events .....	279

This file describes the design, implementation and usage of the FIX Lava extensions.

The Lava FIX system deviates from standard FIX 4.3 in a number of ways which have made it necessary to provide a number of extensions to the FIX adapter. The extensions fall into two areas: IAF configuration and service monitors.

## Files

All paths are relative to the "adapters" subdirectory of the Apama installation:

- doc/ChangeLog-Lava-FIX.txt
- config/FIX43-Lava.xml
- config/FIX-Lava.xml.dist
- catalogs/bundles/FIX-Lava.bnd
- Lava_FX_FIX-adapter-version.txt

## Prerequisites

---

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

## IAF configuration

---

When acting as a client to Lava, the IAF should be configured with the Lava- specific configuration file, the distribution version of which can be found here:

```
${APAMA_HOME}/adapters/config/FIX-Lava.xml.dist
```

Note that the requirement for a Lava-specific configuration file means that an IAF using this configuration file must be used for Lava FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for Lava operation. If you need to connect to other FIX servers as well as Lava, the non-Lava sessions should be configured in a separate IAF instance. However, Lava and non-Lava session can co-exist safely within the same correlator instance.

As well as the standard host, port, sender and target CompID parameters, the following parameters must also be supplied for all Lava sessions:

```
<!-- Username -->
<property name="StaticField.body.A.553" value="<LAVA_USER>"/>
<!-- Password -->
<property name="StaticField.body.A.554" value="<LAVA_PASSWORD>"/>
<!-- start/end time/day properties and default values ---->
<!--
Start/end days/times are based on days as well for LAVA.
LAVA's trading session is Sun 5pm to Fri 5pm.
Note that you must start the adapters before the
start day/time.
```

```
-->
<property name="QuickFIX.StartTime" value="22:31:00" />
<property name="QuickFIX.EndTime" value="21:59:00" />
<property name="QuickFIX.StartDay" value="sun" />
<property name="QuickFIX.EndDay" value="fri" />
```

The session start/end time specifications always use UTC - users may need to update their configuration files to account for daylight savings time changes, depending on their local time zone.

**Note:**

The `QuickFIX.ResetOnLogon` property must be set to "Y" for market data connections and "N" for trading connections.

## Service monitor injection order

Inject the monitors listed in [“FIX order management session” on page 82](#) and [“FIX Legacy Market data session” on page 83](#).

## Session configuration

Lava sessions must be configured with the following configuration parameters:

```
MARKET DATA SESSIONS:
SubscriptionManager.Aggregate = true
SubscriptionManager.SupportNonUniqueMDEntryIDs = true
```

```
TRADING SESSIONS:
OrderManager.SendExecutionAck = true
OrderManager.ServerWorkaround = 2
```

For example, for most Lava client installations the following session configuration could be used:

```
com.apama.fix.SessionConfiguration("LAVA_MARKET_DATA", {"FixChannel":"LAVA_FIX",
  "SubscriptionManager.Aggregate":"true"})
// Lava trading
com.apama.fix.SessionConfiguration("LAVA_TRADING", {"FixChannel":"LAVA_FIX",
  "OrderManager.SendExecutionAck":"true", "OrderManager.ServerWorkaround":"2"})
```

**Note:**

Following features were not supported by server at the time of certification

- "Fill or Kill(FOK)" and "Good Till Date(GTD)" orders that is, "59"="4"(Fill or Kill) or "59":"6"(Good Till Date).
- "All or None(AON)" and "Reinstate on System Failure" type instructions for order handling on exchange trading floor "18"="G" and "18"="H".

## Sample events

Subscription:

```
com.apama.marketdata.SubscribeDepth
("FIX", "LAVA_MARKET_DATA", "Symbol1",{})
```

### Placing Order:

```
com.apama.oms.NewOrder  
("order1", "GOOG", 399, "BUY", "LIMIT", 500, "FIX",  
"", "", "LAVA_TRADING", "", "", {})
```

ICEBERG Order with hidden quantity. Tag 111(MaxFloor) indicates quantity to be shown. The Quantity field of NewOrder holds the total quantity including the hidden amount.

```
com.apama.oms.NewOrder  
("order1", "EUR/USD",1.27378,"BUY","FOREX ICEBERG",  
500000,"FIX","", "", "LAVA_TRADING", "", "",  
{"15":"EUR","59":"0","111":"200000"})
```

### Security Definition Request:

```
com.apama.fix.InstrumentDataRequest  
("LAVA_MARKET_DATA","1", "EUR/USD", {"64":"SPOT"},[])
```

### Security List Request:

By using "SecurityList.CreateSecurityList = true" parameter parameter, it requests for the SecurityList and cache all the securities and their corresponding data for each security available in exchange.

```
com.apama.fix.SessionConfiguration("LAVA_MARKET_DATA",  
{"FixChannel":"LAVA_FIX", "SubscriptionManager.Aggregate":"true",  
"SecurityList.CreateSecurityList":"true"})
```

# 25 Morgan Stanley FX FIX Adapter

---

■ Prerequisites .....	282
■ IAF configuration .....	282
■ Service monitor injection order .....	282
■ Quote stream .....	282
■ MSFX Ladder Pricing .....	284

The Morgan Stanley FX FIX system deviates from standard FIX 4.2 in a number of ways which have made it necessary to provide a number of extensions to the FIX adapter. The extensions are into two areas: IAF configuration and service monitors.

## Prerequisites

---

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

## IAF configuration

---

When acting as a client to Morgan Stanley FX, the IAF should be configured with the Morgan Stanley FX specific configuration file, the distribution version of which can be found here:

```
${APAMA_HOME}/adapters/config/FIX-MSFX.xml.dist
```

Note that the requirement for a MSFX-specific configuration file means that an IAF using this configuration file must be used for MSFX FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for MSFX operation. If you need to connect to other FIX servers as well as MSFX, the non-MSFX sessions should be configured in a separate IAF instance. However, MSFX and non-MSFX session can co-exist safely within the same correlator instance.

## Service monitor injection order

---

The file `FIX_MSFX_Support.mon` provides quote request message handling. MSFX requires a special handling for sending `QuoteRequest/QuoteCancel`. Therefore, the following EPL file `FIX_MSFX_Support.mon` must be injected into the correlator after the standard FIX adapter service monitors.

Inject the monitors listed in [“FIX order management session” on page 82](#) and [“FIX Legacy Market data session” on page 83](#), followed by:

- `${APAMA_HOME}/adapters/monitors/FIX_MSFX_Support.mon`

## Quote stream

---

### Session configuration

MSFX MD sessions must be configured with the following configuration parameters:

```
SERVICEID = MSFX-FIX
```

e.g. for most MSFX client installations the following session configuration could be used:

```
com.apama.fix.SessionConfiguration("MSFX_TD_Connection", {"SERVICEID":"MSFX-FIX",  
"OrderManager.GenerateNewStyleOrderIds":"true"})
```

## Sample events

### Quote subscription and unsubscription requests:

```
com.apama.marketdata.SubscribeDepth("MSFX-FIX", "MSFX_MD_Connection",
  "GBP/USD",{"Quote":"Y","167":"FOR","38":"100000000"})
com.apama.marketdata.SubscribeDepth("MSFX-FIX", "MSFX_MD_Connection",
  "GBP/USD",{"Quote":"Y","167":"FOR","38":"300000000"})

com.apama.marketdata.UnsubscribeDepth("MSFX-FIX", "MSFX_MD_Connection",
  "GBP/USD",{"Quote":"Y","167":"FOR","38":"100000000"})
com.apama.marketdata.UnsubscribeDepth("MSFX-FIX", "MSFX_MD_Connection",
  "GBP/USD",{"Quote":"Y","167":"FOR","38":"300000000"})
```

#### Note:

If required, on Subscription Reject/Cancel the application has to take care of Quote re-subscription after 1 minute time interval.

The behaviour of the exchange when the quote request is sent for multiple quantities to same symbol is as follows: The quote ID may be same because the values of tag117 always begin with d1 and increments each time when there's an update. As long as the values of tag131 are distinct, values of tag117 can be the same for quote updates.

### NewOrder for "FOREX - PREVIOUSLY QUOTED":

```
com.apama.oms.NewOrder("Order1","GBP/USD",1.56546,"BUY",
  "FOREX - PREVIOUSLY QUOTED",100,"MSFX-FIX","", "", "MSFX_TD_Connection","",
  "",{"117":"d13","131":"1","167":"FOR","Account":"TEST1"})
```

### NewOrder for "MARKET":

```
com.apama.oms.NewOrder("Order2","GBP/USD",1.56587,"BUY","MARKET",100,
  "MSFX-FIX","", "", "MSFX_TD_Connection","", "",{"167":"FOR","Account":"TEST1"})
```

### NewOrder for "FOREX - SWAP":

```
com.apama.oms.NewOrder("Order3","GBP/USD",1.56545,"BUY",
  "FOREX - SWAP",100,"MSFX-FIX","", "", "MSFX_TD_Connection","",
  "",{"117":"d18","131":"1","167":"FOR","Account":"TEST1"})
```

### NewOrder for "LIMIT":

```
com.apama.oms.NewOrder("Order4","GBP/USD",1.5656,"BUY","LIMIT",100,
  "MSFX-FIX","", "", "MSFX_TD_Connection","", "",{"15":"GBP","167":"FOR",
  "18":"1","59":"0","Account":"TEST1"})
```

## MSFX Ladder Pricing

### Session configuration

MSFX MD sessions must be configured with the following configuration parameters:

```
SERVICEID = MSFX-FIX
```

For example, for most MSFX client installations the following session configuration could be used:

```
com.apama.fix.SessionConfiguration("MSFX_MD_Connection", {"SERVICEID":"MSFX-FIX",
  "SubscriptionManager.MarketDataFullRefresh":"true",
  "SubscriptionManager.ReuseRequestID":"false",
  "SubscriptionManager.IncludeTimeInRequestID":"true",
  "SubscriptionManager.ForwardMDReqID":"true"})
com.apama.fix.SessionConfiguration("MSFX_TD_Connection", {"SERVICEID":"MSFX-FIX",
  "OrderManager.GenerateNewStyleOrderIds":"true"})
```

### Custom configuration events

Parameter	Type	Description
SubscriptionManager. ForwardMDReqID	boolean Default: False	Used to forward the value of MDReqID used during the laddered pricing subscription in the Depth and Tick Updates.
MSFXFIX. DelayBeforeQuoteResubscription	float Default: 1.0	When a quote subscription expires, the exchange requests for a resubscription with the same MDReqID. This parameter sets the delay before which the resubscription has to be made. The exchange expects the resubscription to happen between 1 - 15 secs.

### Sample events

#### Laddered pricing stream subscription and unsubscription requests

```
com.apama.marketdata.SubscribeDepth("MSFX-FIX", "MSFX_MD_Connection",
  "GBP/USD", {"167":"FOR", "63":"0"})
com.apama.marketdata.SubscribeDepth("MSFX-FIX", "MSFX_MD_Connection",
  "EUR/USD", {"167":"FOR", "63":"0"})

com.apama.marketdata.UnsubscribeDepth("MSFX-FIX", "MSFX_MD_Connection",
  "GBP/USD", {"167":"FOR", "63":"0"})
com.apama.marketdata.UnsubscribeDepth("MSFX-FIX", "MSFX_MD_Connection",
```

```
"EUR/USD",{ "167": "FOR", "63": "0" })
```

## NewOrder events for the Laddered pricing stream

```
com.apama.oms.NewOrder("0","EUR/USD",0,"BUY","FOREX - PREVIOUSLY QUOTED",1000000,
"MSFX-FIX","", "", "MSFX_TD_Connection", "", "", {"117": "d19a1", "131": "20110916-16:09:35.2",
"15": "EUR", "167": "FOR", "18": "W", "59": "4", "Account": "TEST1"})
com.apama.oms.NewOrder("2","GBP/USD",0,"BUY","FOREX - PREVIOUSLY QUOTED",4000000,
"MSFX-FIX","", "", "MSFX_TD_Connection", "", "", {"117": "d21a3", "131": "20110916-16:09:35.1",
"15": "GBP", "167": "FOR", "18": "W", "59": "4", "Account": "TEST1" })
```

where,

- Tag 131 (MDReqID): MDReqID used to subscribe to the particular ladder stream, users have to set "SubscriptionManager.ForwardMDReqID": "true" to get the MDReqID in the Depth updates
- Tag 117: QuoteID: Should be the value of QuoteEntryID picked up from the Depth update
- Tag 18: W/X: The message from which this QuoteID is picked up (W=Snapshot, X=Incremental)

## Sample depth update

```
com.apama.marketdata.Depth("GBP/USD",
[1.5815,1.58148,1.58123,1.5811,1.5811],
[1.58162,1.58164,1.58189,1.58202,1.58202],
[1.58156,1.58156,1.58156,1.58156,1.58156],
[1000000,1000000,2000000,3000000,3000000],
[1000000,1000000,2000000,3000000,3000000],
{"167": "FOR", "264": "0", "35": "W", "52": "20110916-10:39:39.400",
"60": "20110916-10:39:39.400", "63": "0",
"ASK1_1023": "1", "ASK1_1070": "1", "ASK1_FutSettDate": "20110920",
"ASK1_QuoteEntryID": "d1a1", "ASK1_SettlmntTyp": "0",
"ASK2_1023": "2", "ASK2_1070": "1", "ASK2_FutSettDate": "20110920",
"ASK2_QuoteEntryID": "d1a2", "ASK2_SettlmntTyp": "0",
"ASK3_1023": "3", "ASK3_1070": "1", "ASK3_FutSettDate": "20110920",
"ASK3_QuoteEntryID": "d1a3", "ASK3_SettlmntTyp": "0",
"ASK4_1023": "4", "ASK4_1070": "1", "ASK4_FutSettDate": "20110920",
"ASK4_QuoteEntryID": "d1a4", "ASK4_SettlmntTyp": "0",
"ASK5_1023": "5", "ASK5_1070": "1", "ASK5_FutSettDate": "20110920",
"ASK5_QuoteEntryID": "d1a5", "ASK5_SettlmntTyp": "0",
"BID1_1023": "1", "BID1_1070": "1", "BID1_FutSettDate": "20110920",
"BID1_QuoteEntryID": "d1b1", "BID1_SettlmntTyp": "0",
"BID2_1023": "2", "BID2_1070": "1", "BID2_FutSettDate": "20110920",
"BID2_QuoteEntryID": "d1b2", "BID2_SettlmntTyp": "0",
"BID3_1023": "3", "BID3_1070": "1", "BID3_FutSettDate": "20110920",
"BID3_QuoteEntryID": "d1b3", "BID3_SettlmntTyp": "0",
"BID4_1023": "4", "BID4_1070": "1", "BID4_FutSettDate": "20110920",
"BID4_QuoteEntryID": "d1b4", "BID4_SettlmntTyp": "0",
"BID5_1023": "5", "BID5_1070": "1", "BID5_FutSettDate": "20110920",
"BID5_QuoteEntryID": "d1b5", "BID5_SettlmntTyp": "0",
"MDReqID": "20010909-07:16:40.1", "Market": "MSFX_MD_Connection",
"SERVICE_NAME": "MSFX-FIX"})
```

where,

ASK1_1070, ASK2_1070 ..., BID1_1070, BID2_1070 ... represent the MDQuoteType for each corresponding bid/offer entries. The possible values for MDQuotType (1070) are:

- 1: Implies a Tradable Quote
- 2: Implies a Indicative Quote

If this field is omitted, it defaults to 1 (Tradable Quote) and ASK1_QuoteEntryID ....  
BID1_QuoteEntryID .... represent the QuoteID for the corresponding bid/offer entry. users have to use this value as the value of Tag 117 when generating the New Orders.

# 26 Reuters RFA Adapter

---

■ Prerequisites .....	288
■ Transport configuration parameters .....	290
■ Manually loading Reuters configurations .....	293
■ Overriding Default FIDs For TICK/DEPTH generation .....	294
■ Retrieving Level 2 Data From MARKET PRICE model .....	294
■ Publish empty Depths/Ticks .....	295
■ Dictionary path .....	295
■ RIC translator .....	295
■ Service monitor injection order .....	296
■ Event details .....	296
■ Working with OMM consumer .....	306
■ Working with OMM Non-Interactive provider .....	307
■ Working With OMM NEWS .....	308
■ Working with Machine Readable News .....	310

The Apama Reuters RFA adapter uses the Apama Integration Adapter Framework (IAF) and the Reuters RFA library to connect to a Reuters market data system. This supports Open Message Model(OMM) Consumer, OMM Non-Interactive Provider for Reuters Data Feed Direct (RDFD) using Reuters Foundation API (RFA Java).

## Prerequisites

---

### ReutersRFA Adapter Software Requirements

1. `rfa.jar` (v8.0.0.L2) (shipped with the adapter)
2. Configuration Tool:

This comes along with the ReutersRFA Adapter(`config/RFA_ConfigTool.zip`). Platform dependent startup scripts `config_editor.bat` (for Windows) and `config_editor.ksh`(for Unix/Linux) are provided in the `RFA_ConfigTool.zip` to activate the configuration UI tool. The RFA Configuration Tool provides a graphical user interface (GUI) to manage the configuration database of RFA Session Layer Software Package. It can be used to populate (import) configuration database from an XML document, manipulate in-memory configuration data and back up (export) configuration database to an XML document. Configuration done using the UI tool will be stored to the system root.

In Windows this information is stored in the registry that is, `HKEY_LOCAL_MACHINE\Software\JavaSoft\Prefs` for system Preferences and `HKEY_CURRENT_USER\SOFTWARE\JavaSoft\Prefs` for user Preferences in a very fluffy format.

In Linux, user preferences are stored in `~/.java` and System Preferences are stored in `/etc/.java` For more information, refer to Reuters RFA JAVA SDK

### Setup

The Editor is packaged in `config_editor.jar`. Assume that the JAR file is installed in the following directory: `/Reuters/RFAJ/Tools/config_editor.jar`

This file need to be included in the class path in order to run the editor. When setting the `CLASSPATH` environment variable, specify the full file name of the JAR file. The `CLASSPATH` must be set prior to starting the RFA configuration Editor to ensure the VM picks up the correct value.

### Windows

A batch file `config_editor.bat` is provided in the directory `/Reuters/RFAJ/Tools`. The parameter Editor can be invoked by double-clicking on the file `config_editor.bat`. Alternatively, in a Command Prompt shell, type: `set EDITOR_PATH=C:\Reuters\RFAJ\Tools\ set CLASSPATH=%CLASSPATH%;%EDITOR_PATH%` where "C:" is the drive on which both JAR files reside.

## Unix

A ksh script file `config_editor.ksh` is provided in the directory `/Reuters/RFAJ/Tools`. The configuration Editor can be invoked by executing the script file `config_editor.ksh`. Alternatively, in a ksh, type: `export EDITOR_PATH=/Reuters/RFAJ/Tools/config_editor.jar export CLASSPATH=$CLASSPATH:$EDITOR_PATH`

## Running the Editor

By using default XML parser contained in JRE, Java `com.reuters.rfa.tools.config.editor.ConfigEditor`

## Use the Editor

Execute the appropriate script to start the configuration tool then click on `File->Import->File->RSSLsampleConfig.xml` to import the file. The sample configuration file has the value of "localhost" for the configuration variable name `serverList`. Change this value to the name of the host on which the P2PS or Source Distributor is running.

Following steps to be followed for creating namespace

1. Create a namespace node under `users->com->reuters`  
ex: `ApamaNameSpace`
2. Create connection and session nodes under namespace
3. Under connection, create a new connection with the following properties
  - `connectionType` : Consumer Connection Type (RSSL), Non-Interactive Provider connection type(RSSL_NIPROV)

**Note:**  
RFA Adapter doesn't support SSL connection type.

  - `serverList` : Server IP ex: localhost
  - `portNumber` : Server port ex: 14002
4. Under session, create a new session with the following properties
  - `connectionList` : can be mapped to connections configured as above
  - `shareConnections` : false (for better performance)

The namespace is ready and can start configuring RFA adapter.

