



- [Site Map](#)
- [Topic Index](#)

[Home](#)[Install](#)[Migrate](#)[Start](#)[Authoring](#)[Stream](#)[SQL](#)[Test/Debug](#)[Adapters](#)[API](#)[Admin](#)[Samples](#)[Studio](#)[Reference](#)

Current Location: [Home](#) > Adapters Guide

Adapters Guide

Adapters are programs that convert data to and from the StreamBase tuple protocol. You can use adapters to:

- Convert data from real-time sources such as financial market data feeds, including [Reuters Market Data](#).
- Convert data from an external resource such as a CSV file.
- Convert processed StreamBase tuples into the format used by an external resource, such as an XML file.

StreamBase provides two categories of adapters:

Embedded Adapters

Embedded adapters are those that run in the same StreamBase Server process that is hosting and processing your StreamBase application. In contrast to external adapters, embedded adapters do not incur a client-server communication overhead. Embedded adapters start and stop automatically with the StreamBase application, which simplifies many administration tasks.

The base StreamBase installation includes a number of embedded adapters, and others are available from separate installation kits. See [Embedded Adapters](#) for the available embedded adapters.

External Adapters

External adapters are those that run in independent processes separate from the StreamBase Server process. External adapters are provided in separate installation kits. See [External Adapters](#) for the available external adapters.

If you need functionality not provided by an existing StreamBase adapter, you can implement your own custom embedded adapter using the StreamBase Java API. See [Using the Embedded Adapter API](#) for more information.

- [Copyright](#) © 2004-2010 StreamBase Systems, Inc. All Rights Reserved.
- [Contact Us](#)



- [Site Map](#)
- [Topic Index](#)

[Home](#)[Install](#)[Migrate](#)[Start](#)[Authoring](#)[StreamSQL](#)[Test/Debug](#)[Adapters](#)[API](#)[Admin](#)[Samples](#)[Studio](#)[Reference](#)

Current Location: [Home](#) > [Adapters Guide](#) > [StreamBase Embedded Adapters](#) > Reuters RMDS Subscribing Input Adapter

Reuters RMDS Subscribing Input Adapter

Contents

[Introduction](#)

[Properties](#)

[General Tab](#)

[Adapter Properties Tab](#)

[Tuple Properties Tab](#)

[Debugging Tab](#)

[Edit Schemas Tab](#)

[Dynamic Variables Tab](#)

[Concurrency Tab](#)

[Using the Adapter in a StreamBase Application](#)

[Description of This Adapter's Ports](#)

[Adding the Adapter to an EventFlow Application](#)

[Typechecking and Error Handling](#)

[Suspend/Resume Behavior](#)

[Related Topics](#)

Introduction

The StreamBase Reuters RMDS Subscribing Adapter allows a StreamBase application to receive market data from Reuters RMDS 5 and 6 servers. This adapter supersedes the Reuters OMM Subscribing (RMDS 6) adapter and is intended to replace the Reuters Subscribing (RMDS 5) adapter, which is being phased out.

Note

The adapter described on this page supports both RMDS 5 and RMDS 6. By contrast, the older [Reuters Subscribing Input Adapter](#) supports RMDS 5 only. This Reuters Subscribing Adapter remains supported for existing projects, but new StreamBase applications should take advantage of the new adapter described on this page.

This adapter supports the following Reuters message models:

- MarketFeed (RMDS 5)

Adapters Guide

- MarketPrice
- MarketByOrder
- MarketByPrice
- MarketMaker
- SymbolList

The MarketFeed and MarketPrice models consist of a flat set of key/value pairs. The adapter emits a tuple on its market data output port for each MarketFeed or MarketPrice message received. The other models are hierarchical, containing an arbitrary number of elements, each consisting of a fixed set of key/value pairs. For these models, the adapter emits a tuple for each element affected (added, removed, or modified) by a Reuters message.

The adapter supports both initial (static) and dynamic subscriptions. Initial subscriptions are specified within an optional initial subscriptions file, which is processed when the adapter starts. Later, items can be subscribed to, or unsubscribed from, by enqueueing tuples to the adapter's dynamic subscription input port. Snapshots, which return an image but no subsequent updates, can be requested either statically or dynamically.

