Capital Markets Foundation Release Notes

Version 9.12

October 2016

This document applies to Apama Capital Markets Foundation Version 9.12 and to all subsequent releases.

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

Document ID: PAF-RN-912-20161018

Table of Contents

1 What's New in Capital Markets Foundation 9.12

Apama Capital Markets Foundation 9.12 is the successor of Apama Capital Markets Foundation 9.10. There is no version 9.11.

Apama Capital Markets Foundation 9.12 includes new features, enhancements, and changes as described in the following topics.

Miscellaneous enhancements and changes in 9.12

- The DataViewManager now supports the `decimal` field type.
- The following actions in each of the MDA subscriber factories are now static and no longer require an instance of the event to be called:

  ```
  create()
  subscribe()
  subscribeCb()
  multipleSubscribe()
  multipleSubscribeCb()
  ```

  The MDA samples have been updated to show how this can be achieved.

2 What's New in Capital Markets Foundation 9.10

Apama Capital Markets Foundation 9.10 includes new features, enhancements, and changes as described in the following topics.

Miscellaneous enhancements and changes in 9.10

The Forward to Spot Converter (FTSC) now offers Orderbook as an available datastream connection type, along with Depth, BBA and Trade.

The version number has been removed from the file `ACMEDashboardExtensions9.9.jar`. The new name is `ACMEDashboardExtensions.jar`. Due to this change, you may need to update your build scripts and automated tests.

3 What's New in Capital Markets Foundation 9.9

Apama Capital Markets Foundation 9.9 is the successor of Apama Capital Markets Foundation 5.3.

Installation changes in 9.9

As of version 9.9, Apama Capital Markets Foundation is installed using the Software AG Installer, and official fixes are installed using the Software AG Update Manager. See the new *Installing Apama* guide for more information.

In the Software AG Installer, you can select the following options from the product selection tree:

- **Apama > Capital Markets Foundation**. Installs all the components of Apama Capital Markets Foundation, except those for the Software AG Designer integration.

  > **Note:** On UNIX, this is the only installation option. All other options listed here are only available on Windows.

- **Designer > Apama > Capital Markets Application Development**. Installs the Designer integration components of Apama Capital Markets Foundation.

You can quickly select all Apama and Apama Capital Markets Foundation components by clicking **Typical Installations** which is shown next to the product selection tree and then selecting **Apama**.

Due to the above-mentioned changes, all Windows start menu shortcuts have changed. The shortcuts for Apama Capital Markets Foundation are now available from **All Programs > Software AG > Tools > Apama Capital Markets *n.n***. Note that **Software AG** is the default group name that you can change during the installation.

4 What's New in Capital Markets Foundation 5.3

Apama Platform for Capital Markets Release 5.3 adds the following new features and enhancements to the Capital Markets Foundation.

Performance improvements due to extra parameter handling changes

To significantly improve throughput and latency, the event protocol previously used to handle extra parameters has been redesigned to separate schema data from values. Values of extra parameters are now contained in the market data events, which removes the need for a second event.

For book based market data (`Orderbook`, `Depth`, `Quotebook` and `AggregatedBook`), the schema (type and description data) for any extra parameters is sent upon the initial connection. Each level or entry now has its own set of extra parameter values. This removes the need for extra parameters to be encoded into a top-level extra parameter value. Also, the schema can now be delta updated, which removes the need for the large snapshot events that were sent in previous releases.

Two new EPL interfaces provide access to extra parameters:

- `com.apama.md.client.EPSchemaInterface` — Each market data interface now provides the `getEPSchemaInterface()` action, which you can call to access the schema for any extra parameters.
- `com.apama.md.client.EPValuesInterface` — Each market data interface and each book entry interface now provides the `getEPValuesInterface()` action, which you can call to access the values of any extra parameters.

The `AggregatedBook` actions and interfaces are slightly different as they offer access to the extra parameters from the underlying datasources of the Aggregator/FTSC. They require the symbol and source id (known as the XEPKey) of the underlying datasource to access the values and schema.

See the ApamaDoc for details about using these interfaces.

For an example of using the new interfaces, see "Accessing extra parameters" in the *Capital Markets Foundation*.

MATLAB plug-in moved

The MATLAB plug-in has been moved from the Apama Capital Markets Foundation into the Apama 5.3. Documentation for using MATLAB products in a CMF application can now be found in *Developing Apama Applications*, in the section "Using MATLAB products in an application".