**Note:**  
Ensure that the user has root privileges to create preferences using configuration tool.

DISCLAIMER

IAF adapter doesn't make use of RFA_ConfigTool. It is dependent on configurations stored in memory (as explained above). There is no relation between IAF configuration file and RFA configuration Tool.

CMF version equal or better than 10.1

## Transport configuration parameters

The following parameters has to be set in the IAF configuration file ( see the manual "Apama Platform Development Environment - The Integration Adapter Framework" for how to do this):

Name	Parameter Name
REUTERS_RFA_TRANSPORT_NAME	The name of the ReutersRFA adapter transport.
RFA_CONSUMER_THREADCOUNT	This parameter should be set to an optimum value based on the incoming message rate and number of CPUs available on the machine.  Defaults to 4
PUBLISH_DEPTH_FROM	From which the market Depth should be constructed and published Possible valid values: MARKET_PRICE / MARKET_BY_PRICE / MARKET_BY_ORDER  Defaults to MARKET_PRICE
PUBLISH_DEPTH_LEVEL (Optional)	Level of market depth to be published.  Defaults to 5  <b>Note:</b> Single level depth will be published if PUBLISH_DEPTH_FROM is MARKET_PRICE
RIC_TRANSLATOR_FILE_PATH (Optional)	Absolute file path where RIC translation file exists. Sample file exists in adapter parameter directory (ric-translation.txt. Invalid path leads to ignore the RIC translation.
INCLUDE_FIELDS (Optional)	List of fields to be considered while decoding OMM Message for payload. For this the ACRONYMs from RDM dictionary should be used and separated by comma(","). The asterisk character ("*") can be used to include all fields.  Defaults to None ("").  For Example: QUOTIM_MS, TIMCOR_MS
EXCLUDE_FIELDS (Optional)	List of fields to be ignored while decoding OMM Message for payload. For this the ACRONYMs from RDM

Name	Parameter Name
	<p>dictionary should be used and separated by comma(","). The asterisk character ("*") can be used to exclude all fields.</p> <p>Defaults to None ("").</p> <p>Example combinations:</p> <ol style="list-style-type: none"> <li>1. Include all fields INCLUDE_FIELDS="*" and EXCLUDE_FIELDS=""</li> <li>2. Include QUOTIM_MS &amp; TIMCOR_MS INCLUDE_FIELDS="QUOTIM_MS,TIMCOR_MS" and EXCLUDE_FIELDS=""</li> <li>3. Include all fields except QUOTIM_MS &amp; TIMCOR_MS INCLUDE_FIELDS="*" and EXCLUDE_FIELDS="QUOTIM_MS,TIMCOR_MS"</li> <li>4. Exclude all fields INCLUDE_FIELDS="*" and EXCLUDE_FIELDS="*"</li> </ol> <p><b>Note:</b> Exclude fields has high precedence than include fields.</p>
REPORT_UPDATES (Optional)	<p>Subset of include fields to be considered to generate update events. For this the ACRONYMS from RDM dictionary should be used and separated by comma(","). The asterisk character ("*") can be used to specify all include fields.</p> <p>Defaults to None ("") (No updates will be generated on payload changes).</p> <p>For Example: QUOTIM_MS</p>
RFA_LOGGING_LEVEL (Optional)	<p>A measure of severity for a message, it provides control over which messages to be displayed. RFA Java uses four logging levels:</p> <ul style="list-style-type: none"> <li>■ SEVERE : Used for catastrophic errors that require immediate attention</li> <li>■ WARNING : Used for serious problems that must be noted</li> </ul>

Name	Parameter Name
	<ul style="list-style-type: none"> <li>■ INFO : Used for normal informational messages; it is the default logging level</li> <li>■ FINE : Used for detailed logging for debug purposes (that is, it provides tracing information)</li> <li>■ Default value is 'INFO'</li> </ul>
RFA_LOGGING_LOCATION (Optional)	<p>File path where all the logs should be stored.</p> <p>Default value is RFA_%u.log</p>
RFA_LOGGING_APPEND (Optional)	<p>If this is 'true' and given file path exists then appends logs to the file else overwrites them.</p> <p>Default value is true.</p>
EscapeFields (optional)	<p>This property carries set of fields which need to be validated and apply string escaping if there is any special character. Default values are BCAST_TEXT, SEG_TEXT, TOPIC_CODE, CO_IDS, DSPLY_NAME, DSPLY_NMLL, GN_TXT16_2</p>
MPDepth_RICType (optional)	<p>This is applicable if "PUBLISH_DEPTH_FROM" is MARKET PRICE. Valid values are 'tick' or 'depth'.</p> <p>Default value is 'tick'</p> <p>The MARKET PRICE model gets both depth data. This depth can be either top of the book or order book. Client can request for either type by configuring transport properties "Depth.BID.Price", "Depth.ASK.Price" etc. RIC for these two depth types will be different.</p> <p>For example: ricTranslation.txt has following mapping</p> <pre data-bbox="662 1373 1365 1444" style="background-color: #f0f0f0; padding: 5px;">^(\w+).L\$; tick=\1.L; depth=\1.LO</pre> <p>which means, if subscribed for symbol ABC.L then internally for tick request will translate as ABC.L but for depth request it will be translated as ABC.LO</p> <p>If interested in top of the book data then configure:</p> <p>MPDepth_RICType='tick'. RIC will be ABC.L</p> <p>If interested in depth book data then configure:</p> <p>MPDepth_RICType='depth'. RIC will be ABC.LO</p>

Name	Parameter Name
AllowEmptyDepths AllowEmptyTicks	<p>Publish empty depths. By default, TICK/DEPTH is not published when there is no FID in the OMM message which matches default FIDs. Still, if you are interested in receiving empty depths then following property will enable it</p> <pre>&lt;property name="AllowEmptyDepths" value="true"/&gt; &lt;property name="AllowEmptyTicks" value="true"/&gt;</pre> <p>Defaults to false.</p>
LogMessages	<p>There are some messages which will be logged for every request received by the transport. In normal condition this is unnecessary overhead but when there is any problem such messages are required for debugging purpose. You can enable this property using:</p> <pre>&lt;property name="LogMessages" value="true"/&gt;</pre> <p>Defaults to false.</p>
DACS_ApplicationID	<p>Set the value of ApplicationId in LOGIN_REQ. If not set, ApplicationId will default to transport name.</p>

## Manually loading Reuters configurations

### DisableConfigTool

If this is true then you can load Reuters settings from a transport properties rather than accessing the registry. Default value is false.

If property `DisableConfigTool` is set true then you need to define connections/sessions. Reuters have a conventions for this,

```
<namespace>.Connections.<RSSL Connection name>.<parameter>
```

or

```
<namespace>.Sessions.<RSSL session name>.<parameter>
```

For a connection, you have the following parameters defined,

1. `connectionType`. Specifies the connection type. The value of this parameter must be RSSL.
2. `serverList`. Comma-separated list of hostnames and/or IP addresses to attempt connections to.
3. `portNumber`. Port number used when connecting to each server in the `serverList`.

For a session, you have the following parameters,

1. `connectionList`. A comma delimited string that lists connections used by the session

For example, following are the properties to be defined in IAF configuration file

```
<property name="RSSLNamespace.Connections.rsslConnection.connectionType"
  value="RSSL"/>
<property name="RSSLNamespace.Connections.rsslConnection.serverList"
  value="ukimd0ph01s, ukimd0ph02"/>
<property name="RSSLNamespace.Connections.rsslConnection.portNumber"
  value="14002"/>
<property name="RSSLNamespace.Sessions.rsslSession1.connectionList"
  value="rsslConnection"/>
```

## Overriding Default FIDs For TICK/DEPTH generation

MARKET PRICE request to RMD5 gets both TICK/Depth data. But specific set of values are defined in generating TICK/DEPTH.

Ex: Price in TICK will be picked from FID 6 in OMM message

Quantity in TICK will be picked from FID 178 in OMM message.

Following are transport properties with the default FIDs,

```
<property name="Tick.Price" value="6"/>           <!-- defaults to TRDPRC_1 -->
<property name="Tick.Quantity" value="178"/>     <!-- defaults to TRDVOL_1 -->
Below properties support more than one FID separated by comma
<property name="Depth.BID.Price" value="22"/>   <!-- defaults to BID -->
<property name="Depth.BID.Quantity" value="30"/> <!-- defaults to BIDSIZE -->
<property name="Depth.ASK.Price" value="25"/>   <!-- defaults to ASK -->
<property name="Depth.ASK.Quantity" value="31"/> <!-- defaults to ASKSIZE -->
```

If you are interested in editing these properties, that is bid price in depth should be picked from FID -8009 then

```
<property name="Depth.BID.Price" value="-8009"/>
  <property name="Depth.ASK.Price" value="-7999"/>
```

## Retrieving Level 2 Data From MARKET PRICE model

Clients from Mexico/South Africa are interested in retrieving level 2 market data from MARKET PRICE model. According to Reuters, those exchanges are what they call consolidated data (meaning Reuters first sends exchange's market data to one of its datacenters and then redistribute it to others), they will send it always as MARKET_PRICE, both for Level 1 and Level 2.

```
<property name="Depth.BID.Price" value="436,437,438,439,440"/>
  <property name="Depth.BID.Quantity" value="730,731,732,733,734"/>
  <property name="Depth.ASK.Price" value="441,442,443,444,445"/>
  <property name="Depth.ASK.Quantity" value="735,736,737,738,739"/>
```

## Publish empty Depths/Ticks

By default, TICK/DEPTH is not published when there is no FID in the OMM message which matches default FIDs. Still, if you are interested in receiving empty depths/ticks then following property will enable it

```
<property name="AllowEmptyDepths" value="true"/>
  <property name="AllowEmptyTicks" value="true"/>
```

Defaults to false.

Ex:

```
com.apama.marketdata.Depth("ABC", [], [], [], [], [{"ACVOL_1": "100",
  "ASK": "10.2000", "ASK_TIME": "19:12:23", "BID": "9.8000",
  "DIVPAYDATE": "25 DEC 2006", "Market": "ReutersRFATransport1",
  "RDNDISPLAY": "100", "RDN_EXCHID": "SES", "SERVICE_NAME": "DIRECT_FEED",
  "TRDPRC_1": "10.0000"}])

com.apama.marketdata.Tick("ABC.CO", -1, -1, {"Market": "ReutersRFATransport",
  "SERVICE_NAME": "DIRECT_FEED"})
```

## Dictionary path

It is not necessary that every source use same standard RDM dictionary for parsing OMM data. The following properties are introduced to add dictionaries,

```
<property name="FieldDictionary.FileName" value="RDM_Dictionaries/RDMFieldDictionary"/>
  <property name="EnumDictionary.FileName" value="RDM_Dictionaries/enumtype.def"/>
```

```
FieldDictionary.FileName --> File path of Field dictionary
EnumDictionary.FileName --> File path of enum dictionary
```

By default it takes dictionaries path which are shipped with adapter.

## RIC translator

The Reuters instrument code (RIC) used for tick and level 1 data (best bid and ask) is same, and that used for level 2 data (the full order book) are often different on most exchanges.

Ex : On London Stock Exchange

VOD.L is the RIC for tick and level 1 VOD/LO is the RIC for level 2

On Frankfurt Exchange

BMW.DE for tick and level 1 1BMW.DE for level 2

Often the rules for the RIC is consistent on an exchange, but different across exchanges.

The RIC translation utility allows the user of the Reuters service monitors to use a single RIC to subscribe to both tick, level 1 and level 2 data . This translation mainly depends on a file which

contains RIC translation mapping. This file path can be configured through codec property 'RIC_Translator_File_Path'.

Transport converts the raw RIC based on the type of the request and uses it for subscriptions.

## Service monitor injection order

The following monitors should be injected into the correlator in the order listed here when using the RFA adapter:

- `${APAMA_HOME}/adapters/monitors/IAFStatusManager.monReutersRFA_ConfigEvents.mon`
- `${APAMA_HOME}/adapters/monitors/ReutersRFA_Events.mon`
- `${APAMA_HOME}/adapters/monitors/ReutersRFA_IAFEvents.mon`
- `${APAMA_HOME}/adapters/monitors/ReutersRFA_Interface.mon`
- `${APAMA_HOME}/adapters/monitors/ReutersRFA_StatusPublisher.mon`
- `${APAMA_HOME}/adapters/monitors/ReutersRFA_SessionManager.mon`
- `${APAMA_HOME}/adapters/monitors/ReutersRFA_LogonManager.mon`
- `${APAMA_HOME}/adapters/monitors/ReutersRFA_TickUtils.mon`
- `${APAMA_HOME}/adapters/monitors/ReutersRFA_DepthUtils.mon`
- `${APAMA_HOME}/adapters/monitors/ReutersRFA_TickPublisher.mon`
- `${APAMA_HOME}/adapters/monitors/ReutersRFA_DepthPublisher.mon`
- `${APAMA_HOME}/adapters/monitors/ReutersRFA_MP_DepthPublisher.mon`
- `${APAMA_HOME}/adapters/monitors/ReutersRFA_SubscriptionManager.mon`
- `${APAMA_HOME}/adapters/monitors/ReutersRFA_DataPublisher.mon`

## Event details

**com.apama.rfa.SessionConfiguration(string TRANSPORT, dictionary <string,string> params)**

This initiates each instance of the ReutersRFA adapter, allowing multiple instances within the same correlator. Also, ReutersRFA can be used with multiple IAF instances.

Following are the Parameter descriptions and default values

Parameter Name	Description
applicationType	ReutersRFA adapter can be configured as OMMConsumer or OMMNIPProvider

Parameter Name	Description
	Defaults to OMMConsumer.
codecName	Name of the Codec Defaults to ReutersRFACodec
heartBeatInterval	To keep IAF alive Defaults to 10.0
channel	IAF listens on this channel Defaults to ReutersRFA
RFASessionName	RFA session name as configured using parameter tool Defaults to ApamaNameSpace::Session
TokenTimeout	Time limit to receive token from FIX session Default 10.0
OMMNIProvider. LogLatency	Adds support for latency measurement on provider messages, string timestamp set dictionary is added. Defaults to false
RFALogonTimeout	Time limit to receive logon response Default 250.0
RFALoginInterval	Time interval to send next login request Default 5.0
RFALoginsLimit	In a session number of login requests that can be sent. By default there is no limit for login requests.

**Note:**

Session parameters logLevel, logFile, logAppend have been removed.

**com.apama.rfa.Login(string marketId, string userName, dictionary <string,string>  
params)**

Properties

Parameter Name	Description
marketId	Name of the market (TRANSPORT)

Parameter Name	Description
username	Name of the user

## Parameters

Name	Description
DACS_ApplicationID	Set the value of ApplicationId in LOGIN_REQ. If not set, ApplicationId will default to transport name.

### com.apama.marketdata.SubscribeTick(string serviceId, string marketId, string symbol, dictionary <string,string> params)

Parameter Name	Description
serviceId	Name of the service deployed in the RMDS

### com.apama.marketdata.SubscribeDepth(string serviceId, string marketId, string symbol, dictionary <string,string> params)

Parameter Name	Description
serviceId	Name of the service deployed in the RMDS
PublishDepthLevel (Optional)	Level of market depth to be published for the current subscription symbol. Defaults to the value specified at TRANSPORT (PUBLISH_DEPTH_LEVEL)

## Data publishing events

### Directory

```
com.apama.rfa.DIRECTORY_RESP(string TRANSPORT, string RespType
    integer RespTypeNum, string IndicationMask,
    sequence<integer> dataMask,
    dictionary<string, com.apama.rfa.Service> ServicesMap,
    dictionary<string, string> __payload)
```

Following are the parameter descriptions and default values

Parameter	Description
RespType	Response type can be REFRESH_RESP, STATUS_RESP, UPDATE_RESP

Parameter	Description
RespTypeNum	Response type number can be 0 (SOLICITED),1(UNSOLICITED)
IndicationMask	Indication mask can be ATTRIB_INFO_IN_UPDATES, CONFLATION_INFO_IN_UPDATES, CLEAR_CACHE, DO_NOT_CACHE, DO_NOT_CONFLATE, DO_NOT_RIPPLE, REFRESH_COMPLETE
dataMask	Data mask can be INFO, STATE, GROUP, LOAD, DATA, LINK

Parameter	Description
ServicesMap	<code>com.apama.rfa.Service(SDInfoFilterEntry info, SDStateFilterEntry state,SDGroupFilterEntry group, SDLoadFilterEntry load,SDDataFilterEntry data, SDLinkFilterEntry link)</code>
SDInfoFilterEntry	<p>Information about a service that does not update very often.</p> <pre>com.apama.rfa.SDInfoFilterEntry(string Name, sequence &lt;integer&gt; Capabilities, sequence &lt;string&gt; DictionariesProvided, sequence &lt;string&gt; DictionariesUsed, dictionary &lt;string, string&gt; __payload)</pre> <p>Name. Service name</p> <p>Capabilities. MessageModelTypes that the service can provide. List of all message model types are</p> <ul style="list-style-type: none"> <li>■ MARKET_PRICE = 6</li> <li>■ MARKET_BY_ORDER = 7</li> <li>■ MARKET_BY_PRICE = 8</li> <li>■ MARKET_MAKER= 9</li> <li>■ SYMBOL_LIST = 10</li> </ul> <p>DictionariesProvided. This field can be ignored as RMDS doesn't support dictionary publishing</p> <p>DictionariesUsed. List of Dictionary names that may be required to process all of the data from this service.</p>
SDStateFilterEntry	State of a service.