The adapter's set of active subscriptions can be retrieved by enqueueing a tuple to the adapter's information query input port. When queried, the adapter emits a stream of tuples representing the set of active subscriptions on its information query output port. For RMDS 6, the info query mechanism can be used to retrieve the set of fields in the loaded data dictionary.

The adapter emits tuples on its event output port to convey important events to the StreamBase application, such as login success or failure, dictionary loading activity, connection and service up/down events, and item status messages. Event tuples contain fields that specify the event type, object, action, status, and description.

The adapter is configured through a collection of properties set in the adapter's Properties view within StreamBase Studio. Properties specify, among other things, the Reuters message model being used by the adapter instance, the RFA session name, optional RFA configuration and initial subscription files, and dictionary and login information.

The optional RFA configuration (preferences) file defines RFA namespaces, sessions, and connections. It can be created using the StreamBase RFA/Java Configuration Wizard provided with StreamBase Studio or the Reuters Configuration Editor shipped with the Reuters Foundation API - Java Edition SDK. If a preferences file is provided, its contents are imported into the Java preferences database and is subsequently accessed by the RFA/Java library. If other RFA client applications have been used on the system, it may be possible to reuse the existing preferences, in which case the preferences file name can be left blank.

Properties

This section describes the properties you can set for this adapter, using the various tabs of the Properties view in StreamBase Studio.

General Tab

Name: Use this field to specify or change the component's name, which must be unique in the application. The name must contain only alphabetic characters, numbers, and underscores, and no hyphens or other special characters. The first character must be alphabetic or an underscore.

Adapters Guide

Adapter Name: A read-only field that shows the formal name of the adapter.

Class: A field that shows the fully qualified class name that implements the functionality of this adapter. Use this class name when loading the adapter in StreamSQL programs with the `APPLY JAVA` statement. You can right-click this field and select Copy from the context menu to place the full class name in the system clipboard.

Start with application: If this checkbox is enabled, an instance of this adapter starts as part of the containing StreamBase Server. If disabled, the adapter is loaded with the server, but does not start until you send an **sbadmin resume** command, or start the component with StreamBase Manager. With this option disabled, the adapter does not start even if the application as a whole is suspended and later resumed. The default and recommended setting is enabled.

Enable Error Output Port: Check this checkbox to add an Error Port to this component. In the EventFlow canvas, the Error Port shows as a red output port, always the last port for the component. See [Using Error Ports and Error Streams](#) to learn about Error Ports.

Description: Optionally enter text to briefly describe the component's purpose and function. In the EventFlow canvas, you can see the description by pressing **Ctrl** while the component's tooltip is displayed.

Adapter Properties Tab

This section describes the properties on the Adapter Properties tab in the Properties view for the Reuters RMDS Subscribing adapter.

Property	Description
Reuters Message Model	The Reuters Message Model to be used by this instance of the adapter: MarketFeed (RMDS 5), MarketPrice, MarketByOrder, MarketByPrice, MarketMaker, and SymbolList.
RFA Configuration File	The name of the RFA configuration (preferences) file. This file may not be required if the preferences from another RFA client application are being reused. If specified, the contents of this file is imported into the Java preferences database when the adapter starts.
RFA Session Name	The session name within the Java preferences database to use in accessing the Reuters RMDS server. The session name must include a namespace prefix and a double colon delimiter (::).
Download Data Dictionary (RMDS 6 only)	When set, data dictionaries are downloaded from the Reuters server. Dictionaries are downloaded from all available Reuters services that supply them. Later, when processing an OMM message containing market data, the adapter uses the "best available" dictionary, prioritized as follows: 1) dictionary downloaded from the same Reuters service as the OMM message; 2) dictionary loaded from the local file system; 3) dictionary downloaded from a different Reuters service than the OMM message. Note When using MarketFeed (RMDS 5), dictionary downloading is controlled by <code>downloadDataDict</code> Java preference.
Field Dictionary File (RMDS 6 only)	The file name of the field data dictionary. For RMDS 6 message models, a locally-loaded dictionary is used only when a dictionary has not been downloaded for