Migrating to CMF 5.3

Apama is now built with JDK 7. You can no longer use JDK 6 (or older) to develop Apama CMF applications nor can you use JRE 6 (or older) to deploy Apama CMF applications. While it should not be necessary to rebuild applications previously built with an older Java version, the recommendation is that you build your application with the newer JDK for two important reasons:

- To ensure that you detect compatibility problems before you want to deploy the application
- To take advantage of performance improvements in newer Java compiler versions

To develop, build, and test an Apama CMF 5.3 application, the recommendation is that you use Oracle JDK 8. The minimum you can use to build your application with Apama CMF 5.3 libraries is JDK 7.

To deploy an Apama CMF 5.3 application, the recommendation is that you use Oracle JRE 8, which is the version that Apama provides. Use of any JRE other than the one that Apama ships with is discouraged.

Miscellaneous enhancements and changes

The Apama Payload Plugin has been deprecated so the CMF blocks Symbol Attribute v1.* and Order Manager v4.0 have also been deprecated. New Symbol Attribute v2.* and Order Manager v5.0 blocks have been created and should be used. They do not use the Payload Plugin and instead use the inbuilt dictionary `toString()` and `parse()` methods. The deprecated blocks will be removed in a future release.

API reference documentation has been added for the Market Data Adapter Library to aid clients who are creating there own MDA compliant adapters. A link can be found to it from the main documentation start page.

5 What's New in Capital Markets Foundation 5.2

Apama Platform for Capital Markets Release 5.2 adds the following new features and enhancements to the Capital Markets Foundation.

ATA now installed in APAMA_WORK directory

On Windows systems, the Algorithmic Trading Accelerator is now installed and run from the `APAMA_WORK` directory.

The default location is `C:\Users\`*`user_name`*`\SoftwareAG`
`\ApamaWork_5.2\ATA_5.2.0.0`.

CMF installation script now sets up Apama Studio environment

In previous releases, you were required to manually add the `APAMA_FOUNDATION_HOME` variable and the CMF bundles to Apama Studio. With CMF 5.2, the installation script does this automatically. Manual steps to set up Apama Studio for CMF development are no longer required.

CMF now uses channels internally

CMF performance has been enhanced by internal use of the Apama 5.2 channels feature.

MDA market data from adapters is now delivered directly to the subscribing context rather than through the main context, which improves both latency and throughput.

CMF can now use Universal Messaging

Universal Messaging (UM) can now be used as a message bus to deliver market data events to CMF components. Adapters and correlators can connect to UM realms to exchange Apama events. For details, see the following section in the Apama documentation: "Using Universal Messaging in Apama Applications" in *Deploying and Managing Apama Applications*.

MDA protocol and library improvements

The CMF performance has been enhanced by various MDA protocol and library improvements. The following throughput and latency improvements were recorded in laboratory conditions:

Throughput (orderbook updates/ second)	CMF 5.1.1	CMF 5.1.1	CMF 5.2	CMF 5.2
	1 context	4 contexts	1 context	4 contexts
Windows	3880	7140	6650 (+71%)	16400 (+130%)
Linux	4590	10160	6460 (+40%)	25580 (+151%)
Latency (orderbook milliseconds 99th percentile)	CMF 5.1.1	CMF 5.1.1	CMF 5.2	CMF 5.2
	Main context	Non-main context	Main context	Non-main context
Windows	0.82	0.87	0.60 (-27%)	0.60 (-31%)
Linux	0.81	0.85	0.60 (-26%)	0.60 (-29%)

The main/single context results show the performance improvements due to protocol and library improvements.

The non-main/multi-context results also show the performance and scaling improvements due to the use of channels.

These measurements were recorded on:

- Windows machine specifications: Intel® Xeon® CPU E3-1270 V2 @ 3.50GHz 8 Cores, 16Gb RAM, Windows 7 64-bit
- Linux machine specifications: Intel(R) Xeon(R) CPU E5-2690 @ 2.90GHz 32 Cores 64Gb RAM, RHEL 6 64-bit

New helper events for creating DataViews

The CMF now provides helper events to make it easier to create DataViews of `com.apama.utils.Params` events.

The `com.apama.utils.SingleParamsDataView` event lets you create a DataView that contains the contents of a single `com.apama.utils.Params` event. It creates a two-column DataView that contains a row for each key/value pair from the supplied `Params` event. Actions defined in the `SingleParamsDataView` event let you update the contents of the DataView, as well as overwrite or clear the contents, or remove the DataView.

The `com.apama.utils.MultipleParamsDataView` event lets you create a DataView that contains the contents of multiple `com.apama.utils.Params` events that conform to a schema. The `MultipleParamsDataView` event creates a DataView with columns that match a `com.apama.utils.ParamsSchema` event. Each row represents a `Params` events that matches the `ParamsSchema` event. By default, every field in the schema is a key field but you can specify key fields to identify unique rows. You pass `Params` events to `MultipleParamsDataView` events to add, update or delete selected rows.