Parameter	Description
	<pre data-bbox="613 247 1365 342">com.apama.rfa.SDStateFilterEntry(integer ServiceState, integer AcceptingRequests, string Status)</pre> <p data-bbox="613 373 1101 405">ServiceState. 1 (Up/Yes), 0 (Down/No)</p> <p data-bbox="613 432 1385 527">AcceptingRequests. 1 (Yes),0 (No). If the value is 0, then consuming applications should not send any requests to the service provider.</p> <p data-bbox="613 558 1385 653">Status. State that should be fanned out to all open streams in all groups of the service. Format is {Stream,Data,Code,Text}. For example, Unspecified(0),Unspecified(0),None (0),"".</p> <p data-bbox="613 684 1385 758">Stream can accept UNSPECIFIED, OPEN, NONSTREAMING, CLOSED_RECOVER, CLOSED, REDIRECT.</p> <p data-bbox="613 789 1222 821">Data can accept UNSPECIFIED, OK, SUSPECT.</p> <p data-bbox="613 852 1385 1083">Code can accept NONE, NOT_FOUND, TIMEOUT, NOT_ENTITLED, INVALID_ARGUMENT, USAGE_ERROR, PREEMPTED, JIT_CONFLATION_STARTED, REALTIME_RESUMED, FAILOVER_STARTED, FAILOVER_COMPLETED, GAP_DETECTED, NO_RESOURCES, TOO_MANY_ITEMS, ALREADY_OPEN, SOURCE_UNKNOWN, NOT_OPEN, MAX_RESERVED</p>
SDGroupFilterEntry	<p data-bbox="613 1108 1271 1140">Used for transient groups messages for the service.</p> <pre data-bbox="613 1161 1365 1224">com.apama.rfa.SDGroupFilterEntry(string Group, string MergedGroup, string Status)</pre> <p data-bbox="613 1255 1182 1287">Group. Group for this group status message</p> <p data-bbox="613 1318 1385 1381">MergedGroup. Used to merge all of the items from one group into another group.</p> <p data-bbox="613 1413 1385 1476">Status. State that should be fanned out to all open streams in specified group of this service.</p>
SDLoadFilterEntry	<p data-bbox="613 1501 1385 1564">Statistics about how many concurrent streaming requests the service can support and how many it is currently servicing.</p> <pre data-bbox="613 1585 1365 1648">com.apama.rfa.SDLoadFilterEntry(integer OpenLimit, integer OpenWindow, integer LoadFactor)</pre> <p data-bbox="613 1680 1385 1743">OpenLimit. Maximum number of streaming items that the client is allowed to open to this service.</p> <p data-bbox="613 1774 1385 1869">OpenWindow. Maximum number of outstanding new refresh requests (that is, requests for items which are not already open) service can receive at any given time.</p>

Parameter	Description
	LoadFactor. The load level specifies the current load on each source server.
SDDataFilterEntry	<p>Broadcast data that applies to all items requested from that service.</p> <pre>com.apama.rfa.SDDataFilterEntry(integer Type,string Data)</pre> <p>Type. Explanation of the Data.Type is of UInt32.</p> <p>Range is 0 to 1023.Ex : Time(1), Alert (2), Headline (3), Status (4)</p> <p>Data. Data that should be applied to all items from the service that have an ANSI_Page data type.</p>
SDLinkFilterEntry	<p>Provides information which RMDS can use for load balancing.</p> <pre>com.apama.rfa.SDLinkFilterEntry(integer Type, integer LinkState,integer LinkCode,string Data)</pre> <p>Type. This tells us if the back link is interactive or broadcast. This does not describe if the service itself is interactive or broadcast. Range is 1 to 2 (1: Interactive , 2: Broadcast). Default value is 1.</p> <p>LinkState. 0 (Down), 1 (Up)</p> <p>LinkCode:</p> <ul style="list-style-type: none"> <li>■ 0 (None)</li> <li>■ 1 (Ok)</li> <li>■ 2 (RecoveryStarted)</li> <li>■ 3 (RecoveryCompleted)</li> </ul> <p>Data. Explanation of the LinkState and LinkCode.</p>

## Tick

com.apama.rfa.MARKET_PRICE_RESP(string TRANSPORT,string RespType, integer RespTypeNum,string IndicationMask,string serviceId, string symbol,string symbolType, float Price,integer Quantity, dictionary<string,string> __payload)

Following are the Parameter descriptions and default values

Parameter Name	Descriptions
RespType	Response type can be REFRESH_RESP,STATUS_RESP,UPDATE_RESP
RespTypeNum	Response type number can be 0 (SOLICITED),1(UNSOLICITED)
IndicationMask	Indication mask can be ATTRIB_INFO_IN_UPDATES, CONFLATION_INFO_IN_UPDATES,CLEAR_CACHE,DO_NOT_CACHE, DO_NOT_CONFLATE,DO_NOT_RIPPLE,REFRESH_COMPLETE
symbolType	symbolType :0 (UNSPECIFIED),1 (RIC),127 (MAX_RESERVED),255(MAX) Defaults to 1
payload	payload can contain any information like PROD_PERM, CURRENCY, TRD_UNITS, MKT_ST_IND, PR_RNK_RUL, QUOTE_DATE
MktDepthEntries	<p>com.apama.rfa.MKT_BY_PRICE_Entry (string MDUpdateAction, string MDEntryID,string MDEntryType, float MDEntryPRC, integer MDEntrySize,integer NoOrders, dictionary &lt;string, string&gt; __payload)</p> <p>The parameter MP.MapPriceQtyToDefault applies to com.apama.rfa.MARKET_PRICE_RESP</p> <p>Default value of this parameter is "true".</p> <p>By default price and quantity fields are mapped to FIDs "TRDPRC_1" and "ACVOL_1" respectively. While publishing MARKET PRICE data, if you add the parameter MP.MapPriceQtyToDefault to extra parameters:</p> <ul style="list-style-type: none"> <li>■ If its value is "false" then it means that default mappings are not required.</li> <li>■ You can add interested FIDs to extra parameters to which you want to map price/quantity fields.</li> <li>■ If its value is "true" then price/quantity fields are mapped to FIDs "TRDPRC_1" and "ACVOL_1".</li> <li>■ If proper value are assigned to these fields, same will be sent. Else, default values -1.0/-1 are sent out.</li> </ul>

## Depth

```
com.apama.rfa.MARKET_BY_PRICE_RESP (string TRANSPORT,string
  RespType, integer RespTypeNum,string IndicationMask,string serviceId, string
  symbol,string symbolType, sequence<com.apama.rfa.MKT_BY_PRICE_Entry>
  MktDepthEntries, dictionary<string,string> __payload)
```

Following are the Parameter descriptions and default values

Name	Descriptions
RespType	Response type can be REFRESH_RESP,STATUS_RESP,UPDATE_RESP
RespTypeNum	Response type number can be 0 (SOLICITED),1(UNSOLICITED)
IndicationMask	Indication mask can be ATTRIB_INFO_IN_UPDATES, CONFLATION_INFO_IN_UPDATES, CLEAR_CACHE, DO_NOT_CACHE, DO_NOT_CONFLATE, DO_NOT_RIPPLE, REFRESH_COMPLETE
symbolType	symbolType :0 (UNSPECIFIED),1 (RIC),127 (MAX_RESERVED),255(MAX) Defaults to 1
payload	_payload can contain any information like PROD_PERM, CURRENCY, TRD_UNITS, MKT_ST_IND, PR_RNK_RUL, QUOTE_DATE
MktDepthEntries	<pre>com.apama.rfa.MKT_BY_PRICE_Entry(string MDUpdateAction, string MDEntryID,string MDEntryType, float MDEntryPRC, integer MDEntrySize,integer NoOrders, dictionary &lt;string, string&gt; __payload)</pre> <ul style="list-style-type: none"> <li>■ MDUpdateAction : Action.UPDATE,Action.ADD,Action.DELETE</li> <li>■ MDEntryID : unique key</li> <li>■ MDEntryType : Order Side is of enum type. It accepts values defined for it enum.def (1(BID), 2(ASK))</li> <li>■ MDEntryPRC : Order Price</li> <li>■ MDEntrySize : Order Size</li> <li>■ NoOrders : Number of Orders</li> <li>■ payload : Contains any information like QUOTIM_MS</li> </ul>

## Market Maker

```
com.apama.rfa.MARKET_MAKER_RESP(string TRANSPORT, string RespType,integer RespTypeNum,string IndicationMask, string serviceId,string symbol,string symbolType, sequence<com.apama.rfa.MKT_MAKER_Entry> MktMkrEntries, dictionary<string,string>__payload)
```

Following are the Parameter descriptions and default values

Name	Descriptions
RespType	Response type can be REFRESH_RESP, STATUS_RESP, UPDATE_RESP
RespTypeNum	Response type number can be 0 (SOLICITED), 1(UNSOLICITED)

Name	Descriptions
IndicationMask	Indication mask can be ATTRIB_INFO_IN_UPDATES, CONFLATION_INFO_IN_UPDATES, CLEAR_CACHE, DO_NOT_CACHE, DO_NOT_CONFLATE, DO_NOT_RIPPLE, REFRESH_COMPLETE
symbolType	symbolType :0 (UNSPECIFIED),1 (RIC),127 (MAX_RESERVED),255 (MAX) Defaults to 1
payload	payload can contain any information like PROD_PERM, CURRENCY, TRD_UNITS, MKT_ST_IND, PR_RNK_RUL, RDN_EXCHD2, QUOTE_DATE
MktMkrEntries	<pre>com.apama.rfa.MKT_MAKER_Entry(string MDUpdateAction, string MDEntryID,float BidPRC,integer BidSize, float AskPRC,integer AskSize,string MKT_Source, string MKT_MKR_Name,string PRC_Code, dictionary &lt;string, string&gt; __payload)</pre> <ul style="list-style-type: none"> <li>■ MDUpdateAction : Action.UPDATE,Action.ADD,Action.DELETE</li> <li>■ MDEntryID : unique key</li> <li>■ BidPRC : Bid Price</li> <li>■ BidSize: Bid Size</li> <li>■ AskPRC: Ask Price</li> <li>■ AskSize: Ask Size</li> <li>■ MKT_Source: Market Source is of enum type. It accepts values defined in enum.def</li> <li>■ MKT_MKR_Name: Market Maker Name</li> <li>■ PRC_Code: Price qualifier code (PRC_QL_CD, PRC_QL2) is of enum type. It accepts values defined in enum.def</li> <li>■ payload : Contains any information like QUOTIM_MS</li> </ul>

## Market By Order

```
com.apama.rfa.MARKET_BY_ORDER_RESP(string TRANSPORT, string RespType,integer RespTypeNum,string IndicationMask, string serviceId,string symbol,string symbolType, sequence<com.apama.rfa.MKT_BY_ORDER_Entry> MktOrderEntries, dictionary<string,string> __payload)
```

Following are the Parameter descriptions and default values

Parameter Name	Descriptions
RespType	Response type can be REFRESH_RESP, STATUS_RESP, UPDATE_RESP

Parameter Name	Descriptions
RespTypeNum	Response type number can be 0 (SOLICITED), 1(UNSOLICITED)
IndicationMask	Indication mask can be ATTRIB_INFO_IN_UPDATES, CONFLATION_INFO_IN_UPDATES, CLEAR_CACHE, DO_NOT_CACHE, DO_NOT_CONFLATE, DO_NOT_RIPPLE, REFRESH_COMPLETE
symbolType	symbolType :0 (UNSPECIFIED), 1 (RIC), 127 (MAX_RESERVED), 255 (MAX) Defaults to 1
payload	payload can contain any information like <ul style="list-style-type: none"> <li>■ PROD_PERM, CURRENCY, TRD_UNITS,</li> <li>■ MKT_ST_IND, PR_RNK_RUL, OR_RNK_RUL, ACTIV_DATE, STOCK_RIC</li> </ul>
MktOrderEntries	<p>com.apama.rfa.MKT_BY_ORDER_Entry(string MDUpdateAction, string MDEntryID, string MDEntryType, float MDEntryPRC, integer MDEntrySize, dictionary &lt;string, string&gt; __payload)</p> <p>MDUpdateAction. Action.UPDATE, Action.ADD, Action.DELETE</p> <p>MDEntryID. Unique key.</p> <p>MDEntryType. ORDER_SIDE (1(BID), 2(ASK)) is of enum type. It accepts values defined for it enum.def</p> <p>MDEntryPRC. Price Qualifiers.</p> <p>MDEntrySize. BIDSIZE, ASKSIZE, or ORDER_SIZE.</p> <p>payload. Contains any information like Action.UPDATE, Action.ADD, Action.DELETE.</p>

## Symbol List

```
com.apama.rfa.SYMBOL_LIST_RESP(string TRANSPORT, string
  RespType, integer RespTypeNum, string IndicationMask,
  string serviceId, string symbol, string symbolType,
  sequence<com.apama.rfa.SYMBOL_LIST_Entry>
  SLEntries, dictionary<string, string> __payload)
```

Following are the parameter descriptions and default values

Parameter Name	Descriptions
RespType	Response type can be REFRESH_RESP, STATUS_RESP, UPDATE_RESP
RespTypeNum	Response type number can be 0 (SOLICITED), 1(UNSOLICITED)

Parameter Name	Descriptions
IndicationMask	Indication mask can be ATTRIB_INFO_IN_UPDATES, CONFLATION_INFO_IN_UPDATES, CLEAR_CACHE, DO_NOT_CACHE, DO_NOT_CONFLATE, DO_NOT_RIPPLE, REFRESH_COMPLETE
symbolType	symbolType:0 (UNSPECIFIED),1 (RIC),127 (MAX_RESERVED),255(MAX) Defaults to 1
SLEntries	com.apama.rfa.SYMBOL_LIST_Entry(string EntryID, string PROV_Symbol, integer Permission, dictionary<string, string> __payload)  EntryID: Unique key  PROV_Symbol : Original symbol provided by the exchange  Permission : Permission information

## Working with OMM consumer

### Event order for news

1. Configure the OMM Consumer session

```
com.apama.rfa.SessionConfiguration("ReutersRFATransport",
  {"RFASessionName":"ApamaNameSpace::consSession",
   "heartBeatInterval":"30"})
```

2. Send Login request

```
com.apama.rfa.Login("ReutersRFATransport","reuters",{})
```

3. Subscribe for Tick/Depth

```
com.apama.marketdata.SubscribeTick("DIRECT_FEED", "ReutersRFATransport",
  "RTR.L", {})
```

```
com.apama.marketdata.SubscribeDepth("DIRECT_FEED", "ReutersRFATransport",
  "CSCO.BU", {"PublishDepthLevel":"3"})
```

**Note:**

ReutersRFA adapter specific subscriptions can also be done using ReutersRFA_Interface.

SubscribeDirectory, SubscribeDictionary

## Working with OMM Non-Interactive provider

### Constraints

Following are general constraints while publishing data

1. Directory needs to be published first before publishing any market data.
2. Dictionaries should not be published through NON-INTERACTIVE PROVIDER, since it is not supported by RMDS.
3. All messages should be published with unsolicited refresh type.

### Event order

1. Configure the OMM NON-INTERACTIVE PROVIDER session

```
com.apama.rfa.SessionConfiguration("ReutersRFATransport",
  {"applicationType":"OMNIProvider",
   "RFASessionName":"ApamaNameSpace::nonInteractiveProviderSession",
   "heartBeatInterval":"30"})
```

2. Send Login request

```
com.apama.rfa.Login("ReutersRFATransport","reuters",{})
```

3. Publishing Directory

```
com.apama.rfa.DIRECTORY_RESP("ReutersRFATransport","REFRESH_RESP",1,
  "REFRESH_COMPLETE",[1,2,4,8,16,32],
  {
    "DIRECT_FEED":
    com.apama.rfa.Service(
      com.apama.rfa.SDInfoFilterEntry("DIRECT_FEED",[5,6],
        [],["RWFFld","RWEnum"],{"IsSource":"0",
          "SupportsOutOfBandSnapshots":"1","SupportsQoSRange":"0",
          "Vendor":"Reuters"}),
      com.apama.rfa.SDStateFilterEntry(1,1,""),
      com.apama.rfa.SDGroupFilterEntry("","",""),
      com.apama.rfa.SDLoadFilterEntry(0,0,0),
      com.apama.rfa.SDDataFilterEntry(0,""),
      com.apama.rfa.SDLinkFilterEntry(1,0,0,"")
    )
  },
  {
    "DataState":"OK",
    "StatusCode":"NONE",
    "StreamState":"OPEN"
  })
```

#### 4. Publish Data MARKET_PRICE/MARKET_BY_PRICE/MARKET_BY_ORDER/MARKET_MAKER/SYMBOL_LIST

```
com.apama.rfa.MARKET_PRICE_RESP("ReutersRFATransport","REFRESH_RESP",1,
  "DO_NOT_CONFLATE","DIRECT_FEED","ABC","RIC",12.0000,2100,
  {
    "DataState":"OK",
    "StreamState":"OPEN",
    "StatusCode":"NONE",
    "BID":"11.8000",
    "ASK":"12.2000"
  })
com.apama.rfa.MARKET_BY_PRICE_RESP("ReutersRFATransport","REFRESH_RESP",
1,"REFRESH_COMPLETE","DIRECT_FEED","ABC","RIC",
[com.apama.rfa.MKT_BY_PRICE_Entry(
  "DELETE","123","ASK",10.09,100,200,{}))
],
{})
```

## Working With OMM NEWS

Thomson Reuters Machine Readable News is the industry's most advanced service for automating the consumption and systematic analysis of news. It delivers deep historical news archives, ultra-low latency structured news and leading edge news analytics directly to applications. Our current RFA's TICK subscription model can be used for subscribing NEWS too. NEWS subscription supports two RICS,

Our current RFA's TICK subscription model can be used for subscribing NEWS too. NEWS subscription supports two RICS,

- N2_UBMS
- N2_STORY

## Subscribing stories

You can subscribe for stories using RIC 'N2_STORY'. On successful subscription, you will get continuous updates carrying the stories of all news. Every story is composed of several distinct elements, which provide you with a range of information.

## Subscribing headlines

You can subscribe for headlines using RIC 'N2_UBMS' NEWS update received from Reuters will be converted to com.apama.marketdata.TICK event. Below is an example Tick event for Headlines:

```
com.apama.marketdata.Tick("N2_UBMS",-1.0,-1,{"DSPLY_NAME":"2",
  "AREA_ID":"LO","CO_IDS":"ADRJ.J",
  "TAKE_TIME":"12:02:26","PROC_DATE":"25 MAR
2013","ATTRIBTN":"RTRS","STORY_TIME":"11:47:58", "SF_NAME":"LYNX1
","STORY_ID":"nn0003xLtx","RECORDTYPE":"232","PNAC":"nEMS45IKLC",
  "PROD_CODE":"O EMK OIL","TOPIC_CODE":"US GB CRU ENR PROD DRV AGA LEN RTRS REP
ENER AMERS WEU EUROP COM NRG","NAMED_ITEM":"O/ICE",
  "CROSS_REF":"L3N0CH19F","REG_ID1":"2",
```

```
"BCAST_TEXT":"Speculators raise Brent, cut gasoil net longs -ICE",
"DSO_ID":"0", "STORY_TYPE":"S","Market":"ReutersRFATransport",
"PROD_PERM":"511","TAKE_SEQNO":"1", "LANG_IND":"EN",
"SERVICE_NAME":"dIDN_SELECTFEED","STORY_DATE":"25 MAR 2013")
```

## Event injection order

1. Configure the OMM NEWS session

```
com.apama.rfa.SessionConfiguration("ReutersRFATransport",
{"applicationType":"OMMNews",
"RFASessionName":"ApamaNameSpace::consSession"})
```

2. Send Login request

```
com.apama.rfa.Login("ReutersRFATransport","reuters",{})
```

3. Subscribe for headlines/stories

```
com.apama.marketdata.SubscribeTick("IDN_SELECTFEED","ReutersRFATransport",
"N2_UBMS",{})
com.apama.marketdata.SubscribeTick("IDN_SELECTFEED","ReutersRFATransport",
"N2_STORY",{})
```

## RMDS services

A new event 'RMDS_ServiceInfo' is introduced which will give us list of services existing in RMDS and their service state. This gives an idea to end user which service is active and to which service he can subscribe.

```
event RMDS_ServiceInfo{string marketId; string name; string state;
dictionary<string,string> extraParams;}
```

## Enable DEBUG level logging in monitors

You can enable DEBUG logging for RFA monitors using

```
engine_management -r setApplicationLogLevel <LogLevel> <Package_name>
```

For Example:

```
engine_management -r setApplicationLogLevel DEBUG com.apama.rfa
```

## Notice

This is to notify that MARKET_PRICE_RESP event will be changed from next release. Price and quantity fields will added to _payload. Please make necessary changes (Defect #14419).

## Present event

```
MARKET_PRICE_RESP {string TRANSPORT,string RespType,integer
  RespTypeNum,string IndicationMask, string serviceId,string symbol, string
  symbolType,float Price,integer Quantity, dictionary<string,string>
  __payload }
```

## New event

```
event MARKET_PRICE_RESP {string TRANSPORT,string
  RespType,integer RespTypeNum,string IndicationMask, string serviceId,string
  symbol, string symbolType,dictionary<string,string> __payload }
```

## Working with Machine Readable News

Thomson Reuters Machine Readable News (MRN) delivers deep historical news archives, ultra-low latency structured news and leading edge news analytics directly to applications. Currently there are three MRN data feeds available over Elektron out of which RFA adapter supports Real-time NEWS.RFA's TICK subscription model can be used for subscribing NEWS . This feed is sourced from news alerts and stories from Reuters and dozens of third-party news sources. It contains the headline, story body text, and associated metadata available at news publication time.