Features removed

The following, previously deprecated features have been removed:

- MDA Quote datastream — superseded by the Quotebook datasource, which was introduced in CMF 5.0.1.
- Legacy risk firewall — superseded by the CMF risk firewall, which was introduced in CMF 5.1.
- Legacy position services — superseded by the CMF position service framework, which was introduced in CMF 5.0.
- Legacy currency converter service — superseded by the CMF currency converter service, which was introduced in CMF 5.1.1.

6 What's New in Capital Markets Foundation 5.1.1

Apama Platform for Capital Markets Release 5.1.1 adds the following new features and enhancements to the Capital Markets Foundation.

CMF 5.1.1 requires Apama 5.1.1

CMF release 5.1.1 requires Apama release 5.1.1. Be sure to install Apama 5.1.1 before you install the CMF.

New risk firewall features

The risk firewall introduced in CMF 5.1 provides the following enhancements in CMF 5.1.1:

- New default rule classes

  - Order operation ratio — Ensures that the number of amend order operations and/or cancel order operations in a specified time window do not exceed the allowed number of amend/cancel operations as defined by the specified ratio between amend/cancel order operations and new orders.

  - Order to trade ratio — Ensures that the number of order operations (new, amend, cancel) in a specified time window do not exceed the allowed number of operations as defined by the specified ratio between order operations and trades.

  - Reservation enforcer — Ensures that orders placed against reservation contracts do not exceed the minimum/maximum quantities set for reserved positions.

  CMF samples for the default risk firewall rule classes have been updated to show the use of the new rules. See `Risk Firewall Rules Samples` in the `samples` directory of your CMF installation directory.

- DataViews are exposed

  - `RiskFirewallDataview` — For each risk firewall that has been created in this correlator, this DataView contains the name of the risk firewall, a string representation of the risk firewall parameters object, and whether or not the risk firewall is locked.

  - `FirewallRuleDefinitionDataview` — For all rules registered with all risk firewalls in this correlator, this DataView contains the name of the risk firewall, the name and description of the rule class, a string representation of the parameter schema for the rule class and the priority of the rule class.

  - `AllRuleClassInstance` — This DataView is exposed by the MemoryStore table that contains entries for all rule class instances for all risk firewalls in this correlator. This DataView contains the name of the risk firewall, the name of the rule class, the ID of the rule instance and a string representation of the rule instance's configuration.

  - OrderOperationCache — This DataView is exposed by the MemoryStore table that contains entries for the order operation cache, which contains information for each failed order operation that has gone through any risk firewall in this correlator. This DataView contains extensive details for each failed order operation. The entries in the order operation cache, and hence in the MemoryStore table, are recovered upon correlator re-start and can be automatically purged according to the settings of the new risk firewall configuration parameters listed in the next item.

- New risk firewall configuration parameters to be set in the factory before creating the risk firewall:

  - ORDER_OPERATION_CACHE_TIMEOUT — Specifies the number of seconds that a row can be in the order operation cache before it is eligible to be purged.
  - ORDER_OPERATION_CACHE_PURGE_INTERVAL — Specifies the number of seconds between automatic purges of the cache.
  - ORDER_OPERATION_CACHE_PURGE_TIME — Specifies a time of day for automatic purging of the cache.
  - ORDER_OPERATION_CACHE_MAX_ROWS — Specifies the maximum number of rows that can be removed in one purge operation.
  - ORDER_OPERATION_CACHE_PATH_DEFAULT — Specifies the default file and path for the tables in the order operation store.
  - ORDER_OPERATION_CACHE_PATH_KEY — Specifies the configuration key value for setting the risk firewall default file and path for the tables in the order operation store.

  The order operation cache can get very large. Be sure to either set parameters to enable automatic purging or manually remove rows from the order operation cache by invoking the com.apama.firewall.RiskFirewall.clearObjection() action, which is new in this release.

- New configuration parameters for the default order throttle rule class:

  - REJECT_NEW_PARAMETER — Indicates whether this risk firewall rule instance should reject NewOrder objects if the throttle limit has been reached.
  - REJECT_AMEND_PARAMETER — Indicates whether this risk firewall rule instance should reject AmendOrder objects if the throttle limit has been reached.
  - REJECT_CANCEL_PARAMETER — Indicates whether this risk firewall rule instance should reject CancelOrder objects if the throttle limit has been reached.
  - REJECT_NEW_DEFAULT — Default value for whether this risk firewall rule instance should reject NewOrder objects is the throttle limit has been reached. By default, this is true, which means that new order requests are rejected if the throttle limit has been reached.
  - REJECT_AMEND_DEFAULT — Default value for whether this risk firewall rule instance should reject AmenOrder objects if the throttle limit has been reached. By

default, this is false, which means that amend order requests are allowed even if the throttle limit has been reached.

- REJECT_CANCEL_DEFAULT — Default value for whether this risk firewall rule instance should reject CancelOrder objects if the throttle limit has been reached. By default, this is false, which means that cancel order requests are allowed even after the throttle limit has been reached.

Also, the Algorithmic Trading Accelerator has been updated to use the new risk firewall. It no longer uses the legacy risk firewall.

New position service framework features

The following position service framework enhancements have been added in 5.1.1:

- The new default tracker, com.apama.position.tracker.RealizedPnLTracker provides actual profit and loss values for executed orders normalized to a specified currency. Sample code for using a realized profit and loss tracker is in the samples\Position Trackers Sample\objects directory of your CMF installation directory. For information and instructions for using the new default tracker, see "Creating and subscribing to the realized profit and loss tracker" in *Capital Markets Foundation*.
- The new action, com.apama.position.PSInterface.addSymbolSliceUpdateCallback(), registers a user-defined callback action to be called every time that the position changes for a specified symbol slice in a specified subscription. You can define more than one update callback per subscription.
- New configuration parameters for all default trackers are provided in com.apama.position.tracker.GenericPositionTrackerConstants:
  - TRACKER_CONFIG_TRACK_SEPARATE_CURRENCIES — Indicates that you want to also track FX currency positions as separate currencies. For example, a symbol slice of EUR/USD and GBP/USD will also track the positions of EUR, USD, and GBP separately.
  - TRACKER_CONFIG_CURRENCY_SEPARATOR — Lets you specify the currency separator to use when splitting symbols. The default, /, is specified by the TRACKER_CONFIG_DEFAULT_CURRENCY_SEPARATOR parameter.
- The com.apama.position.tracker.PSTrackerInterface event provides two new actions for custom position trackers:
  - subscribeSymbolSliceUpdates()
  - unsubscribeSymbolSliceUpdates()

  See the ApamaDoc for details.

New CMF position service code samples

The new `Position Trackers Sample` in the `samples` directory of the CMF installation directory contains source code for the CMF default position trackers. You can use the source code as a base for customized trackers. The new sample includes code for

- Using the position service
- Using the default position trackers, including the new realized profit and loss default position tracker
- Tracking FX single currencies
- Using the new `com.apama.position.PSInterface.addSymbolSliceUpdateCallback()` action

New currency converter service

When you create a realized profit and loss default position tracker you must specify a currency converter. The tracker uses the currency converter to obtain values that are normalized to a base currency that you specify.

Currency converter configuration parameters let you specify the symbols whose values you need to normalize. If you specify for example, `"USD/EUR"` the currency converter will automatically obtain values for `EUR` to `USD` as well as for `USD` to `EUR`. Also, if there is not a direct exchange rate between two symbols, you can specify those symbols as a cross currency and the currency converter will automatically obtain information for a third symbol that it can use to convert between the two specified symbols.

The new APIs for creating and configuring a currency converter are in the `com.apama.ccyconverter` package and follow the same conventions used for the CMF APIs for session handlers, market data subscribers, and risk firewalls:

Event	Description
`CalculationExtension`	Defines the interface used by the currency converter to communicate and control calculation objects that supply values.
`CalculationExtensionFactory`	Creates a new instance of a `CalculationExtension` object. The created object is a default implementation. As needed, you should override the provided `CalculationExtension` actions with your own implementations.

Event	Description
`Consts`	Defines the constant values used to specify `CalculationExtension` configuration values.
`CurrencyConverter`	Defines the interface for using the currency converter.
`CurrencyConverterFactory`	Creates a new instance of a `CurrencyConverter` object.
`ErrorConsts`	Defines the constant values used for error handling in the currency converter.

The steps for using a currency converter are as follows:

1. Create a currency calculation extension.
2. Create a currency converter that uses the currency calculation extension you created.
3. Provide an interface to the currency converter you created. For example, when you want to use a realized profit and loss position tracker, you must specify a currency converter.

Sample code for using a currency converter is in the `samples\Currency Converter Sample` directory in your CMF installation directory. In addition, there are two sample currency calculation extensions in the `Currency Converter Sample\eventdefinitions` directory. Information and instructions for using a currency converter and a currency calculation extension are in the "Analytics" section of the *Capital Markets Foundation*.

Deprecated bundle

The `Currency Convertor Datasource` bundle, which is in the `projects\analytics\datasources\currencyconvertor` directory of your CMF installation directory, is deprecated in this release. It will be removed in a future release.

7 What's New in Capital Markets Foundation 5.1

Apama Platform for Capital Markets Release 5.1 adds new features and enhancements to the Capital Markets Foundation. The new features and changes are described in this document.