## Subscribing stories

You can subscribe for the stories using RIC 'MRN_STORY'. On Successful subscription, you will receive continuous updates carrying the stories of all news. Every story is composed of several distinct elements, which provide you with a range of information.

## Samples

### ■ Initial refresh response

```
com.apama.marketdata.Tick("MRN_STORY",-1,-1,{"ACTIV_DATE":"20 MAY 2017",
  "CONTEXT_ID":"0.37520000000000003","DDS_DSO_ID":"8328",
  "FRAG_NUM":"1","MRN_SRC":"HDC_PRD_A","MRN_TYPE":"STORY",
  "MRN_V_MAJ":"2","MRN_V_MIN":"10","Market":"ReutersRFATransport",
  "PROD_PERM":"10001","RDN_EXCHD2":"MRN","RECORDTYPE":"30",
  "SERVICE_NAME":"API_ELEKTRON_EPD_RSSL","SPS_SP_RIC":
  ".[SPSML0L1' &#","TIMACT_MS":"55021640","TOT_SIZE":"0"})
```

### ■ Update message

```
com.apama.marketdata.Tick("MRN_STORY",-1,-1,{"ACTIV_DATE":"17 JUL 2017",
  "FRAG_NUM":"1","GUID":"6c73dad415635bb","MRN_SRC":"HDC_PRD_Awwrwr",
  "MRN_TYPE":"STORY","MRN_V_MAJ":"2","MRN_V_MIN":"10",
  "Market":"ReutersRFATransport","SERVICE_NAME":"DIRECT_FEED",
  "TIMACT_MS":"1500295187995","TOT_SIZE":"1448","altId":"nL4N1IR3DJ",
  "audiences":["NP:C, NP:D, NP:DNP, NP:E, NP:GRO, NP:M, NP:MTL, NP:PSC,
  NP:RNP, NP:SOF, NP:T, NP:Z"],"body":" SHANGHAI, May 25 (Reuters) -
  The world's largest lender,\r\nIndustrial and Commercial Bank of China
  <601398.SS><1398.HK>,\r\nhas signed a 26 billion yuan ($3.79 billion)
```

```

debt-for-equity\r\nswap framework agreement with Shandong Iron & Steel
Group, the\r\nofficial Xinhua news agency reported on Thursday.\r\n
China's lenders are signing deals with struggling,\r\nndebt-laden state
firms to lower their leverage and cut financing\r\ncosts following
instructions from Beijing.\r\n    The deal will help state-owned
Shandong Iron improve its\r\ncapital strength and promote diversification
in its corporate\r\nownership structure, the agency reported. \r\n
This is the fifth such swap signed in the northern province\r\nof Shandong,
the agency added. \r\n    In December, ICBC signed three debt-for-equity
swaps with\r\nShanxi province's highly indebted state-owned coal and
steel\r\nfirms. [nL4N1EL2IE] \r\n    Heavy industries such as coal and
steel have suffered from\r\novercapacity as China relies increasingly
on consumption for\r\neconomic growth. \r\n    The deputy general manager
of Shandong Iron was investigated\r\nby the ruling Communist Party,
according to the party's\r\nanti-graft watchdog. [nL4N1D01XF]
\r\n($1 = 6.9 yuan)\r\n\r\n\r\n\r\n\r\n (Reporting by Beijing monitoring
desk and Engen Tham in\r\nShanghai; Editing by Nick Macfie)\r\n
((engen.tham@thomsonreuters.com; +86-21-6104-1769; Reuters\r\n
Messaging: engen.tham.thomsonreuters.com@reuters.net))\r\n\r\n
Keywords: CHINA ICBC/DEBT\r\n\r\n", "firstCreated": "2017-05-25T09:38:10.000Z",
"headline": "China's ICBC signs $4 bln debt swap with
Shandong Steel - Xinhua<1398.HK><601398.SS><SDONGG.UL>",
"id": "L4N1IR3DJ_1705252Anyj7AIu52AJqjNqkRfGfB0Y67BG5dkxHVLyp",
"instancesOf": "[]", "language": "en", "mimeType": "text/plain",
"provider": "NS:RTRS", "pubStatus": "stat:usable",
"subjects": "[A:2, A:4, A:T, A:U, B:12, B:125, B:126, B:127, B:128,
B:19, B:20, B:22, B:34, B:56, E:1, E:4V, E:6, E:W, G:1, G:3H, G:6,
G:B1, G:K, G:S, M:B6, M:E7, M:N, M:R, M:Z, R:1398.HK, R:601398.SS,
R:SDONGG.UL, U:C, U:FA, U:H, U:Q, U:T, N2:ASIA, N2:ASXPAC, N2:BACT,
N2:BANK, N2:BISV, N2:BMAT, N2:BSVC, N2:CDM, N2:CLIST1, N2:COMPNY,
N2:CN, N2:COA, N2:COM, N2:CORPD, N2:DBT, N2:DBTR, N2:EASIA, N2:EMRG,
N2:FERR, N2:FINS, N2:GEN, N2:HK, N2:INDS, N2:ISU, N2:METL, N2:MEVN,
N2:MIN, N2:MINE, N2:MTAL, N2:NRG, N2:POL, N2:STE, N2:STEE, N2:TRAN,
N2:TRD, P:4295863742, P:5000045132]", "takeSequence": "1", "urgency": "3",
"versionCreated": "2017-05-25T09:38:10.000Z"}

```

## Message fragmentation

Most MRN data items will fit inside a single message, but the message format does allow for a large data item to be split across multiple messages. Tick carrying MRN will be published on receiving all fragments.

## Transport properties

Property	Description
LogNEWS	If enabled, raw OMM data received from MRN will be logged at INFO level.  Default is false.
MRN_OutageTime	All messages related to multi-fragment are expected to be received within 60 seconds, although they should all appear within one second. This property configures time interval

Property	Description
	within which all fragments are supposed to be received. Default is 60 seconds. If fragment is received after the configured time, then the data is discarded and expected to contact SoftwareAG.

## Event injection order

1. Configure the OMM NEWS session

```
com.apama.rfa.SessionConfiguration("ReutersRFATransport",
  {"applicationType":"OMMNews","RFASessionName":"ApamaNameSpace::consSession"})
```

2. Send Login request

```
com.apama.rfa.Login("ReutersRFATransport","reuters",{})
```

3. Subscribe for headlines/stories

```
com.apama.marketdata.SubscribeTick("API_ELEKTRON_EPD_RSSL",
  "ReutersRFATransport","MRN_STORY",{})
```

## Migrating from NFD to MRN

NFD delivery is being shut down this year 2017. You should upgrade to Machine Readable News . The changes include:

- RIC N2_UBMS/N2_STORY has been replaced with MRN_STORY. Old data model has been migrated to new JSON OMM data model.
- FID names in which story/headline/id is received has been changed.

## Integrating RFA with MDA

The Market Data Architecture can be used together with RFA adapter that use legacy components. The Capital Markets Foundation includes a Market Data Bridge service that allows you to implement applications with all the benefits of the Market Data Architecture and still use an adapter that uses legacy interfaces. After the Market Data Bridge has been created, the service handles all the connection/communication protocol for both session management and datastream management. The legacy adapter appears as an Market Data Architecture session.

### ➤ To integrate RFA with Market Data Architecture

1. Start RFA adapter by sending session configuration event where application type is OMMMRN.
2. Send RFA login and ensure that it is successful.

3. Send subscription requests to RFA using Market Data Bridge. In the following EPL, the connect request for RIC MRN_STORY is transformed to RFA subscribe tick request.

```

package com.apama.news.test;
using com.apama.session.SessionHandler;
using com.apama.session.SessionHandlerFactory;
using com.apama.md.NewsSubscriberFactory;
using com.apama.md.NewsSubscriber;
using com.apama.md.client.CurrentNewsInterface;
using com.apama.md.DatastreamConstants;
using com.apama.md.bridge.BridgeConstants;
using com.apama.md.bridge.ConfigMDBridge;
using com.apama.md.bridge.MDBridgeInterface;
using com.apama.md.bridge.MDBridgeExtensionFactory;
using com.apama.md.bridge.MDBridgeExtensionInterface;
using com.apama.md.bridge.NewsHelper;
using com.apama.md.adapter.ConnectDatastream;
using com.apama.md.adapter.DisconnectDatastream;
using com.apama.marketdata.SubscribeTick;
using com.apama.marketdata.UnsubscribeTick;
using com.apama.marketdata.Tick;
event RFA_News_Subscription_Helper {
    ConfigMDBridge bridgeConfig;
    MDBridgeExtensionInterface bridgeExtension;
    action init(string serviceId, string marketId) {
        // Setup the Bridge configuration
        bridgeConfig.serviceId := serviceId;
        bridgeConfig.marketId := marketId;
        bridgeConfig.streamTypes := [DatastreamConstants.DATASTREAMTYPE_NEWS];
        bridgeConfig.extraConfig := {BridgeConstants.CONST_DISABLE_STATUS_SUPPORT:"true"};
        //Add our Bridge extension
        bridgeExtension := (new MDBridgeExtensionFactory).createInterface();
        bridgeExtension.subscribeOthers := subscribeOthers;
        bridgeExtension.unsubscribeOthers := unsubscribeOthers;
    }
    action getMDBridgeConfig() returns ConfigMDBridge { return bridgeConfig; }
    action getMDBridgeExtensionInterface()
        returns MDBridgeExtensionInterface { return bridgeExtension; }
    action subscribeOthers(string serviceId, string marketId,
        ConnectDatastream cds) returns boolean {
        log "Bridge extension for Others subscription for Service Id : "
            + serviceId + " Market Id : " + marketId + " symbol : " + cds.symbol at INFO;
        SubscribeTick subTick := new SubscribeTick;
        subTick.symbol := cds.symbol;
        subTick.serviceId := serviceId;
        subTick.marketId := marketId;
        route subTick;
        NewsHelper newsHelper := new NewsHelper;
        newsHelper.setConnectionId(cds.clientId + 1);
        newsHelper.setSessionId(cds.sessionId);
        newsHelper.setSymbol(cds.symbol);
        Tick tick;
        on all Tick(symbol=cds.symbol):tick
        and not UnsubscribeTick(serviceId=serviceId, marketId=marketId, symbol=cds.symbol)
        {
            if (tick.extraParams.containsKey("body")) then {
                newsHelper.setStory(tick.extraParams["body"]);
                tick.extraParams.remove("body");
            }
        }
    }
}

```

```

    if (tick.extraParams.containsKey("headline")) then {
        newsHelper.setHeadline(tick.extraParams["headline"]);
        tick.extraParams.remove("headline");
    }
    if (tick.extraParams.containsKey("GUID")) then {
        newsHelper.setNewsId(tick.extraParams["GUID"]);
        tick.extraParams.remove("GUID");
    }
    newsHelper.setUpdateType(0);
    newsHelper.setData(tick.extraParams);
    boolean success := newsHelper.publishN();
}
return true;
}
}
action unsubscribeOthers(string serviceId, string marketId,
    DisconnectDatastream dcds) returns boolean {
    log "Bridge extension for Others unsubscription for Service Id : "
        + serviceId + " Market Id : " + marketId + " symbol : " + dcds.symbol at INFO;
    UnsubscribeTick unsubTick := new UnsubscribeTick;
    unsubTick.symbol := dcds.symbol;
    unsubTick.serviceId := serviceId;
    unsubTick.marketId := marketId;
    route unsubTick;
    return true;
}
}
}
monitor News_Subscription_Tester {
    context mainContext := context.current();
    constant string REUTERS_RFA_SERVICE_ID := "DIRECT_FEED";
    constant string REUTERS_RFA_TRANSPORT_NAME := "ReutersRFATransport";
    action onload() {
        RFA_News_Subscription_Helper rfaHelper := new RFA_News_Subscription_Helper;
        rfaHelper.init(REUTERS_RFA_SERVICE_ID, REUTERS_RFA_TRANSPORT_NAME);
        (new MDBBridgeInterface).createMDBBridge( mainContext,
            rfaHelper.getMDBBridgeConfig(),
            rfaHelper.getMDBBridgeExtensionInterface()
        );
        //start Session
        SessionHandler sessionHandler := (new SessionHandlerFactory).connect(
            mainContext,
            rfaHelper.getMDBBridgeConfig().getSessionName(),
            BridgeConstants.CONST_TRANSPORT_NAME
        );
        NewsSubscriber newsSubscriber := (new
        NewsSubscriberFactory).create(sessionHandler);
        integer retVal := newsSubscriber.addSubscribedCallback(onNewsSubscribed);
        newsSubscriber.subscribeCb("N2_STORY", onNewsUpdate);
    }
    action onNewsSubscribed(NewsSubscriber handler,
        com.apama.md.adapter.ConnectionKey connKey) {
        log "Successfully subscribed for symbol: " + connKey.getSymbol() at INFO ;
    }
    action onNewsUpdate(CurrentNewsInterface newsIface) {
        log "Got News for " + newsIface.getSymbol() + "\n News ID : " +
        newsIface.getNewsId()
        + "\n Headline : " + newsIface.getHeadline()
        + "\n Story : " + newsIface.getStory() at INFO;
    }
}
}
}

```

After the subscription is successful, NEWS events appear:

News ID : b601d90010161028

Headline : Changes In Boardroom Announcement -  
Change in Boardroom(AXIATA GROUP BHD)<AXIA.KL>

Story : Company Name : AXIATA GROUP BHD

Stock Code : 6888 (RIC : AXIA.KL)

Change in Boardroom

Name : BELLA ANN ALMEIDA

Designation : DI

Status : RS

Date of Appointment : 2017/05/25

Directorate : Ind & N/Exec

Age : 60

Country :

Nationality : United Kingdom

Working Experience : You are advised to read the entire contents of the announcement or attachment. To read the entire contents of the announcement or attachment, access the Bursa website at <http://www.bursamalaysia.com>

Other Directorship :

Remarks : Resigned as Member of Board Nomination Committee and Board Remuneration

Committee with effect from the date hereof.

You are advised to read the entire contents of the announcement or attachment.

To read the entire contents of the announcement or attachment, access the Bursa website at <http://www.bursamalaysia.com>

Qualifications : MA in Economics, Cambridge University, UKMBA, Imperial College, London, UK

Description : Nil

Interest In the securities : Nil

Composition Of Audit Committee :



# 27 Reuters MAPI Adapter

---

■ Prerequisites .....	318
■ Transport configuration parameters .....	318
■ Service monitor injection order .....	321
■ RFA MAPI event details .....	321
■ Market Data subscriptions .....	324
■ Query symbol list .....	324
■ Custom OMM models .....	325
■ Working with RFA MAPI .....	326
■ Working with Fix-MAPI .....	327
■ Login management .....	329
■ Order management .....	329
■ Password management .....	336
■ Status reporting .....	339
■ RFA and MAPI-FIX .....	340

Reuters MAPI(Matching API) offers two separate interfaces, one dedicated for price discovery and another dedicated for order management. The Reuters Foundation API (RFA) is used for price discovery and an industry standard FIX interface has been introduced for order management and post trading processing. The Matching API solution will ultimately replace our current Auto Quote API. Market data from the Matching service is provided via a Point to Point Server(P2PS) which is accessible via the RFA API. The Matching service requires all users to be first authenticated via Matching FIX interface before they can successfully connect to P2PS. The FIX logon response provides the requesting authentication token which can then be used to login to the P2PS via the RFA. The authorization token is renewed by the Matching Host each time the user logs into the FIX interface.

**Note:**

The adapter is not backward compatible. It will not work with MAPI v1.5.5

## Prerequisites

---

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3
- RFA adapter version better than or equal to 10.3
- RFA library version is 8.0.0.L2 (shipped with the adapter)

## Transport configuration parameters

---

Name	Description
PUBLISH_DEPTH_FROM	Matching service provides depth of book from MARKET BY PRICE data model.  Defaults to MARKET_BY_PRICE
INCLUDE_FIELDS	In OMM message received from server, you get some fields which carry significant information.  Such fields are added as default. Defaults to PRIMACT_1, ACT_TP_1, BID, ASK, BIDSIZE, ASKSIZE, REG_AMOUNT, BIDSIZEIND, ASKSIZEIND, STD_AMOUNT, BOOK_DEPTH, PRICETHOLD, SEQ_NO, ORDBK_VOL, VALUE_TS1, VALUE_DT1, QUOTE_DATE, QUOTIM_MS, VDATE_P1C1, VDATE_P2C1, Symbol, LOTSIZE3, MN_ORD_QT, STD_QTY, RGL_QTY, PIPSIZE, MINQUL, LEFT_DP, RGT_DP, ITDM
QuantityInLots	Boolean variable which tells how the received quantity in TICK/DEPTH to be.

Name	Description
	<p>The term "lot" refers to the quantity (usually &gt; 1) that is the unit of trading for an instrument, that is, trades in this instrument will always be a whole number of lots. It's often equal to the minimum trade size but they are not quite the same thing,</p> <p>e.g. the minimum trade size might be 10 lots. The lot size used by MAPI for all currency pairs is 1e6. Therefore the behaviour of this configuration would be:</p> <ul style="list-style-type: none"> <li>■ If "QuantityInLots" is false, all quantities seen by the application are the actual quantities, e.g. "3000000" or "15000000".</li> <li>■ If "QuantityInLots" is true, all quantities seen by the application are scaled by the lot size for the pair, e.g. "3" or "15".</li> </ul>
PublishUpdatedTickData	<p>Boolean variable when set true reports all fields in the TICK event.</p> <p>Generally, Matching service reports only those fields which are updated from last sent message. Every time a TICK is generated our adapter will send all fields that is, fields which were not updated but send in previous update messages.</p>
FieldDictionary. FileName	<p>It is not necessary that every source use same standard RDM dictionary for parsing OMM data.</p>
EnumDictionary. FileName	<p>The following properties are introduced to add dictionaries defaults:</p> <ul style="list-style-type: none"> <li>■ <code>&lt;property name="FieldDictionary.FileName" value="MAPI_Dictionaries/MAPIFieldDictionary"/&gt;</code></li> <li>■ <code>&lt;property name="EnumDictionary.FileName" value="MAPI_Dictionaries/MAPIEnumTypes"/&gt;</code></li> </ul>
QuickFIX. EnableExpectedNextSeqNumOnLogon	<p>This parameter setting is introduced by the QuickFIX patch to automatically add the NextExpectedMsgSeqNum field and must be enabled to function correctly with the FIX Gateway.</p> <pre>&lt;property name="QuickFIX.EnableExpectedNextSeqNumOnLogon" value="Y" /&gt;</pre>

Name	Description
QuickFIX. ResendApplicationMessagesOnLogon	<p>This parameter is introduced by QuickFIX patch for replaying messages on login. This parameter must be set false.</p> <pre data-bbox="703 390 1378 499">&lt;property name="QuickFIX.ResendApplicationMessagesOnLogon" value="N" /&gt;</pre>
ExcludeTagsFromMessage	<p>This parameter is used to filter field tags from a given message which needs to be sent to exchange. Field tag could be a part of header/body/trailer. This property applies to upstream messages. Also, field tag cannot be a group tag or tag in group. The format is Message type followed by field tag separated with a space as.</p> <p>Example: BE 43 97 50 553 554</p> <p>The above message says that from message type 'BE', the field tags 43,97,50,553, 554 must be removed from the message.</p> <p>Each message group should be separated from another message group with a comma:</p> <p>Example: D 11 55, BE 43 97 50 553 554</p> <pre data-bbox="703 1125 1378 1192">&lt;property name="ExcludeTagsFromMessage" value="D 11 55, BE 43 97 50 553 554" /&gt;</pre> <p>Default value is empty.</p>
MaxSessionLogonAttempts	<p>This parameter configures the maximum number of session login attempts. For MAPI, it is configured as 5. For unlimited attempts, you can configure as -1. Defaults to -1.</p> <pre data-bbox="703 1444 1378 1512">&lt;property name="MaxSessionLogonAttempts" value="5" /&gt;</pre> <p>Once the session reaches maximum limit, you will be notified using com.apama.fix.NotifyUser event followed by stopping connection. User needs to retry by sending com.apama.fix.doLogin event.</p>
AllowEmptyDepths AllowEmptyTicks	<p>Publish empty depths. By default, TICK/DEPTH is not published when there is no FID in the OMM message which matches default FIDs. Still, if you are interested in receiving empty depths then following property will enable it</p>

Name	Description
	<pre>&lt;property name="AllowEmptyDepths" value="true"/&gt;</pre> <pre>&lt;property name="AllowEmptyTicks" value="true"/&gt;</pre> <p>Defaults to false.</p> <p>Example:</p> <pre>com.apama.marketdata.Depth("ABC",   [], [], [], [], [{"ACVOL_1": "100",   "ASK": "10.2000",   "ASK_TIME": "19:12:23",   "BID": "9.8000",   "DIVPAYDATE": "25 DEC 2006",   "Market": "ReutersRFATransport1",   "RDNDISPLAY": "100",   "RDN_EXCHID": "SES",   "SERVICE_NAME": "DIRECT_FEED",   "TRDPRC_1": "10.0000"}])</pre>

## Service monitor injection order

Inject the monitors listed in [“FIX order management session” on page 82](#) and [“Service monitor injection order” on page 296](#), followed by:

1. `${APAMA_HOME}/adapters/monitors/ReutersMAPI_Events.mon`
2. `${APAMA_HOME}/adapters/monitors/FIX_MAPI_Events.mon`
3. `${APAMA_HOME}/adapters/monitors/FIX_MAPI_Support.mon`
4. `${APAMA_HOME}/adapters/monitors/ReutersMAPIManager.mon`

## RFA MAPI event details

### `com.apama.rfa.mapi.SessionConfig(string TRANSPORT, dictionary <string,string> params)`

This initiates each instance of the ReutersRFA adapter, allowing multiple instances within the same Correlator. Also, ReutersRFA can be used with multiple IAF instances.