New risk firewall architecture

With Release 5.1, the Capital Markets Foundation supports a new architecture for its risk firewall. The risk firewall is a pre-trade risk firewall. It provides a mechanism for ensuring that orders being placed to a service comply with a set of rules/criteria.

The new architecture replaces the previous CMF risk firewall that was supported in previous releases. That implementation is now referred to as the legacy risk firewall.

Note: The legacy risk firewall provided in previous releases is deprecated and will be removed in a future release. You should stop using the legacy risk firewall and use only the new risk firewall provided with CMF 5.1.

The CMF 5.1 risk firewall is significantly faster and more scalable than the previous risk firewall. The new risk firewall architecture supports multiple contexts as well as multiple separate instances of a risk firewall in the same application. In addition, the developer interface for adding custom rules has been streamlined for ease of use.

The risk firewall lets applications define a set of rules that orders must be validated against. Evaluated orders are either approved and forwarded to an order management system, or rejected and sent back to the order sender that originated the order. After an approved order is processed by an order management system, an order update passes back through the risk firewall and is processed by each rule class registered with the risk firewall. This enables the risk firewall to enforce limits. The risk firewall provides default rule classes that enforce limits on individual orders for:

- Order price
- Order quantity
- Order throttle (number of orders per second)
- Order value (quantity multiplied by price)

Also, the risk firewall provides default rule classes that enforce cumulative limits across multiple orders for:

- Client credit limit
- Open/pending position limit

The new interfaces for implementing one or more uniquely named risk firewalls in your application are:

- In the `com.apama.firewall` package:
  - `RiskFirewallFactory` creates risk firewall instances and provides connections to existing risk firewalls.

  - `RiskFirewall` is either a local risk firewall or a connection to a remote (in another context or monitor) risk firewall. Connected risk firewalls do not support all `RiskFirewall` actions.
  - `RuleClassFactory` creates the rule class interface for defining customized rule classes that can be registered with a risk firewall.
  - `RuleClass` defines the interface that the risk firewall uses to communicate with its registered rule classes.
  - `OrderSender` sends orders to the risk firewall and handles order updates after the risk firewall has processed them.
  - `OrderReceiver` receives approved orders from the risk firewall, and can forward them to order management systems, receive order updates back from the OMS, and forward those back through the risk firewall to the order senders.
- In the `com.apama.firewall.rules` package:
  - `ClientCreditLimitRiskFirewallRule`
  - `OrderPriceLimitRiskFirewallRule`
  - `OrderQuantityLimitRiskFirewallRule`
  - `OrderThrottleLimitRiskFirewallRule`
  - `OrderValueLimitRiskFirewallRule`
  - `PositionLimitRiskFirewallRule`

Note: By default, approval from the risk firewall requires an order to match at least one instance of every registered rule class. This is different from the legacy risk firewall, which approves orders only if there are no rules that match the order.

Information and instructions for using the new risk firewall architecture are in the "Risk Firewall" section of the *Capital Markets Foundation*.

New utility interfaces

This release introduces two utility interfaces that you can use with the new session interfaces, new subscribers, and new risk firewall.

- `com.apama.utils.Params`

  This object provides actions for adding, obtaining and removing parameters.
- `com.apama.utils.Error`

  This object provides actions for setting and obtaining information associated with an error.

New session management interfaces

A session provides the interface between your CMF application and a source of data, such as an external adapter. In this CMF release, you can use sessions only with market data architecture (MDA) subscribers. It is expected that future CMF releases will extend the use of sessions to other CMF components.

The event interfaces for implementing session management have been streamlined for ease of use. Common tasks, such as connection to an external datasource, can be accomplished in a single line of code.

The interfaces for using sessions are:

- `com.apama.session.SessionHandlerFactory`
- `com.apama.session.SessionHandler`

The factory object lets you create a session handler and optionally connect to a session. The session handler lets you manage the session interaction between your application and the external datasource. You use it to connect, disconnect, add parameters, add callbacks to be executed upon successful subscription, and override default error behavior for sessions. Information about how to use sessions in the "Session Management" section of the *Capital Markets Foundation*.

You can use the new utility interfaces, `com.apama.utils.Params` and `com.apama.utils.Error`, with the session management interfaces.

New market data subscribers

In a CMF application, to obtain market data, you now set up one or more market data subscribers. CMF provides subscribers for the following types of market data:

- BBA
- Depth
- News
- Orderbook
- Quotebook
- Trade

The event interfaces for subscribing to market data have been streamlined for ease of use. Common tasks, such as subscription to an external datasource, can be accomplished in a single line of code.