Following are the parameter descriptions and default values

Parameter name	Description
<code>applicationType</code>	ReutersRFA adapter can be configured as MAPI Defaults to MAPI.
<code>codecName</code>	Name of the Codec

Parameter name	Description
	Defaults to ReutersRFACodec
heartBeatInterval	To keep IAF alive Defaults to 10.0
channel	IAF listens on this channel Defaults to ReutersRFA
RFASessionName	RFA session name as configured using configuration tool Defaults to ApamaNameSpace::Session
FIX_TRANSPORT	The name of FIX transport to request for RFA token
TCID	4 letter credit code for getting screened prices. This will be appended to screened subscriptions.
TokenTimeout	Total time (in seconds) limit to receive token from FIX session. It is a float value. Default 10.0 seconds
checkSubscriptionsWithSymbolList	This should be set true if you want to check subscription symbol exists in the received symbol list from server. Default: True.
SymbolListSequence	MAPI supports instruments of SPOT market and FORWARD market. To retrieve symbol lists from Matching server, you need to raise separate requests.  Using this parameter, client is flexible to request for either of them or both. By default adapter requests for 'FX_SPOT,FX_FORWARD'. In case, <ul style="list-style-type: none"> <li>■ SymbolListSequence="FX_SPOT" and checkSubscriptionsWithSymbolList="true" all subscriptions for FORWARD instruments will be rejected.</li> <li>■ SymbolListSequence="FX_FORWARD" and checkSubscriptionsWithSymbolList="true" all subscriptions for SPOT instruments will be rejected.</li> </ul>

**com.apama.rfa.mapi.Login(string marketId, string userName, dictionary <string,string> params)**

Properties:

Parameter name	Description
marketId	Name of the market (TRANSPORT)
username	Name of the user same as FIX session login user

**com.apama.marketdata.SubscribeTick(string serviceld,string marketId, string symbol, dictionary <string,string> params)**

Parameters	Description
Screened_Price	Two views of the market are provided to subscribers, <ul style="list-style-type: none"> <li>■ Market view (unscreened) contains all of the prices currently available in the system whilst</li> <li>■ Screened view contains the prices that are available to each individual subscriber after taking into account the remaining bi-lateral credit that they have available to the other subscribers.</li> </ul>
TCID	4 letter credit code for getting screened prices. This will appended to screened subscriptions

For Screened subscriptions, you need to add 'Screened_Price' set to true.Defaults to false.Separate subscriptions are required for Screened and Unscreened items. Screened items will use the same item name as the unscreened ones but with subscriber code appended. Subscriber code should be send as an extra parameter 'TCID' in session configuration.

**com.apama.marketdata.SubscribeDepth(string serviceld,string marketId, string symbol, dictionary <string,string> params)**

Parameters	Description
PublishDepthLevel (Optional)	Level of market depth to be published for the current subscription symbol.  Defaults to the value specified at TRANSPORT (PUBLISH_DEPTH_LEVEL)
Screened_Price	Two views of the market are provided to subscribers,

Parameters	Description
	<ul style="list-style-type: none"> <li>Market view (unscreened) contains all of the prices currently available in the system whilst</li> <li>Screened view contains the prices that are available to each individual subscriber after taking into account the remaining bi-lateral credit that they have available to the other subscribers.</li> </ul>
TCID	4 letter credit code for getting screened prices. This will be appended to screened subscriptions

## Market Data subscriptions

Matching host provides separate items for 'Top of Book' and 'Depth of Book' using OMM 'Market by Price'. Use `com.apama.marketdata.SubscribeTick` and `com.apama.marketdata.SubscribeDepth` respectively.

Aggregate "worst" Prices are only provided in Market By Price model type with FIDs BID(22) and ASK(25) and are only available as screened prices updates. Aggregate price represent lowest bid and highest offer away from Best Quote required to obtain Standard Quantity. Standard amount varies based on currency pair. Aggregate price will only display a rate which is within a pre-defined threshold from Best Quote.

## Query symbol list

Client at any point of time after session is established, he can query symbol list using below event for particular market type,

```
event QuerySymbolList {
    string marketId;
    string marketType;
    dictionary<string,string> __payload;
}
```

The marketType here can be 'FX_SPOT' and 'FX_FORWARD_SWAP'

Example:

```
com.apama.rfa.mapi.QuerySymbolList("ReutersRFATransport","FX_SPOT",{})
```

```
com.apama.rfa.mapi.QuerySymbolList("ReutersRFATransport","FX_FORWARD_SWAP",{})
```

The result will be a list of symbols for that particular market type,

```
event ServiceSymbolList {
    string marketId;
    string marketType;
    sequence<string> serviceSymbolList;
}
```

Example:

```
com.apama.rfa.mapi.ServiceSymbolList("ReutersRFATransport", "FX_SPOT", ["AUD=", "PLN=", "EURJPY=", "EURGBP="])
```

## Custom OMM models

MAPI has introduced some user defined models that are very specific and cannot be re-usable by any other product. In order to parse these models MAPI provides its own set of dictionaries. Adapter requests for these models in following cases once login is successful,

- When IAF of base RFA is started for first time
- When trading mode changes from disabled mode to active mode.

Adapter supports following custom models,

### ■ Reference Data

List of instruments for a particular asset class. Along with the names it will also include various properties for each instrument like PIP size, decimal places, Minimum Quote Life (MQL) and much more. These properties are mainly used in order to submit an order via the FIX protocol. End user can listen for below event,

```
com.apama.rfa.mapi.REFERENCE_DATA("ReutersRFATransport", "REFRESH_RESP", 0,
"REFRESH_COMPLETE", "MAPI", "FX_FORWARD_SWAP", "EUR/ISKDOM",
{"RGT_DP": "2", "STD_QTY": "1.0", "RGL_QTY": "10.0", "DataState": "OK",
"PIPSIZE": "1", "Market": "ReutersRFATransport", "StreamState": "OPEN",
"LOTSIZE3": "1.0", "LEFT_DP": "4", "MINQUL": "499", "StatusCode": "NONE",
"__type": "REFERENCE_DATA_RESP", "SERVICE_NAME": "MAPI", "MN_ORD_QT": "1.0"})
```

By default the following properties of an instrument are included in transport property 'IncludeFields'

Symbol,LOTSIZE3,MN_ORD_QT,STD_QTY,RGL_QTY,PIPSIZE,MINQUL,LEFT_DP,RGT_DP

Only these properties will be pushed in com.apama.rfa.mapi.REFERENCE_DATA. If user is interested in more fields, he can add their FID names to this property. Our adapter needs PIP size, MQL which is received in Reference data event. Monitor stores these values for each instrument on receiving it.

### ■ Configuration Data

This model gets information that is not specific to a particular instrument. It contains 1. Asset Classes (ASSCLS) 2. Market Data Groups (MKTGRPMAP) 3. Tenors (TNRMAP) 4. Tip delay range (ITDM)

This data will be logged as well pushed as an event as follows. By default group ITDM will be pushed in com.apama.rfa.mapi.CONFIGURATION_DATA. If user interested in more groups, they can add it to "IncludeFields"

```
com.apama.rfa.mapi.CONFIGURATION_DATA("ReutersRFATransport",
"REFRESH_RESP", 0, "REFRESH_COMPLETE", "MAPI", "PV6", "ITDM", "",
{"2": {"MAXTIPD": "\2000", "MINTIPD": "\1000"}, "DataState": "OK",
"Market": "ReutersRFATransport", "StreamState": "OPEN",
"StatusCode": "NONE", "__type": "CONFIGURATION_DATA_RESP",
"SERVICE_NAME": "MAPI", "1": {"MAXTIPD": "\1000", "MINTIPD": "\250"},
{"MAXTIPD": "\3000", "MINTIPD": "\2000"}})
```

In above event, payload will carry "key":{"FID_NAME":"FID_VALUE".....} In case an iceberg order submitted using tip delay range then user can choose any key of above ranges. Same data will be logged in iaf log as below ,

```
GROUP NAME : ITDM      KEY :1
FIELD_ENTRY MAXTIPD : 1000
FIELD_ENTRY MINTIPD : 250      and so on...
```

## ■ Query Reference/configuration Data

At any point of time after Login successful, if user is interested in Reference data or configuration data he can request using

```
event QueryModelData{
    string TRANSPORT;
    string modelName;
    dictionary<string,string> __payload;
}
```

Field "modelName" can be ReferenceData or ConfigurationData or both or * (to get both models)

User can query reference data per symbol by adding "Symbol" field to payload.

Its value can be comma separated symbol names or *(to get all symbol information).

Sample Event:

```
com.apama.rfa.mapi.QueryModelData("ReutersRFATransport","*",{})
// To get all Reference/Configuration Data
com.apama.rfa.mapi.QueryModelData("ReutersRFATransport","ReferenceData",
{"Symbol":"EUR/ISK DOM"})
// To request reference data of particular symbol.
com.apama.rfa.mapi.QueryModelData("ReutersRFATransport",
"ReferenceData,ConfigurationData",{"Symbol":"*"})
// To request reference data of all symbols.
```

## Working with RFA MAPI

### ➤ Event injection order

#### 1. Configure the MAPI session

Example:

```
com.apama.rfa.mapi.SessionConfiguration("ReutersRFATransport",
{"applicationType":"MAPI","channel":"RFA_MAPPI","RFASessionName":
"testNM::testSession2","logLevel":"DEBUG","FIX_TRANSPORT":"MAPI_TRADING"})
```

#### 2. Send Login request

Example: `com.apama.rfa.mapi.Login("ReutersRFATransport","User1",{})`

#### 3. Subscribe for Tick/Depth

Example:

```
com.apama.marketdata.SubscribeTick("MAPI", "ReutersRFATransport", "AUDJPY=", {})
com.apama.marketdata.SubscribeDepth("MAPI", "ReutersRFATransport", "EURJPY=", {})
```

ServiceID in above subscriptions should always be MAPI.

## Working with Fix-MAPI

The high level steps required in working with Fix-MAPI

1. [“IAF configuration ” on page 327](#)
2. [“Session configuration” on page 328](#)
3. [“Throttle” on page 329](#)

## IAF configuration

When acting as a client to MATCHING FIX GATEWAY(MFG), the IAF should be configured with the MFG specific configuration file, the distribution version of which can be found here:

```
${APAMA_HOME}/adapters/config/FIX-MAPI.xml.dist
```

Note that the requirement for a MAPI-specific configuration file means that an IAF using this configuration file must be used for MAPI FIX sessions only. The QFIX adapter will not interoperate correctly with other FIX servers once it has been configured for MAPI operation. If you need to connect to other FIX servers as well as MAPI, the non-MAPI sessions should be configured in a separate IAF instance. However, MAPI and non-MAPI session can co-exist safely within the same correlator instance.

As well as the standard host, port, sender and target CompID the following parameters must be supplied for MAPI session:

MFG will not allow client to use message header tags 1128 and 1156 to specify a different protocol version than that which was already defined as the default version within initial to the session.

**Tag#1128 (ApplVerID)** - Supported values are 9 (FIX50SP@). All others result in reject.

**Tag#1156 (ApplExtID)** - Valid value ="100". All others result in reject

These are defined in .dist file as:

```
<property name="QuickFIX.DefaultApplVerID" value="9" />
<property name="StaticField.header.*.1156" value="100"/>
```

The logon message will identify the default version of FIX to be carried over the given FIX session in tag 1137 (DefaultApplVerID) and tag 1147 (DefaultApplExtID)

These are defined in .dist file as:

```
<property name="StaticField.body.A.1137" value="9"/>
```

```
<property name="StaticField.body.A.1147" value="100"/>
```

You can assign username and password using,

```
<!-- Username -->
<property name="StaticField.body.A.553" value="@USERNAME@" />
<!-- Password -->
<property name="StaticField.body.A.554" value="@PASSWORD@" />
```

## Session configuration

MAPI FIX Session can be configured with the following configuration parameters:

Parameter	Description
FixChannel	The channel to emit upstream events. It defaults to FIX.
QuantityInLots	This should be set true if external application is sending quantity in millions. <ul style="list-style-type: none"> <li>■ If "QuantityInLots" is false, all quantities sent by the application are the actual quantities, e.g. "3000000" or "15000000".</li> <li>■ If "QuantityInLots" is true, all quantities sent by the application are scaled by the lot size for the pair, example, "3" or "15".</li> </ul>
ThrottlePeriod	Time period (in seconds) to be applied for sending orders. Defaults to 5.0.
ThrottleOrderRate	Number of orders to be sent on throttle period. Defaults to 15
RejectOrderOnReachingThrottleLimit	Orders exceeding the predetermined throttle rate are rejected. Default: True. If ThrottlePeriod = -1 and ThrottleOrderRate = -1 then throttling will not be applied on orders.
MAPI.OrderUpdateUsesTCR	This should be set true if user wants to get Order Updates from Trade Capture Reports rather than Execution Reports. Defaults to False.

Sample configuration event:

```
com.apama.fix.mapi.SessionConfiguration("Connection Name",{"FixChannel":"FIX"})
```

## Throttle

MFG throttling algorithm is based on x orders in y seconds. For example, if ThrottlePeriod = 5 and ThrottleOrderRate = 15, it is possible for a FIX client to send all 15 orders in the first second after which the MFG throttle ensures no further orders are processed until the remaining 4 seconds have elapsed. Orders are throttled using a sliding window.

All messages exceeding the users predetermined throttle rate are queued in monitor until next rolling window if session config parameter "RejectOrderOnReachingThrottleLimit" is set false.

The maximum number of orders and time period used for throttling will be configured by Matching service. This cannot be overridden until further notice from Matching Reuters.

## Login management

### User Logon Request

After sending SessionConfiguration and the session has started, the user must send application login:

```
com.apama.fix.mapi.UserLogonRequest {string TRANSPORT;string requestId;
  string userId;
  string password;
  string senderSubId;
  dictionary <string, string> extraParams;}
```

Example:

```
com.apama.fix.mapi.UserLogonRequest("MAPI_TRADING","", "123",
  "ASP","234",{})
```

### User Logout Request

User can do send a application logout using:

```
com.apama.fix.mapi.UserLogoutRequest {string TRANSPORT;string requestId;
  string userId;string senderSubId;}
```

Example:

```
com.apama.fix.mapi.UserLogoutRequest("MAPI_TRADING Name","",
  "ASP","XYZ")
```

**Note:**

MAPI user session gets logged off but FIX session will not be disconnected.

## Order management

MAPI-FIX supports BID/OFFER or HIT/TAKE

## Sending orders using OMS interface NewOrder (NewOrderSingle)

To populate the Currency(Tag 15) from supplied symbol, you can use configuration parameter "OrderManager.UseCurrencyFromSymbol". Allowed values are "BASE","TERM".

### Orders (Trade Requests) EXTRA PARAMS

Parameter	Type	Description
167	SecurityType	FXSPOT/FXSWAP
59	TimeInForce	3 = HIT/TAKE
231	ContractMultiplier	Always 1000000
854	QtyType	Always 1 (Contracts)
1138	DisplayQty	Required for FX SPOT More Quantity orders.  Not valid for FX SWAP/HIT/TAKE orders
1028	ManualOrderIndicator	Y = True/Yes – represents manual user entry  N = False/No – represents electronic or algorithmic order entry.  Default is Y.

#### Note:

Since ReutersMapi 1.4 upgrade, the tag 1028 (ManualOrderIndicator) has become a mandatory tag in Order Entry message (NewOrderSingle, NewOrderList). If the NewOrder message doesn't have the tag 1028 in its extraparams, the adapter will add it with the default value.

The 35=A (Logon) message simply authenticates the session. This is required to start and maintain a FIX connection but on its own it will not be able to perform any trading on the system. The 35=BE (UserRequest) is a trader login. It is this login that allows you to trade. There can be many trader logins on the same session simultaneously. However, as the Username(554) field is not valid in an order, you have to have a way of differentiating orders made from the different traders. This is performed using the SenderSubID(50). So the value supplied for it on an order must match back to the value used in a UserRequest message. If it doesn't, the order will be rejected. The SenderSubID(50) is added to New order as "Header:50": "xxxxxee" to extra parameters, where "Header:50" says that the tag 50 needs to be added to Header of the respective message.

#### Example:

```
com.apama.oms.NewOrder("4","eur/usd",99.025,"BUY","LIMIT",120,"MAPI-FIX","","",
```

```
"MAPI_TRADING", "", "", {"167": "FXSPOT", "59": "0", "Header:50": "234", "854": "1",
"231": "1000000.0", "15": "eur", "1": "RBCL", "1028": "Y"})
```

For submitting IOC order or yours/mine order or hit/take order, Tag 59 should be set to specify hit/take order

Example:

```
com.apama.oms.NewOrder("4", "eur/usd", 99.025, "BUY", "LIMIT", 120, "MAPI-FIX", "", "",
"MAPI_TRADING", "", "", {"167": "FXSPOT", "59": "3", "Header:50": "234", "854": "1",
"231": "1000000.0", "15": "eur", "1": "RBCL", "1028": "Y"})
```

**Note:**

Jobbing order is no more supported in MAPI 1.6.

**Note:**

Amending an order is not supported.

## More quantity orders

There are two tags, OrderQty(38) and DisplayQty(1138).

Order Quantity(tag 38): Specified in millions. Message is rejected if orderQty is not greater than 0 or is not multiples of one million.