The following table lists the new interfaces that you use to subscribe to market data. ApamaDoc provides details about the fields in each interface.

Interface	Description

`com.apama.md.BBASubscriberFactory`
`com.apama.md.BBASubscriber`
`com.apama.md.DepthSubscriberFactory`
`com.apama.md.DepthSubscriber`
`com.apama.md.NewsSubscriberFactory`
`com.apama.md.NewsSubscriber`
`com.apama.md.OrderbookSubscriberFactory`
`com.apama.md.OrderbookSubscriber`
`com.apama.md.QuotebookSubscriberFactory`
`com.apama.md.QuotebookSubscriber`
`com.apama.md.TradeSubscriberFactory`
`com.apama.md.TradeSubscriber`
	For each type of market data, there is a factory object and a subscriber object. The factory object for each type of market data provides the same actions. Likewise, the subscriber object for each type of market data provides the same actions.
	Each factory object creates its eponymous subscriber object. In addition, each factory object provides the most common actions you would execute on the associated subscriber object. Many of these interfaces provide pairs of actions in which one takes a user-defined callback and one does not, for example, `TradeSubscriber.subscribe()` and `TradeSubscriber.subscribeCb()`
	For example, the `BBASubscriberFactory` object lets you create a BBA subscriber, or create and also subscribe to BBA data. A subscription can be for a single symbol or for multiple symbols. The `BBASubscriber` object lets you manage the subscription to BBA data. You use it to subscribe, unsubscribe, set subscription parameters, override default error behavior, manage callbacks, and obtain information about the subscription.

`com.apama.md.HandlerErrorConstants`	This object provides the possible values for the `errorType` parameter in a

Interface	Description
	com.apama.utils.Error event of an error callback of a subscriber. They also provide the possible keys for the extraParams dictionary.

You should use the new market data subscriber interfaces, in conjunction with the new session management interfaces, in place of the low-level market data interfaces, such as com.apama.session.SessionManagerFactory. Also, you can use the new utility interfaces, com.apama.utils.Params and com.apama.utils.Error, with the session management interfaces.

Documentation for using market data subscribers is in the "Market Data Management" section of the *Capital Markets Foundation*.

Machine readable news datastream added

With this release, you can use CMF's market data architecture to subscribe to news stories from common sources. News subscribers use the new session interfaces provided in this release, and behave similarly to the subscribers (also new in this release) provided for BBA, Depth, Orderbook, Quotebook, and Trade events.

When working with data from news sources you can listen for news stories that are new or updated, or have been deleted. In addition, a news event has a field that contains a dictionary of extra parameters. This ensures that you can process all information associated with a news story regardless of the source. The adapter for the news source defines any extra parameters that might be provided with news stories.

The following event interfaces are provided for working with news. For details, see "Subscribing to news" in the *Capital Markets Foundation* as well as the ApamaDoc.

- com.apama.md.N - Basic news event
- com.apama.md.ND - News story deletion event
- com.apama.md.NS - Schema of metadata that describes the extra parameters for news stories sent from the connected datasource
- com.apama.md.NSI - Metadata for a single extra parameter
- com.apama.md.NSD - Event for updating the extra parameters schema
- com.apama.md.client.CurrentNewsInterface - Interface for reading news stories
- com.apama.md.WrappedN - First event the adapter sends to your application upon subscription

Deprecated features

The CMF risk firewall provided in previous releases is deprecated and will be removed in a future release. You should stop using the legacy risk firewall and use only the new risk firewall provided with CMF 5.1.

The Quote datastream is deprecated in CMF 5.1 and will be removed in a future release. It is superseded by the Quotebook datasource, which was introduced in CMF 5.0.1.

As documented in the previous CMF release, the legacy position service is deprecated and will be removed in a future release. If you have not already done so, you should stop using the legacy position service and use only the new position service framework.

Removed features

The following features, which were deprecated in previous releases, have been removed:

- `ATOMIC_DELTA` transfer protocol, which was in the new Market Data Architecture. Use the compound delta mode instead.
- The C Math Library correlator plug-in. Use the math functions provided natively in EPL.

8 What's New in Capital Markets Foundation 5.0.1

Apama Platform for Capital Markets Release 5.0.1 is a service pack release that adds new features and enhancements to the Capital Markets Foundation. The new features and changes are described in this document.

New position service framework

Applications can use the new position service framework to calculate positions in different ways. The framework provides three default postion tracker implementations as well as EPL interfaces that let you build custom position tracker implementations.

The position service framework provides support for your applications to track:

- Open positions - actual long and short positions that result from buy or sell order executions.
- Pending positions - potential long and short positions that could occur if currently open buy or sell orders are filled.
- Reserved positions - potential future positions that have been reserved.

Position services provided in previous releases are deprecated and will be removed in a future release. You should stop using the legacy position service and use only the new position service framework.

The new position service framework uses position tracker names that are consistent with standard finance terminology but some are different from the names of the legacy order position service trackers. Specifically, the functionality provided by the legacy closed order position tracker is now provided by the open position tracker and the functionality provided by the legacy open order position tracker is now provided by the pending position tracker. The following table highlights the differences:

Deprecated Position Service Tracker/Bundle	Corresponds to New Position Service Framework Tracker/Bundle
Closed order position tracker	Open position tracker
Open order position tracker	Pending position tracker
Reserved order position tracker	Reserved position tracker
`Position Services` bundle	`Position Management Service` bundle
`Simple Position Service` bundle	`Position Trackers` bundle

Quotebook datastream added

The CMF now provides support for the Market Data Architecture (MDA) quotebook datastream, structuring quote data (price, quantity, id, etc.) into a book similar to the orderbook datastream. Using the quotebook datastream is similar to using the other Capital Markets Foundation built-in MDA datastreams. A quotebook can be accessed and iterated over by price ordering and quantity ordering. Prices in a quotebook use the Apama `decimal` data type, which was introduced in the Apama 5.0 release.

For more information on using the quotebook datastream, see "Market Data Management" in the *Capital Markets Foundation*.

For details on the quotebook datastream API, see the ApamaDoc for `com.apama.md.client` and `com.apama.md.user`.

Multi-context support for MATLAB plug-in

Applications can now use the EPL interface to the MATLAB plug-in in multiple contexts. Also, the MATLAB plug-in is now asynchronous so the processing of events on the input queue does not block.

The new recommended way to use the MATLAB plug-in is to use the `MatlabManager` event, and call the relevant action and supply a callback. The call goes directly to the MATLAB plug-in so you do not need to route a request event.

Because the MATLAB plug-in is now multi-context aware, response events are routed from the MATLAB plug-in to the calling context. Each `MatlabManager` request action automatically sets up a listener for the `*Response` event that will call the supplied callback.

New Internal Market bundle

The CMF now includes initial support for an internal market component that consists of an order management component and a matching engine. The order management component receives orders and subsequently rejects or accepts them based on its validation conditions. Accepted orders are either placed in the order book or directly passed on to the matching engine for immediate transaction. The matching engine executes the orders in the book in accordance with the market's trading mechanism. Also, the matching engine uses the same exchange engine to manage an internal order book as the existing exchange simulator uses.

You can use the internal market component in a multi-context application.

See the CMF ApamaDoc for details.

Extension interface for exchange simulator and internal market component

The order matching engine now inludes an extension interface that is used by the exchange simulator and by the internal market component. This extension interface lets you customize the handling of orders in the matching engine for:

- Validation of new orders
- Custom handling of order amendments/cancellations
- Order matching for execution

See the CMF ApamaDoc for details.

9 What's New in Capital Markets Foundation 5.0

Apama Platform for Capital Markets Release 5.0 is a fully supported release that enhances the features provided in the earlier Apama Capital Markets Extension. The new features and changes in the Apama Platform for Capital Markets, also referred to as the Capital Markets Foundation, are described in this document.

Legacy Apama finance APIs moved to CMF

Prior to Apama Release 5.0, Apama included finance APIs that are defined in the following packages:

- `com.apama.oms`
- `com.apama.mlom`
- `com.apama.marketdata`

These APIs have been moved from Apama's Finance Support bundle to CMF's Legacy Finance Support bundle.

However, with CMF 5.0, you should start using the Market Data Architecture as described in the *Capital Markets Foundation*. The legacy APIs are provided for backwards compatibility and will be deprecated in a future release.

Legacy Apama Event Modeler smart blocks moved to CMF

Prior to Apama Release 5.0, Event Modeler included a number of standard smart blocks related to capital market applications. These smart blocks are now provided with the Capital Markets Foundation instead of with Apama. The smart blocks that have been moved to the Capital Markets Foundation are listed below:

- Basket Calculator
- EWMA Calculator
- MACD Calculator
- Market Depth
- Multi-Leg Order Manager
- OBV Calculator
- Order Flow
- Order Manager
- P&L Calculator
- Position Calculator
- RSI Calculator
- Volume Distributor

- VWAP Calculator

CMF APIs distributed in correlator deployment packages

CMF 5.0 APIs are distributed in correlator deployment packages (CDPs). A CDP contains application EPL code in a proprietary, non-plain-text format. This treats EPL files similarly to Java files in a JAR file. You can inject a CDP file to the correlator just as you inject an EPL file or a JAR file containing a JMON application. When you inject a CDP file into the correlator, the package ensures that all the EPL files in the CDP are injected and injected in the correct order.