Display Quantity (tag1138): Specified in millions. Conditionally required for More quantity order.

The behavior of orders with more quantity is as follows:

1. A given order's primary quantity must be fully exhausted before the more quantity associated with the same order may be considered for matching
2. An order cannot be entered with 'more' quantity and zero primary quantity
3. The 'more' quantity can be greater than the primary quantity

When the primary quantity of an order is fully consumed via matching, the more quantity will be used until no further orders are eligible to match at the requested price. Once the primary quantity is fully matched the order will be removed from the book regardless of the amount of 'more' quantity that was matched or unmatched.

If DisplayQty exists and (DisplayQty less than OrderQty) then a More Quantity exists and the

FX Matching order quantity = DisplayQty(1138)

FX Matching More Quantity = OrderQty(38) - DisplayQty(1138)

If DisplayQty is not present or if DisplayQty=0 or if DisplayQty=OrderQty then

FX Matching order quantity = OrderQty(38)

FX Matching More Quantity=0

## Price granularity change

Matching had introduced PIP size for specific instrument pairs(for MXN, RUB and ZAR). PIP size for these instruments will be 0.0005 and only 0 or 5 will be allowed values for the 4th decimal place. If the client submits a price for above instruments in a wrong format that is, 4th decimal place is not 0/5 then adapter will round the price.

Example:

Price 12.078123 is changed to 12.0785

price 8.981234 is changed to 8.9815

Price 8.979567 is changed to 8.9800

You receive PIP value in Reference model. PIP value gets updated every time IAF starts or there is a notification from Matching that trading mode changed from disabled to active. User need not worry about updating it. The following interfaces are supported:

```
com.apama.fix.mapi.PipSize {string TRANSPORT;string symbol;string pip;}
```

where symbol is the instrument name and pip is the PIP size.

Client can use this event in case there is a change to current PIP values. You will not be releasing MAPI whenever there is a change to PIP value.

## Minimum Quote Life

The term Minimum Quote Life (MQL) is used to denote a continuous window of time (in milliseconds) for which limit orders are required to be active (that is, available for trading) within the Matching order book before they are eligible to be Held or cancelled by the originating user. Most of the FX Spot instruments traded on the Thomson Reuters Matching service are subject to a Minimum Quote Life (MQL). Order cancel request is affected with MQL. If a cancel request is received before MQL period, the adapter will wait till MQL expires and then is resent.

You will receive MQL value in Reference model. MQL value gets updated every time IAF starts or there is a notification from Matching that trading mode changed from disabled to active. Following interface is still supported:

```
com.apama.fix.mapi.SetMQLEntry {  
    string TRANSPORT;  
    string serviceId;  
    string symbol;  
    float value; // in seconds  
}
```

MQL for each instrument is initialized with the values given in above link. Client can use the above event if there is a change in these values. The value should be in seconds.

## Maximum order size

Prime broker parents have the option to specify a Maximum Order Size (MOS) that can be entered by their Prime Broker Clients (PBC) on Matching. The Prime Broker parent can optionally specify the MOS value for each of the clients on a per Matching Site Id (Credit Code) basis. Maximum Order Size is specified by Prime Brokers in whole units of 1 million USD.

Maximum Order size for a given instrument can be 999m as per MAPI specification. MOS values can be set using,

```
com.apama.fix.mapi.SetMOSEntry {
    string TRANSPORT;
    string serviceId;
    string symbol;
    float value;
}
```

Default MOS value is 999m. As MQL above, you don't get MOS values for each instrument from Matching service. They need to be configure using above event.

## Trade capture report

In MAPI, you receive trade capture report (TCR) for every execution report (ER) which is a partial fill or fill. These reports carry 'MatchStatus' which says if the trade is 'Confirmed' or 'Unconfirmed'. If it is unconfirmed, then the client needs to contact Reuters support. In the monitor, on receiving a TCR, the `com.apama.fix.mapi.TradeCaptureReport` is routed if 'MatchStatus' is 'unconfirmed'. Then a warning is logged saying client needs to contact Reuters support.

### How does our adapter/monitor handle TCR and ER?

In MAPI support monitor, on receiving an execution report with a partial fill or fill, a listener is opened for corresponding TCR with same matching Id. This execution report is sent to base FIX Ordermanager to process it. When you get a TCR, it is converted to `com.apama.fix.mapi.TradeCaptureReport` and is routed out. If you don't receive a TCR within 120 seconds and the order is complete, the thread is killed. If by any chance, TCR arrives after that, it will be logged in system log at WARN level. `com.apama.fix.mapi.TradeCaptureReport` will be routed without OrderId.

Received unmatched TradeCapture report:

```
com.apama.fix.mapi.MAPI_TCR("MAPI_TRADING","FIXT.1.1:FXM->TRFX_MATCHING",
    "136793964","0","0","2","54","142568748","eur/usd",6,6,1.2986,"20121129",
    "20121129-10:30:48.912",
```

Execution report and TCR carry same information except that TCR carries Root party and settlement information. The TCR can be delayed by up to 60 seconds. The execution report is sent quickly though so you are aware of the match and not adversely affected by the delay. This means the execution reports and TCRs could arrive in various different orders and you should not rely on any particular sequence.

In a scenario, you have ER's arriving first followed by TCRs. Use “__APAMA_ORDER_STATE” extra parameter to handle order completion. On receiving final execution report, MAPI support monitor will add an extra parameter ‘__APAMA_ORDER_STATE’:‘final’

This parameter says that the order has reached done state but still need to received TCRs which carry settlement details. On receiving all TCRs, support monitor will route an orderupdate with ‘__APAMA_ORDER_STATE’ as ‘settled’ and kill its thread.

In MAPI, an OrderUpdate with ‘__APAMA_ORDER_STATE’:‘settled’ will be the last after which order can be considered done.

```
event TradeCaptureReport {
  string orderId;
  string TRANSPORT;
  string TradeID;           ->Unique trade token assigned to order
  string TradeTransactionType; -> It will 0
  string MatchStatus;       -> It can be 0(confirmed) or 1(unconfirmed)
  string TradeReportType;   -> It will be 2
  string TrdType;           -> It will be 54 (OTC Trade)
  string TrdMatchID;        -> Match token allocated by Matching engine
  string Symbol;
  float OrderQty;
  float LastQty;
  float LastPx;
  string TradeDate;         -> Date of Trade
  string TransactTime;      -> Time of the trade event associated with this TCR
  string Side;
  string Account;
  boolean AggressorIndicator;
  string OrderID;
  string ClOrdID;
  sequence <com.apama.fix.mapi.RootPartyIds>
    NoRootPartyIds; -> Party information
  sequence <com.apama.fix.mapi.TradeCaptureLimitAmt>
    NoLimitAmts; -> LimitAmount information
  sequence <com.apama.fix.mapi.TradeCaptureReportSettlDetails>
    NoSettlDetails; -> Settlement information
  dictionary<string,string> extraParams;
}
```

### Example:

```
com.apama.fix.mapi.TradeCaptureReport("1","MAPI_TRADING","136793964","0","0","2",
"54","142568747","eur/usd",6,6,1.2986,"20121129","20121129-10:30:48.912",
[com.apama.fix.TradeCaptureReport_RootPartyIds("BANK GENEVA LN","C",13,
[com.apama.fix.TradeCaptureReport_RootPartySubIds("NNLO",25,{}),
com.apama.fix.TradeCaptureReport_RootPartySubIds("USRF",2,{})],{}),
com.apama.fix.TradeCaptureReport_RootPartyIds("BANK GENEVA AB","C",56,
[com.apama.fix.TradeCaptureReport_RootPartySubIds("NNAB",25,{})],{})),
[com.apama.fix.mapi.MAPI_TradeCaptureReport_Side("2","RBCL",true,"88331","1:0:0",
[com.apama.fix.mapi.MAPI_TradeCaptureLimitAmt(0,0,-1,"USD",{})],
[com.apama.fix.mapi.MAPI_TradeCaptureReportSettlDetails("5",
com.apama.fix.mapi.MAPI_TradeCaptureReportSettlPartyIds
("No Instructions Specified","C",27,[],{}),{}),
com.apama.fix.mapi.MAPI_TradeCaptureReportSettlDetails("4",
[com.apama.fix.mapi.MAPI_TradeCaptureReportSettlPartyIds
("RBS London","C",27,
[com.apama.fix.mapi.MAPI_TradeCaptureReportSettlPartySubIds
("123456",16,{})],{})],{})],{})],{}),{"1056":"7791600","120":"USD",
```

```
"15": "EUR", "150": "F", "167": "FXSPOT", "17": "09388ccc-246d-0000-0005-002481817554",
"231": "1000000", "325": "Y", "35": "AE", "423": "20",
"52": "20121129-10:30:48.919", "64": "20121204", "854": "1"}}
```

## Iceberg orders

For an iceberg order, there is a separate tag 847=1001 to indicate Matching Iceberg, and then the NoStrategyParameters(957) repeating group to provide the various properties related to the Iceberg's tip. Iceberg order should also carry a repeating group which specifies tip sizes .

Tip size or primary quantity can be specified in 2 ways:

1. A list of 1 to 3 values (Q1, Q2 & Q3)
2. An inclusive range (Qmin -> Qmax)

Below part is mandatory for an Iceberg and specifies whether the tip is a defined list (Q1, Q2, Q3) or a random range (Qmin-Qmax):

```
StrategyParameterName(958)=T
StrategyParameterValue(960)=D or R <defined or random>
```

If using a defined list (958=T & 960=D above) then below part becomes mandatory, otherwise it is forbidden:

```
StrategyParameterName(958)=Q1
StrategyParameterValue(960)=<qty for 1st tip size>
StrategyParameterName(958)=Q2
StrategyParameterValue(960)=<qty for 2nd tip size>
StrategyParameterName(958)=Q3
StrategyParameterValue(960)=<qty for 3rd tip size>
```

If using a random range (958=T & 960=R above) then this part becomes mandatory, otherwise it is forbidden:

```
StrategyParameterName(958)=Qmin
StrategyParameterValue(960)=<Minimum random tip>
StrategyParameterName(958)=Qmax
StrategyParameterValue(960)=<Maximum random tip>
```

This part is optional, and specifies the pre-defined range from which delays are randomly chosen (if not specified there is no delay):

```
StrategyParameterName(958)=D
StrategyParameterValue(960)=<ID number of the delay range>
that is, this is the key value in com.apama.rfa.mapi.CONFIGURATION_DATA
```

For iceberg orders, following extra parameter is introduced for providing strategy parameters,

```
'Strategyx' where "Strategyx": "key,value" and x=0,1,2,3...
```

Here, key is StrategyParameterName and value is StrategyParameterValue

Example for random range

```
com.apama.oms.NewOrder("2", "eur/usd", 1.1183, "SELL", "LIMIT",
```

```
9000000,"MAPI-FIX","", "", "MAPI_TRADING", "", "",
{"167": "FXSPOT", "59": "0", "Header:50": "234", "854": "1",
"231": "1000000.0", "15": "eur", "1": "RBCL", "847": "1001",
"Strategy1": "T,R", "Strategy0": "Qmin,2", "Strategy2": "Qmax,3"}})
```

### Example for definite list

```
com.apama.oms.NewOrder("2", "eur/usd", 1.1188, "SELL", "LIMIT",
9000000, "MAPI-FIX", "", "", "MAPI_TRADING", "", "",
{"167": "FXSPOT", "59": "0", "Header:50": "234", "854": "1",
"231": "1000000.0", "15": "eur", "1": "RBCL", "847": "1001",
"Strategy1": "T,D", "Strategy0": "Q1,2", "Strategy2": "Q2,3",
"Strategy3": "Q3,4"}})
```

## Locked orders

For an order to be locked, add tag 5007 in the new order. IOC order cannot be locked.

### Example:

```
com.apama.oms.NewOrder("2", "eur/usd", 1.1125, "SELL", "LIMIT", 9000000, "MAPI-FIX", "",
"", "MAPI_TRADING", "", "",
{"167": "FXSPOT", "59": "0", "Header:50": "234", "854": "1", "231": "1000000.0", "15": "eur",
"1": "RBCL", "5007": "Y"}})
```

## Password management

The Matching service requires that all passwords conform to specific password complexity rules. Passwords are not case sensitive. However when changing a Matching password in a FIX session login (35=A) or User level login (35=BE), the new password must comply with the following rules:

- A password may only consist of alphabetic characters or digits
- There must be at least 3 characters that are different at corresponding positions in the old and new passwords
- A password may not contain consecutive instances of the same character
- Minimum length = 6 characters
- Maximum length = 30 characters
- Minimum number of digits= 1
- Maximum number of digits = 5

The very first time you receive credentials from Reuters for both session or trader/user login, you need to reset default password . Here the password given by Reuters will be expired and you need to change it.If the "Password Expired" UserResponse is sent by the MAPI server, then user can request for change of password using:

```
com.apama.fix.mapi.PasswordChangeRequest {string TRANSPORT;
string requestId;
string userId;string senderSubId;
string oldPassword;
```

```
string newPassword;}
```

### Example:

```
com.apama.fix.mapi.PasswordChangeRequest("MAPI_TRADING", "456", "ASP",
    "123", "234", "345")
```

If the password is changed successfully, `PasswordChangeResponse("MAPI_TRADING", "456", "ASP", true, "5", "PasswordChanged");` is received.

For Session login, you have the following transport properties

```
<!-- Username -->
<property name="StaticField.body.A.553" value="@USERNAME@"/>
<!-- Password -->
<property name="StaticField.body.A.554" value="@PASSWORD@"/>
```

The very first time you are trying to login to MFG, you need to send tag 925 carrying new password in message type 'A' as,

```
<!-- New Password -->
<property name="StaticField.body.A.925" value="@NEW_PASSWORD@"/>
```

In case of Session login, if the given password doesn't meet above standards, it will not establish session and keeps throwing error 'connection reset by peer'. It is the client's responsibility to make sure that the password is well formed. Once password is changed, it will never expire. From the next run, you need not add below property in .dist file,

```
<property name="StaticField.body.A.925"
    value="@NEW_PASSWORD@"/>
```

For User/Trader login,

For the very first time, you can add tag 925 with new password and 924=1 to extra params to change password and login successfully. In case of user/trader login, if the given password doesn't meet above standards, you get an error message in User response (35=BF) as 'Not logged in'.

Here, you have two ways,

1. The monitor will automatically generate a well-formed password for you and send user login request. If you were able to login successfully and password is changed, monitor will route 'PasswordChangeResponse'. The external application should handle this event.
2. Else, client can send 'PasswordChangeRequest' on receiving this error. Monitor will send user login request. If you were able to login successfully and password is changed, monitor will route 'PasswordChangeResponse'. The external application should handle this event.

Event injection order:

1. Configure MAPI FIX session using

```
com.apama.fix.mapi.SessionConfiguration("FIXTransport", {"FixChannel": "FIX"})
```

2. Configure MAPI RFA session using

```
com.apama.rfa.mapi.SessionConfiguration("ReutersRFATransport",
    {"applicationType": "MAPI", "channel": "MAPI_RFA",
```

```
"RFASessionName":"testNM::testSession2","logLevel":"DEBUG",
"FIX_TRANSPORT":"FIXTransport")
```

### 3. Subscribe for status message using

```
com.apama.statusreport.SubscribeStatus("MAPI-FIX", " Adapter", "", "FIXTransport")
com.apama.statusreport.SubscribeStatus("MAPI", " Adapter", "", "RFATransport")
```

### 4. If user is interested in trading, he needs to login as trader using

```
com.apama.fix.mapi.UserLogonRequest("FIXTransport", "", "trading_user",
"progress", "APAMA", {})
```

### 5. To login to P2PS server

```
com.apama.rfa.mapi.Login("ReutersRFATransport", "fix_session_user", {})
```

### 6. Subscribe data

Market view:

```
com.apama.marketdata.SubscribeTick("MAPI",
"ReutersRFATransport", "AUD=", {})
com.apama.marketdata.SubscribeDepth("MAPI",
"ReutersRFATransport", "AUD=", {})
```

Screened view:

```
com.apama.marketdata.SubscribeTick("MAPI",
"ReutersRFATransport", "AUD=NNLG0", {"Screened_Price":"true"})
com.apama.marketdata.SubscribeDepth("MAPI",
"ReutersRFATransport", "AUD=NNLG0", {"Screened_Price":"true"})
```

### 7. Place order

```
com.apama.oms.NewOrder("3", "eur/usd", 53.025, "BUY", "LIMIT", 20, "MAPI-FIX", "",
"", "FIXTransport", "", "", {"167":"FXSPOT", "59":"0", "Header:50":"APAMA",
"854":"1", "231":"1000000.0", "15":"eur", "1":"RBCL", "1138":"10", "1028":"Y"})
```

### 8. Cancel order

```
com.apama.oms.CancelOrder("2", "MAPI-FIX", {"Header:50":"APAMA"})
```

### 9. Unsubscription

```
com.apama.marketdata.UnsubscribeDepth("MAPI",
"ReutersRFATransport", "AUD=", {})
```

### 10. Logoff from RFA MAPI

```
com.apama.rfa.mapi.Logout("ReutersRFATransport", "fix_session_user", {})
```

### 11. Trader logoff

```
com.apama.fix.mapi.UserLogoutRequest("FIXTransport", "", "trading_user", "APAMA")
```

### 12. FIX Session logoff

```
com.apama.fix.doLogout("FIXTransport")
```

When FIX session disconnects, RFA session will be automatically logged off.

**Note:**

`com.apama.rfa.mapi.Login` now uses the username of the FIX session login, `com.apama.fix.mapi.UserLogonRequest` carries the trader's username and `senderSubID`. `SenderSubID` identifies the originating user in received application messages. The value must be unique within each client `SendercompID` and captured as part of the administration process. It need be same as User login ID but can be. For post trade processing it is suggested that this value is set to the user long name. In every new order or cancel order, you should have "Header:50" which is `senderSubID` used in `UserLogonRequest`.

## Status reporting

MAPI comprises two adapters 1) Reuters-MAPI 2) FIX-MAPI. There are separate status subscriptions for each of them.

### Reuters-MAPI:

You can subscribe/unsubscribe for status using `com.apama.statusreport.SubscribeStatus("MAPI", "Adapter", "", "RFATransport")`

On successful subscriptions, it will send a `com.apama.statusreport.Status` with `SERVICEID` 'MAPI' and the 'available' field here will be true

- When RFA successfully logs into P2PS server and receive a symbol list from server.

the 'available' field here will be false

- if not logged in
- if you didn't receive symbols list
- if you are logged in but didn't receive symbol list. In such case, once you receive symbol list, a status will be routed with 'available' field set to true

Also, status coming from RFA will be captured, RFA `SERVICENAME` will be changed to MAPI `SERVICENAME` and will be published. Also, maintain reference count of status subscriptions. You can track the same using, "Received status subscription, reference count =2"

You can unsubscribe status using `com.apama.statusreport.UnsubscribeStatus("MAPI", "Adapter", "", "RFATransport")`

### FIX-MAPI:

You can subscribe/unsubscribe for status using `com.apama.statusreport.SubscribeStatus("MAPI-FIX", "Adapter", "", "FIXTransport")` Status coming from FIX will be captured, FIX `SERVICENAME` will be changed to MAPI-FIX `SERVICENAME` and will be published.

You can unsubscribe status using `com.apama.statusreport.UnsubscribeStatus("MAPI-FIX", "Adapter", "", "FIXTransport")` In both cases,

You also accept wildcard subscriptions like, `com.apama.statusreport.SubscribeStatus("", "", "", "")`

## RFA and MAPI-FIX

---

Market data from the Matching service is provided via a Point to Point server (P2PS) which is accessible via the Reuters Foundation API (RFA). The Matching service requires all users to be first authenticated via the Matching FIX interface before they can successfully connect to P2PS. The FIX login response provides the requesting authentication token which can then be used to login to the P2PS via the RFA. The authorization is renewed by the Matching Host each time the user logs into the FIX interface.