To support CDPs, Apama 5.0 includes a number of new features, which you can learn about in the *Release Notes* of Apama.

Distribution of the CMF APIs in CDPs means that you cannot modify monitors and event types defined in the CMF APIs. If this becomes a problem, contact Software AG Global Support.

Performance improvements to New Market Data Architecture adapter libraries

The performance and robustness of the New Market Data Architecture (NMDA) adapter libraries have been significantly improved.

Automated installation of CMF

Installation of the Capital Markets Foundation is now automated. Download the appropriate zip file from Software AG’s Software Download Center (SDC) in Empower at https://empower.softwareag.com, extract the contents, and run the installation executable.

FIX adapter support for New Market Data Architecture

This release of the FIX adapter uses a new transport library that works in the CMF New Market Data Architecture (NMDA) as opposed to the legacy MDA.

The Currenex FIX adapter uses the NMDA functionality.

While not all FIX adapter sessions use NMDA, the new FIX adapter transport library provides the same behavior as the old transport library. When run in "legacy" mode, existing FIX adapter configurations will work unchanged with the 5.0 FIX adapter.

In future releases, it is expected that additional FIX adapters will also use the NMDA functionality.

Improved CMF documentation

Documentation for using the CMF is now provided in a single guide with the following main sections:

- Installation and configuration
- Learning about the CMF
- Service framework
- Configuration service
- Market data management
- Synthetic datasources
- Order management
- Analytic components and services
- Utility and general component bundles
- Samples
- Legacy market data management
- Apama finance interfaces
- Using CMF smart blocks to develop scenarios

Removed and deprecated features

The following `com.apama.config.ServiceFramework` actions, which were previously deprecated, have been removed:

- `initialiseServiceInterface()`
- `getLogger()`
- `setLogPrefix()`
- `getLogPrefix()`
- `setLogLevel()`
- `getLogLevel()`
- `setLogDir()`
- `getLogFileDir()`
- `setLogFileName()`
- `getLogFile()`

- `getLogFileName()`
- `isLogDebug()`
- `isLogInfo()`
- `isLogWarn()`
- `isLogError()`
- `isLogFatal()`
- `doLog()`
- `debug()`
- `info()`
- `warn()`
- `error()`
- `fatal()`
- `DEBUG()`
- `INFO()`
- `WARN()`
- `ERROR()`
- `FATAL()`

The Logging bundle has been removed. Prior to CMF 5.0, the Logging bundle provided component actions in the `LoggingManager2` and `LatencyLogger` monitors.

In place of the logging features that have been removed, use the `engine_management` correlator utility with the `-r` *`logRequest`* option to manage log levels and log files. See the following topic in the Apama documentation: "Setting logging attributes for packages, monitors, and events" in *Deploying and Managing Apama Applications*.

The following APIs are deprecated and will be removed in a future release:

- `ATOMIC_DELTA` transfer protocol, which is in the new Market Data Architecture. Use the compound delta mode instead. Atomic delta mode has more overhead than compound delta mode, is more prone to cause problems in the application of the delta to a snapshot, and is more prone to errors in the order book if there are network problems, that is, lost events.
- The C Math Library correlator plug-in. Use the math functions provided natively in EPL.

Upgrading to CMF 5.0

If you have applications that are already using the Capital Markets Foundation and you want to upgrade to CMF 5.0 consider the following:

- The APIs that are in the `com.apama.oms`, `com.apama.mlom`, and `com.apama.marketdata` packages, which used to be included in the Apama Studio Finance Support bundle, have been moved to CMF's Legacy Finance Support bundle. To continue using these legacy APIs, add the Legacy Finance Support bundle to your project. If you have scripts that are using the default CMF ANT targets and these scripts are explicitly injecting these APIs, you can remove those injection statements, or replace them with the Legacy Finance Support bundle ANT target.
- Applications that are not using the CMF but that are using the APIs that have been moved to the Legacy Finance Support bundle must now use CMF 5.0.
- Ensure that your application no longer attempts to use the `com.apama.config.ServiceFramework` actions that were previously deprecated and have now been removed. See "Removed and deprecated features" on page 40.
- Ensure that your application does not attempt to use the Logging bundle, which has been removed. See "Removed and deprecated features" on page 40.
- Avoid using the following APIs. They are deprecated and will be removed in a future release:
  - `ATOMIC_DELTA` transfer protocol.
  - The C Math Library correlator plug-in.
- In the unusual situation that your application uses a CMF service that you have modified, contact Software AG Global Support to evaluate your options. With CMF 5.0, CMF services are provided in Correlator Deployment Packages (CDPs), which you cannot modify.