RFA MAPI user need to send a login request, `com.apama.rfa.mapi.Login("RFA_MAPI", "User1", {})` the username mentioned here should be same as FIX username used in session login or 35=A message.

MAPI internally requests FIX for authentication token using,  
`com.apama.fix.mapi.RequestRFAToken("ReutersRFATransport", "User1", "567", {})`

The token is routed from FIX using  
`com.apama.fix.mapi.ResponseRFAToken("ReutersRFATransport", "User1", "567", "APAMA-123-456-789", {})`

End user need not worry about `com.apama.fix.mapi.RequestRFAToken` and `com.apama.fix.mapi.ResponseRFAToken`. Monitor will take care of this routing.

**Note:**

In ReutersMAPI.xml.dist file, “@ADAPTERS_JARDIR@” and “@RFA_JARDIR@” are referring to the same location that is, “\$(APAMA_HOME)\Apama\adapters\lib”

```
<java>
  <classpath path="@ADAPTERS_JARDIR@/ReutersRFATransport.jar" />
  <classpath path="@ADAPTERS_JARDIR@/ReutersRFACodec.jar" />
  <classpath path="@RFA_JARDIR@/RFA/8.0.0.L2/rfa.jar" />
  <!-- <jvm-option>-Xmx1024m</jvm-option> -->
</java>
```

# 28 Trading Technologies Adapter

---

■ Trading Technologies Gateway platform .....	342
■ Trading Technologies FIX platform .....	351

## Trading Technologies Gateway platform

---

The Trading Technologies(TT) FIX system deviates from standard FIX 4.2 in a number of ways which have made it necessary to provide a number of extensions to the FIX adapter. This adapter is certified for TT Specification 7.17.x The extensions are into two areas: IAF configuration and service monitors.

### Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

### IAF configuration

When acting as a client to Trading Technologies(TT), the IAF should be configured with the Trading Technologies(TT) specific configuration file, the distribution version of which can be found here:

- `${APAMA_HOME}/adapters/config/FIX-TT.xml.dist`
- `${APAMA_HOME}/adapters/config/FIX-static-TT.xml`

Note that the requirement for a TT-specific configuration file means that an IAF using this configuration file must be used for TT FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for TT operation. If you need to connect to other FIX servers as well as TT, the non-TT sessions should be configured in a separate IAF instance. However, TT and non-TT session can co-exist safely within the same correlator instance.

As well as the standard host, port, sender and target CompID parameters, the following parameters must also be supplied for Trading session:

```
<!-- params supplied by TT -->
<property name="StaticField.body.A.96" value="@TT_PASSWORD@"/>
<property name="StaticField.body.D.1" value="@FIX_ACCOUNT@"/>
<property name="StaticField.body.F.1" value="@FIX_ACCOUNT@"/>
<property name="StaticField.body.G.1" value="@FIX_ACCOUNT@"/>
```

### Service monitor injection order

Inject the monitors listed in [“FIX order management session”](#) on page 82 and [“FIX Legacy Market data session”](#) on page 83, followed by:

- `${APAMA_HOME}/adapters/monitors/FIX_TT_Support.mon`

### Session configuration

TT sessions must be configured with the following configuration parameters:

## Market Data Sessions

- `TT_SERVICEID = TT-FIX`
- `FixChannel = FIX`
- `SubscriptionManager.UpdateDataWithoutEntryId = true`

To handle TT incremental mode market data updates.

Optional configuration parameters

Parameter	Description
SubscriptionManager. AdditionalLMDEntryTypes:"Y Z r s"	To handle additional market data types. For example: For TT MarketData Session <pre>com.apama.fix.SessionConfiguration("TT_MARKET_DATA",   {"FixChannel":"FIX",   "SubscriptionManager.UpdateDataWithoutEntryId":"true",   "TT_SERVICEID":"TT-FIX",   "SubscriptionManager.RepeatingGroupTags":     [146 55 167 200 10455]})</pre>
SubscriptionManager. EnableGatewayStatus:"true"	To enable support for queuing market data requests if the gateway is not available. Once the gateway is up, these requests will be sent.
SubscriptionManager. GatewayStatusDelay	To enable support delaying subscriptions before sending to TT gateway after the gateway is up. Default is 0.0.
SecurityStatusManager. EnableGatewayStatus	To enable support for queuing security status requests if the gateway is not available. Once the gateway is up, these request will be sent. Default is False.
SecurityStatusManager. GatewayStatusDelay	To enable support for delaying subscriptions to TT gateway after the gateway is up. Default is 0.0.
SubscriptionManager. GetSecurityParams:"true"	To update subscription request with security parameters. If this property is false, User is responsible to provide all parameters like security type, symbol in extra parameters defaults to true if User wish to make this property is false, User should provide below extra parameters without fail. <ul style="list-style-type: none"> <li>■ Security type(Tag 167)</li> <li>■ Security Exchange(Tag 207)</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>Exchange-provided product symbol(Tag 55)</li> <li>SecurityID(Tag 48)</li> </ul> <p>For example: Sample Subscription Event</p> <pre>com.apama.marketdata.SubscribeDepth("TT-FIX",   "TT_MARKET_DATA", "ES:MLEG:CME:FUT:ES:2010",   {"167": "FUT", "207": "CME", "48": "00A0CK00ESZ",   "MarketDepth": "1",   "55": "ES:MLEG:CME:FUT:ES:201009-FUT:ES:201103",   "22": "5"})</pre> <p>Default is set to true.</p>
SubscriptionManager. RemoveSubscriptionOnReject:"true"	<p>To enable support for removing/not removing subscriptions when market data request will be rejected. If this property is true then on Market Data Request Reject due to GatewayStatus down the subscription will be removed.</p> <p>Default is set to true.</p>

## Trading Sessions

- OrderManager.GenerateNewStyleOrderIds = true
- FixChannel = FIX

For example: For TT Trading Session

```
com.apama.fix.SessionConfiguration("TT_TRADING",
  {"FixChannel": "FIX", "OrderManager.GenerateNewStyleOrderIds": "true",
  "OrderManager.KillOrdersOnSessionDown": "false"})
```

## Optional configuration parameters

Parameter	Description
OrderManager. GetSecurityParams:"true"	<p>To update Order with security parameters. If this property is false, User is responsible to provide all parameters like security type in extra parameters defaults to true if User wish to make this property is false, User should provide below extra parameters without fail.</p> <ul style="list-style-type: none"> <li>Security type(Tag 167)</li> <li>Security Exchange(Tag 207)</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>Exchange-provided product symbol(Tag 55)</li> <li>SecurityID(Tag 48)</li> </ul> <p>For example: Sample NewOrder Event</p> <pre>com.apama.oms.NewOrder("14","ES",109525,"BUY","LIMIT",1, "TT-FIX","","","TT_TRADING","","",{ "167": "FUT", "207": "CME", "48": "00A0CK00ESZ", "MarketDepth": "1", "55": "ES:MLEG:CME:FUT:ES:201009-FUT:ES:201103", "22": "5", "1": "progress2"})</pre> <p>Default value is set to true.</p>

## Symbol normalization configuration

### Market Data Session

```
com.apama.fix.SessionConfiguration("TT_MARKET_DATA", {"FixChannel": "FIX",
"SubscriptionManager.UpdateDataWithoutEntryId": "true",
"DataManager.SymbolFormationTags": "167 207 200 [146 309 310 313]",
"TT_SERVICEID": "TT-FIX",
"SubscriptionManager.RepeatingGroupTags": "[146 55 167 200 10455]"}))
```

### Trading Session

```
com.apama.fix.SessionConfiguration("TT_TRADING", {"FixChannel": "FIX",
"OrderManager.GenerateNewStyleOrderIds": "true",
"OrderManager.KillOrdersOnSessionDown" : "false",
"DataManager.MappedSecDefListenerKey": "TT_MARKET_DATA"})
```

#### Note:

Value of property `DataManager.MappedSecDefListenerKey` should be Connection of `MarketDataSession` i.e `TT_MARKET_DATA` in above example.

## Notes

The `serviceId` to be used is "TT-FIX". All the requests `Subscription/Order` requests should now be changed to TT-FIX.

User can subscribe for additional `MarketData` types like `Implied`, `Indicative` etc, by providing options in session configuration parameter: `SubscriptionManager.AdditionalLMEntryTypes`

For example, `"SubscriptionManager.AdditionalLMEntryTypes": "Y, Z, r, s"`. Here, additional options are provided for `Implied bid and ask`, `Indicative bid and ask`.

## Security definition request

- Request By Product Type: SecurityType(167)

```
com.apama.fix.InstrumentDataRequest("TT_MARKET_DATA","1","","{"167":"FUT"},[])
```

- Request By Specific Product: Symbol(55)

```
com.apama.fix.InstrumentDataRequest("TT_MARKET_DATA","3","ES",{},{},[])
```

- Request By Entire Market: SecurityExchange(207):

```
com.apama.fix.InstrumentDataRequest("TT_MARKET_DATA","16","","{"207":"CME"},[])
```

- Request By Combination of: Symbol(55), SecurityType(167), SecurityExchange(207)

```
com.apama.fix.InstrumentDataRequest("TT_MARKET_DATA","19","ES,{"167":"FUT","207":"CME"},[])
```

## Subscribe or unsubscribe depth

### Note:

If user make use of the symbol normalization feature he do not need to mention the individual tags for security identification mapped symbol make life easier.

- For specifying a unique Instrument Symbol(tag 55), SecurityID(tag 48), SecurityExchange(tag 207) must be used:

```
com.apama.marketdata.SubscribeDepth("TT-FIX", "TT_MARKET_DATA",
  "Henry Nat Gas",{"207":"ICE_IPE","48":"1460200","22":"5"})
com.apama.marketdata.UnsubscribeDepth("TT-FIX", "TT_MARKET_DATA",
  "Henry Nat Gas",{"207":"ICE_IPE","48":"1460200","22":"5"})
```

By default full book with incremental updates are subscribed.

- For Top of Book, Incremental Refresh use:

```
com.apama.marketdata.SubscribeDepth("TT-FIX", "TT_MARKET_DATA",
  "ES",{"167":"FUT","207":"CME","48":"00A0CK00ESZ","MarketDepth":"1","22":"5"})
```

- For Best N tier Book, Incremental Refresh use: Eg (for 3 tier book):

```
com.apama.marketdata.SubscribeDepth("TT-FIX", "TT_MARKET_DATA",
  "ES",{"167":"FUT","207":"CME","48":"00A0CK00ESZ","MarketDepth":"3","22":"5"})
```

- For Full Book, Full Refresh use:

```
com.apama.marketdata.SubscribeDepth("TT-FIX", "TT_MARKET_DATA",
  "ES",{"167":"FUT","207":"CME","48":"00A0CK00ESZ", "MDUpdateType":"0","22":"5"})
```

## Entering, modifying, or cancelling orders

```
com.apama.oms.NewOrder("14","ES",109525,"BUY","LIMIT",1,"TT-FIX",
  "", "", "TT_TRADING", "", "", {"207":"CME","48":"00A0CK00ESZ","22":"5","1":"progress2"})
com.apama.oms.AmendOrder("14","TT-FIX","ES",109550,"BUY",
```

```
"LIMIT",5,{ "207":"CME", "48":"00A0CK00ESZ", "22":"5", "1":"progress2"})
com.apama.oms.CancelOrder("14", "TT-FIX", { "207":"CME",
"48":"00A0CK00ESZ", "22":"5", "1":"progress2"})
```

## Staged Order

Order staging allows a user to enter and route a selected order to an execution trader or broker for customized handling and execution. TT introduced functionality for staged orders, also known as care orders, in X_TRADER® 7.11. TT's staged order solution allows a user to specify just the contract, the quantity and buy/sell direction, or specify up to a fully qualified order specification including detailed order parameters. The user can also include textual instructions to help the execution trader or broker better understand the user's intentions.

Added following tags to support same:

- Tag 21 (HandlInst): Order handling instructions
- Tag 16104 (TTUserTagData): Data to supply with an order. It corresponds to the X_TRADER User Tag field.
- Tag 16105 (TTOOrderTagData): Data to supply with an order. It corresponds to the X_TRADER Order Tag field.
- Tag 16106 (StagedOrderMsg): Message text associated with the staged order.
- Tag 16111 (RoutingLevel) :Indicator of who can work the staged order. Possible values include:
  - B: Broker order visible to traders with TTORD and exchange trader logins
  - I : Internal order visible only to traders with TTORD logins

## MultiLeg orders

### Note:

To Get MultiLeg Orders to work properly, "Send summary fills" option must be enabled for your account by Trading Technologies.

Orders can be placed using SecurityID(tag 48) of MLEG instrument. For example:

```
com.apama.oms.NewOrder("18", "ES", 35350, "SELL", "LIMIT", 1,
"TT-FIX", "", "", "TT_TRADING", "", "",
{"207":"CME", "48":"0GFES100B0BK00ES40F09AZD67E97E01DA85064", "22":"5", "1":"progress2"})
com.apama.oms.NewOrder("19", "ES", 35350, "BUY", "LIMIT", 1, "TT-FIX",
"", "", "TT_TRADING", "", "",
{"207":"CME", "48":"0GFES100B0BK00ES40F09AZD67E97E01DA85064", "22":"5", "1":"progress2"})
```

Order Updates of the whole MLEG instrument are obtained as:

```
com.apama.oms.OrderUpdate("19", "ES", 35350, "BUY", "LIMIT", 1, false, false,
false, false, false, false, false, "0G22GS006", 1, 0, 1, 35350, 35350, "Filled:SummaryFill",
{"10455":"UD:1V:ST0113911579", "10527":"272979", "10762":"Straddle",
"146":"2", "15":"USD", "167":"MLEG", "18203":"CME", "198":"00005UMV",
"204":"0", "207":"CME", "35":"8", "375":"CME000A", "442":"3", "47":"A",
"48":"0GFES100B0BK00ES40F09AZD67E97E01DA85064", "52":"20100118-10:32:44.342",
```

```

"59": "0", "60": "20100118-10:32:44.439", "6038": "20100118-10:32:44.311",
"77": "0", "Account": "progress2", "ExecID": "1m1swmm17jfgbq", "ExecType": "2",
"Market": "TT_TRADING", "OrdStatus": "2", "SERVICE_NAME": "TT-FIX",
"__APAMA_ORDER_STATE": "settled"})
com.apama.oms.OrderUpdate("18", "ES", 35350, "SELL", "LIMIT", 1, false, false,
false, false, false, false, false, "0G22GS005", 1, 0, 1, 35350, 35350,
"Filled:SummaryFill", {"10455": "UD:1V:ST0113911579", "10527": "272980",
"10762": "Straddle", "146": "2", "15": "USD", "167": "MLEG", "18203": "CME",
"198": "00005UMU", "204": "0", "207": "CME", "35": "8", "375": "CME000A",
"442": "3", "47": "A", "48": "0GFES100B0BK00ES40F09AZD67E97E01DA85064",
"52": "20100118-10:32:44.358", "59": "0", "60": "20100118-10:32:44.439",
"6038": "20100118-10:32:30.420", "77": "0", "Account": "progress2",
"ExecID": "zcq0qrlywz3po", "ExecType": "2", "Market": "TT_TRADING",
"OrdStatus": "2", "SERVICE_NAME": "TT-FIX", "__APAMA_ORDER_STATE": "settled"})

```

Individual leg fills are routed to the application as `com.apama.fix.DisplayEvent`:

```

com.apama.fix.DisplayEvent("TT Spread 'Leg' Execution Report",
[com.apama.fix.DisplayField("TRANSPORT", "TT_TRADING"),
com.apama.fix.DisplayField("SESSION", "FIX.4.2:PROGRESS2->TTDEV140"),
com.apama.fix.DisplayField("OrderID", "0G22GS006"),
com.apama.fix.DisplayField("ClOrdID", "1:1:1263810764.6"),
com.apama.fix.DisplayField("OrigClOrdID", ""),
com.apama.fix.DisplayField("ExecID", "64vvpawvej30"),
com.apama.fix.DisplayField("ExecTransType", "New"),
com.apama.fix.DisplayField("ExecType", "Fill"),
com.apama.fix.DisplayField("OrdStatus", "Fill"),
com.apama.fix.DisplayField("Account", "progress2"),
com.apama.fix.DisplayField("Symbol", "ES"),
com.apama.fix.DisplayField("Side", "BUY"),
com.apama.fix.DisplayField("OrderQty", "1"),
com.apama.fix.DisplayField("OrdType", "LIMIT"),
com.apama.fix.DisplayField("Price", "35350"),
com.apama.fix.DisplayField("LastShares", "1"),
com.apama.fix.DisplayField("LastPx", "35325"),
com.apama.fix.DisplayField("LeavesQty", "0"),
com.apama.fix.DisplayField("CumQty", "1"),
com.apama.fix.DisplayField("AvgPx", "35325"),
com.apama.fix.DisplayField("10455", "ESG0 C0680"),
com.apama.fix.DisplayField("10527", "272981"),
com.apama.fix.DisplayField("NoRelatedSym", "0"),
com.apama.fix.DisplayField("Currency", "USD"),
com.apama.fix.DisplayField("SecurityType", "OPT"),
com.apama.fix.DisplayField("18203", "CME"),
com.apama.fix.DisplayField("SecondaryOrderID", "00005UMV"),
com.apama.fix.DisplayField("MaturityMonthYear", "201002"),
com.apama.fix.DisplayField("PutOrCall", "1"),
com.apama.fix.DisplayField("StrikePrice", "68000"),
com.apama.fix.DisplayField("CustomerOrFirm", "0"),
com.apama.fix.DisplayField("SecurityExchange", "CME"),
com.apama.fix.DisplayField("MsgType", "8"),
com.apama.fix.DisplayField("MultiLegReportingType", "2"),
com.apama.fix.DisplayField("Rule80A", "A"),
com.apama.fix.DisplayField("SecurityID", "00B0BK10ES40F09AZ"),
com.apama.fix.DisplayField("SendingTime", "20100118-10:32:44.358"),
com.apama.fix.DisplayField("Text", "Leg Fill"),
com.apama.fix.DisplayField("TimeInForce", "0"),
com.apama.fix.DisplayField("TransactTime", "20100118-10:32:44.439"),
com.apama.fix.DisplayField("6038", "20100118-10:32:44.311"),
com.apama.fix.DisplayField("OpenClose", "0")], [])

```

```

com.apama.fix.DisplayEvent("TT Spread 'Leg' Execution Report",
[com.apama.fix.DisplayField("TRANSPORT","TT_TRADING"),
com.apama.fix.DisplayField("SESSION","FIX.4.2:PROGRESS2->TTDEV140"),
com.apama.fix.DisplayField("OrderID","0G22GS005"),
com.apama.fix.DisplayField("ClOrdID","1:0:1263810750.6"),
com.apama.fix.DisplayField("OrigClOrdID",""),
com.apama.fix.DisplayField("ExecID","quhbhg1e5tzl6"),
com.apama.fix.DisplayField("ExecTransType","New"),
com.apama.fix.DisplayField("ExecType","Fill"),
com.apama.fix.DisplayField("OrdStatus","Fill"),
com.apama.fix.DisplayField("Account","progress2"),
com.apama.fix.DisplayField("Symbol","ES"),
com.apama.fix.DisplayField("Side","SELL"),
com.apama.fix.DisplayField("OrderQty","1"),
com.apama.fix.DisplayField("OrdType","LIMIT"),
com.apama.fix.DisplayField("Price","35350"),
com.apama.fix.DisplayField("LastShares","1"),
com.apama.fix.DisplayField("LastPx","35325"),
com.apama.fix.DisplayField("LeavesQty","0"),
com.apama.fi

```

## Using alternative security ID method of identifying securities

To use the alternative security id feature the following new session configuration properties need to be set:

Property	Description
SubscriptionManager.UseAltSecurityId	To use this feature for market data request.
OrderManagerA.UseAltSecurityId	To use this feature for order management events.

These parameters should be set to the FIX tag for alternative security id, for TT, parameters are set to "10455". User should put the alternative security id in the "symbol" field of a market data request or order management event, and have this mapped to the right tag on the FIX message (and vice-versa for downstream messages).

Example market data event:

```

com.apama.marketdata.SubscribeDepth("TT-FIX", "TT_MARKET_DATA", "Feb10-Mar10",
{"207":"ICE_IPE","48":"1460200","55":"Henry Nat Gas","167":"NRG","22":"5"})
com.apama.marketdata.SubscribeDepth("TT-FIX", "TT_MARKET_DATA",
"Feb10-Mar10",{"207":"ICE_IPE","48":"1460200","Symbol":"Henry Nat
Gas","167":"NRG","22":"5"})

```

Example OMS event:

```

com.apama.oms.NewOrder("1040","Feb10-Mar10",5.031,"SELL","LIMIT",10,
"TT-FIX","","","TT_TRADING","","",{ "207":"ICE_IPE","55":"Henry Nat Gas",
"167":"NRG","1":"TT 000","440":"649","16102":"009","22":"5","1":"progress2"})
com.apama.oms.NewOrder("1040","Feb10-Mar10",5.031,"SELL",
"LIMIT",10,"TT-FIX","","","TT_TRADING","","",{ "207":"ICE_IPE",
"Symbol":"Henry Nat Gas","167":"NRG","1":"TT 000","440":"649",

```

```
"16102":"009", "22":"5", "1":"progress2"}})
```

After these are processed by the respective monitors the SYMBOL and Alternate Security ID are exchanged and Alternate Security ID will be assigned to 10455 tag.

## Market Data

Following is the sample market data that user receives when the session is configured with additional market data types like "A p q r s t". Here, subscription is made with additional market data types Imbalance, Indicative open, close, bid, ask and settlement.

### Session configuration

```
com.apama.fix.SessionConfiguration("TT_MARKET_DATA", {"FixChannel":"FIX",
  "SubscriptionManager.UpdateDataWithoutEntryId":"true",
  "TT_SERVICEID":"TT-FIX", "SubscriptionManager.AdditionalMDEntryTypes":"A p q r s
t",
  "SubscriptionManager.RepeatingGroupTags":"[146 55 167 200 10455]"}))
```

### Depth Event

```
com.apama.marketdata.Depth("ES", [550,450,435], [410], [480,-1,-1], [2,20,100], [1],
{"167":"MLEG", "207":"CME", "263":"1", "35":"X", "48":"00CESDERFEK00ESZ110ASDEDFEZ1",
"52":"20100601-14:38:27.327", "IMBALANCE_10455":"ESM0-SU0", "IMBALANCE_15":"USD",
"IMBALANCE_167":"MLEG", "IMBALANCE_207":"CME",
"IMBALANCE_48":"00CES200A0FK00ESZ1100A0IK00ESZ1",
"IMBALANCE_MDEntryPx":"350", "IMBALANCE_MDEntrySize":"2",
"INDICATIVE_CLOSE_10455":"ESM0-ESU0", "INDICATIVE_CLOSE_15":"USD",
"INDICATIVE_CLOSE_167":"MLEG", "INDICATIVE_CLOSE_207":"CME",
"INDICATIVE_CLOSE_48":"00CES200A0FK00ESZ1100A0IK00ESZ1",
"INDICATIVE_CLOSE_MDEntryPx":"350", "INDICATIVE_CLOSE_MDEntrySize":"2",
"INDICATIVE_OPEN_10455":"ESM0-ESU0", "INDICATIVE_OPEN_15":"USD",
"INDICATIVE_OPEN_167":"MLEG", "INDICATIVE_OPEN_207":"CME",
"INDICATIVE_OPEN_48":"00CES200A0FK00ESZ1100A0IK00ESZ1",
"INDICATIVE_OPEN_MDEntryPx":"350", "INDICATIVE_OPEN_MDEntrySize":"2",
"INDICATIVE_SETTLEMENT_10455":"ESM0-ESU0",
"INDICATIVE_SETTLEMENT_15":"USD", "INDICATIVE_SETTLEMENT_167":"MLEG",
"INDICATIVE_SETTLEMENT_207":"CME",
"INDICATIVE_SETTLEMENT_48":"00CES200A0FK00ESZ1100A0IK00ESZ1",
"INDICATIVE_SETTLEMENT_MDEntryPx":"350",
"INDICATIVE_SETTLEMENT_MDEntrySize":"2", "INDICATIVE_ASK_0_MDEntryPx":"325",
"INDICATIVE_ASK_0_MDEntrySize":"2",
"INDICATIVE_BID_0_MDEntryPx":"440",
"INDICATIVE_BID_0_MDEntrySize":"10"}))
```

Extra parameters with following prefix indicates respective market data entries:

- IMBALANCE_#TAG
- INDICATIVE_OPEN_#TAG
- INDICATIVE_CLOSE_#TAG
- INDICATIVE_SETTLEMENT_#TAG

- INDICATIVE_BID_#LEVEL_#TAG
- INDICATIVE_ASK_#LEVEL_#TAG

where,

- #LEVEL: represents the market data level for the entry.
- #TAG: represents the respective tag under market data type indicated in prefix.

## Trading Technologies FIX platform

### Prerequisites

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

### IAF configuration

When acting as a client to Trading Technologies(TT), the IAF should be configured with the Trading Technologies(TT) specific configuration file, the distribution version of which can be found here:

- `${APAMA_HOME}/adapters/config/FIX-TT-MDA.xml.dist`
- `${APAMA_HOME}/adapters/config/FIX-static-TT-groups.xml.xml`
- `${APAMA_HOME}/adapters/config/FIXTransport-BaseFIXLibrary-TT.xml`

Note that the requirement for a TT-specific configuration file means that an IAF adapter using this configuration file must be used for TT FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for TT operation. If you need to connect to other FIX servers as well as TT, the non-TT sessions should be configured in a separate IAF instance. However, TT and non-TT session can coexist safely within the same correlator instance.

As well as the standard host, port, sender and target CompID parameters, the following parameters must also be supplied for Trading session:

```
<!-- params supplied by TT -->
<property name="StaticField.body.A.96" value="@TT_PASSWORD@"/>
<property name="StaticField.body.*.1" value="@FIX_ACCOUNT@"/>
<property name="StaticField.header.*.116" value="@ONBEHALFOF_SUBID@"/>
```

### Service monitor injection order

Inject the EPL files as listed in [“FIX MDA session” on page 82](#), followed by:

- `${APAMA_HOME}/adapters/monitors/FIX_TT_Events.mon`

## Subscription management

Trading Technologies FIX adapter uses the following streams of CMF MDA:

- MBP
- BBA
- Trade

You can only subscribe by SecurityID (Tag 48).

To subscribe Trade Stream, you must add AggregatedBook to Y to control parameters.

## Order management

Order sessions must be configured with the following configuration parameters:

```
"OrderManager.KillOrdersOnSessionDown" : "false",
"OrderManager.IgnoreStatusOnOrderCancelReject": "true",
"OrderManager.CopyExecTypeToOrdStatus": "true"
```

For example,

```
com.apama.fix.SessionConfiguration("TT_FIX_TRADING",
{"FixChannel": "FIX_CHANNEL", "LogLevel": "DEBUG",
"OrderManager.GenerateNewStyleOrderIds": "true",
"OrderManager.KillOrdersOnSessionDown" : "false",
"OrderManager.IgnoreStatusOnOrderCancelReject": "true",
"OrderManager.CopyExecTypeToOrdStatus": "true"})
```

Party information must be in the "Party0": "1,D,10,2376:22" format in the extra parameters of new order.

For example,

```
com.apama.oms.NewOrder("5", "ES", 272325, "SELL", "LIMIT", 1,
"FIX", "", "", "TT_FIX_TRADING", "", "", {"1": "softwareag", "1028": "N",
"1385": "1", "16106": "blabla", "16111": "I", "1724": "5", "18": "2",
"207": "CME", "21": "3", "22": "96",
"2593": "[com.apama.fix.tt.OrderAttributes(2, \"Y\", {})]",
"48": "5505520756192796493", "528": "I", "59": "1", "77": "0",
"Party0": "1,D,10,2376:22"})
```

### Note:

You must wait for `com.apama.fix.News` to ensure that the recovery is completed, and then place the orders.

## Connectivity plug-in configuration

The YAML configuration file `FIX-TT-Connectivity.yaml` describes the chain and its components. The YAML file contains place holders for the necessary configuration parameters to connect to the Currenex ESP service. The connection related configuration is specified in the `fixSession` section on the `FixConnectivityTransport` instance. It requires the custom data dictionary `FIX42-`

TT-MDA.xml. This file must be referenced by the element `DataDictionary` of the `fixSession` section. You must update the `fixSession` section with the relevant values.

## Service monitor injection order

Inject the EPL files as listed in [“FIX MDA session” on page 82](#) and [“Connectivity plug-in order management session” on page 84](#), followed by:

- `${APAMA_HOME}/adapters/monitors/FIX_TT_Events.mon`

## Subscription management

Trading Technologies FIX adapter uses the following streams of CMF MDA:

- MBP
- BBA

You can only subscribe by SecurityID (Tag 48).

## Order management

Order sessions must be configured with the following configuration parameters:

```
"OrderManager.IgnoreStatusOnOrderCancelReject":"true",
"OrderManager.CopyExecTypeToOrdStatus":"true",
"OrderManager.KillOrdersOnSessionDown" : "false"
```

For example,

```
com.apama.fix.SessionConfiguration("FIX_TT_TRADE_Connectivity",
{"FixChannel":"FIX_TT_TRADE_Connectivity","CONNECTIVITY_PLUGIN":"true",
"OrderManager.IgnoreStatusOnOrderCancelReject":"true",
"OrderManager.CopyExecTypeToOrdStatus":"true",
"OrderManager.KillOrdersOnSessionDown" : "false"})
```

### Note:

The order attributes group is not supported in the connectivity based adapter.

You must wait for `com.apama.fix.News` to ensure that the recovery is completed, and then place the orders.

For example,

```
com.apama.oms.NewOrder("9","ES",272475,"SELL","LIMIT",1,
"FIX","","","FIX_TT_TRADE_Connectivity","","",
{"207":"CME","48":"5505520756192796493","22":"96","1724":"5","21":"3",
"16106":"blabla","16111":"I","528":"I","1028":"N","59":"1","77":"0",
"18":"2","1":"softwareag","1385":"1","Party0":"1,D,10,2376:22"})
```



# 29 UBS Fx2B FIX Adapter

---

■ Prerequisites .....	356
■ IAF configuration .....	356
■ Service monitor extensions .....	356
■ Service monitor injection order .....	357
■ Service ID configuration .....	357
■ Secure Tunnel .....	357
■ Session configuration .....	357
■ UBS subscriptions and unsubscriptions .....	358
■ UBS order management .....	359

The UBS FIX system deviates from standard FIX 4.3 in a number of ways which have made it necessary to provide a number of extensions to the FIX adapter. The extensions are into two areas: IAF configuration and service monitors. The adapter is certified against v1.16 of the UBS Fx2B Liquidity API.

## Prerequisites

---

- Apama version equal or better than 10.3
- CMF version equal or better than 10.3
- FIX adapter version equal or better than 10.3

## IAF configuration

---

When acting as a client to UBS, the IAF should be configured with the UBS specific configuration file, the distribution version of which can be found here:

```
${APAMA_HOME}/adapters/config/FIX-UBS.xml.dist
```

### Note:

The requirement for a UBS-specific configuration file means that an IAF using this configuration file must be used for UBS FIX sessions only. The FIX adapter will not interoperate correctly with other FIX servers once it has been configured for UBS operation. If you need to connect to other FIX servers as well as UBS, the non-UBS sessions should be configured in a separate IAF instance. However, UBS and non-UBS session can co-exist safely within the same correlator instance.

As well as the standard host, port, sender and target CompID parameters, the following parameters must also be supplied for the trading session:

```
<!-- Username/Password (supplied by UBS) -->  
<property name="StaticField.body.A.553" value="@UBS_USERNAME@"/>  
<property name="StaticField.body.A.554" value="@UBS_PASSWORD@"/>  
<!-- Party ID (supplied by UBS) -->  
<property name="StaticField.body.D.448" value="@PARTY_ID@"/>  
<!-- Party ID Role (supplied by UBS) -->  
<property name="StaticField.body.D.452" value="@PARTY_ID_ROLE@"/>
```

## Service monitor extensions

---

The `FIX_UBS_Support.mon` file provides message handling for quote subscriptions and unsubscriptions.

Therefore, the following EPL file must be injected into the correlator, after the standard FIX adapter service monitors:

```
FIX_UBS_Support.mon
```

## Service monitor injection order

Inject the monitors listed in “[FIX order management session](#)” on page 82 and “[FIX Legacy Market data session](#)” on page 83, followed by:

1. `${APAMA_HOME}/adapters/monitors/FIX_UBS_Support.mon`

## Service ID configuration

The Service ID to be used is UBS-FIX.

## Secure Tunnel

UBS connection requires secure tunnel to be running. The IAF should connect to the stunnel host (for example, localhost).

Use the following stunnel configuration file:

```
; File Name: stunnel_000.conf
; Title: Stunnel configuration for UBS IB FX FIX B2B
; Environment: 000
; Socket parameters tuning
socket = l:TCP_NODELAY=1
socket = r:TCP_NODELAY=1
socket = l:SO_KEEPALIVE=1
socket = r:SO_KEEPALIVE=1
; Security level
verify = 2
; Path to key and certificate files
cert = %SSL_CERTIFICATES_DIRECTORY%\btobxpte0229.pem
key = %SSL_CERTIFICATES_DIRECTORY%\btobxpte0229.key
CAfile = %SSL_CERTIFICATES_DIRECTORY%\ca_pte.pem
; Uncomment for troubleshooting purposes
debug = 7
; Log file path
output = %STUNNEL_LOG_DIRECTORY%\stunnel.log
client = yes
[UBS_FXFIXB2B_PTE]
accept = 2500
connect = fxfixb2bpte1.ibb.ubstest.com:2500
```

## Session configuration

```
com.apama.fix.SessionConfiguration("UBS_MARKET_DATA",
  {"FixChannel":"UBS_FIX", "SERVICEID":"UBS-FIX",
  "SubscriptionManager.MarketDataFullRefresh":"true",
  "SubscriptionManager.ReuseRequestID":"true",
  "SubscriptionManager.IncludeTimeInRequestID":"true"})
com.apama.fix.SessionConfiguration("UBS_TRADING",
  {"FixChannel":"UBS_FIX", "SERVICEID":"UBS-FIX",
  "OrderManager.GenerateNewStyleOrderIds":"true"})
```

## SessionConfig parameters

Parameter	Description
SERVICEID	Service ID to be used is UBS-FIX. This parameter configures the UBS specific monitors to handle UBS specific events or special handling of the events.
SubscriptionManager. MarketDataFullRefresh	Whether to request full refreshes or snapshots rather than snapshots and updates.
SubscriptionManager. ReuseRequestID	Whether to reuse the requestID when resubscribing or to generate a new one each time.
SubscriptionManager. IncludeTimeInRequestID	Whether to include timestamp in the requestID for QuoteRequest.
OrderManager. GenerateNewStyleOrderIds	Generates the ID with the complete current (full length) time for uniqueness.
SubscriptionManager. SupportZeroQuantities	Overwrites old quantity, if the new quantity is zero.

### Note:

Test symbols enabled for a test connection EUR/USD (works for both Spot and Forwards).

## UBS subscriptions and unsubscriptions

### Note:

No tick data support.

To identify a Forward include the appropriate extra parameters in the subscribe depth request (for example, CFICode=FFCPN0;FutSettDate=20091201).

## Market data subscriptions

### Quote subscriptions

```
com.apama.marketdata.SubscribeDepth("UBS-FIX",
  "UBS_MARKET_DATA",
  "EUR/USD", {"CFICode": "FFCPN0", "FutSettDate": "20091201"})
```

### Sample subscription

UBS Requires Party Information to be provided while subscribing for Quotes.

Parties info should be provided in extraparams of subscribe quote as: "Party0": "X,Y,Z".

Here, PartyID(448)=X, PartyIDSource(447)=Y, PartyRole(452)=Z.

Information about multiple parties can also be provided as "Partyx": "PartyID, PartyIDSource, PartyRole", where x starts with '0' and increments based on the number of parties.

### Sample subscription:

*SubscribeDepth* event is sent for quote request with Quote=Y in the extra parameter.

```
FORWARD Quote
com.apama.marketdata.SubscribeDepth("UBS-FIX", "UBS_MARKET_DATA",
  "EUR/USD", {"Quote": "Y", "CFICode": "FFCPNO", "Currency": "EUR",
  "OrderQty": "100", "FutSettDate": "20091201", "Party0": "X,Y,Z"})
```

```
SPOT Quote
com.apama.marketdata.SubscribeDepth("UBS-FIX", "UBS_MARKET_DATA",
  "EUR/USD", {"Quote": "Y", "CFICode": "RCSXXX", "Currency": "EUR",
  "OrderQty": "100", "FutSettDate": "20091201", "Party0": "X,Y,Z"})
```

```
NDF Quote
com.apama.marketdata.SubscribeDepth("UBS-FIX", "UBS_MARKET_DATA",
  "EUR/USD", {"Quote": "Y", "CFICode": "FFCENNO", "Currency": "EUR",
  "OrderQty": "100", "FutSettDate": "20091201", "Party0": "X,Y,Z"})
```

```
SWAP Quote
com.apama.marketdata.SubscribeDepth("UBS-FIX", "UBS_MARKET_DATA",
  "EUR/USD", {"Quote": "Y", "CFICode": "FFCPNW", "Currency": "EUR",
  "OrderQty": "100", "FutSettDate": "20091201", "Party0": "X,Y,Z"})
```

### Note:

*CFICode(tag 461)* is required in the *SubscribeDepth(quotes)*.

## UBS order management

UBS Requires Party Information to be provided while placing orders.

Parties info should be provided in extraparams of subscribe quote as: "Party0": "X,Y,Z".

Here, PartyID(448)=X, PartyIDSource(447)=Y, PartyRole(452)=Z.

Information about multiple parties can also be provided as "Partyx": "PartyID, PartyIDSource, PartyRole", where x starts with '0' and increments based on the number of parties.

### Sending orders using OMS interface NewOrder

```
com.apama.oms.NewOrder("order1", "EUR/USD", 1.457852, "BUY", "LIMIT",
  2000000, "UBS-FIX", "", "", "UBS_TRADING", "", "", {"CFICode": "FFCPNO",
  "OrderQty": "2000000", "FutSettDate": "20091201", "TimeInForce": "4", "Party0": "X,Y,Z"})
```

### Sending orders (PREVIOUSLY QUOTED) using OMS interface NewOrder

```
com.apama.oms.NewOrder("pq1", "EUR/USD", 1.496914, "BUY", "PREVIOUSLY QUOTED",
  100, "UBS-FIX", "", "", "UBS_TRADING", "", "", {"Quote": "Y",
  "QuoteID": "rfq3043400039111143608658", "CFICode": "FFCPNO",
  "OrderQty": "100", "FutSettDate": "20091201", "TimeInForce": "4", "Party0": "X,Y,Z"})
```