Adapters Guide

	<p>the Reuters service from which the OMM message was received.</p> <p>Note</p> <p>When using MarketFeed (RMDS 5), the local dictionary file is specified by <code>masterFidFile</code> Java preference.</p>
Enumeration Dictionary File (RMDS 6 only)	<p>The file name of the enumeration data dictionary. For RMDS 6 message models, a locally-loaded dictionary is used only when a dictionary has not been downloaded for the Reuters service from which the OMM message was received.</p> <p>Note</p> <p>When using MarketFeed (RMDS 5), the local enumeration file is specified by <code>enumTypeFile</code> Java preference.</p>
Initial Subscription File	<p>The name of the file containing zero or more subscription requests processed when the adapter starts. An example initial subscription file is provided with the adapter sample. The example file explains the syntax of a subscription request and contains several commented-out subscriptions.</p>
Auto Resubscribe (RMDS 5 only)	<p>When set, the adapter explicitly resubscribes to all items after a Reuters RMDS 5 service down/up event. This is normally not necessary, but in some Reuters environments subscriptions have been observed to get lost when a service restarts.</p>
Username	<p>The username value used to log in to the Reuters server and/or enforce entitlements.</p>
Position	<p>The position value used to log in to the Reuters server and/or enforce entitlements.</p>
Application	<p>The application value used to log in to the Reuters server and/or enforce entitlements.</p>

Tuple Properties Tab

This section describes the properties on the Tuple Properties tab in the Properties view for the Reuters RMDS Subscribing adapter.

Property	Description
Send Thin Tuples	When set, sends one tuple per field in the Reuters message.
Send Unchanged Fields As Null	When set, fields not present in a MarketFeed or MarketPrice update message are sent as <code>null</code> in the resulting market data tuple. Otherwise, the previous value of that field is sent in the tuple.
Send Image on Duplicate Subscribe	When set, the adapter emits an image tuple when a subscription is received for an item that has already been subscribed to.
Send Enum Field Display Values	When set, Reuters ENUMERATION fields are emitted as strings using the "display" value. For example, <code>USD</code> if <code>CURRENCY</code> contains <code>840</code> .

Debugging Tab

This section describes the properties on the Debugging tab in the Properties view for the Reuters RMDS Subscribing adapter.

Property	Description
Display Entitlement Events (RMDS 5 only)	When set, the adapter emits a message on receipt of each entitlement authentication event. The message includes the principal identity and entitlement status.

Adapters Guide

Display Reuters Messages Sent (RMDS 6 only)	This property is primarily a diagnostic feature. When set, the adapter displays the contents of messages it sends to the Reuters server.
Display Reuters Messages Received	This property is primarily a diagnostic feature. When set, the adapter displays the contents of messages it receives from the Reuters server.
Display Raw Marketfeed Data (RMDS 5)	This property is used in conjunction with the "Display Reuters Messages Received" property to display the raw marketfeed bytes of each received message.
Display Marketfeed Dictionary (RMDS 5)	This property is used in conjunction with the "Display Reuters Messages Received" property to debug dictionary-related problems. When enabled, the contents of the marketfeed data dictionary is displayed once, upon processing the first Reuters message.
Reuters Message Display Filter	This property is primarily a diagnostic feature. If blank, every Reuters message is subject to display based on the setting of the previous two properties. Otherwise, the filter is a set of message model names delimited with the pipe symbol (). Valid values include: LOGIN, DIRECTORY, DICTIONARY, MARKET_FEED, MARKET_PRICE, MARKET_BY_ORDER, MARKET_BY_PRICE, MARKET_MAKER, and SYMBOL_LIST. Thus, for example, to see all Login and MarketPrice OMM messages received, set this property to LOGIN MARKET_PRICE and set Display Reuters Messages Received to true.

Edit Schemas Tab

Use the Edit Schemas tab to specify the schema of the market data output for this adapter.

For general instructions on using the Edit Schemas tab, see [Defining Input Streams](#).

Dynamic Variables Tab

Use the Dynamic Variables tab to define variables for this operator that can then be used in one of its expressions. A dynamic variable can be updated by any input stream or output stream in your application. For further information, see [Using Dynamic Variables](#).

Concurrency Tab

Use the Concurrency tab to specify parallel processing options for this instance of this component.

Caution

Concurrency settings are not suitable for every application, and using these concurrency options requires a thorough analysis of your application. For details, see [Execution Order, Concurrency, and Parallelism](#), which includes important guidelines for using the concurrency options.

Run this component in a separate thread

This option causes the server to process the component's requests concurrently with other processing in the application. You can distribute the processing of the threads automatically across multiple processors.

Adapters Guide

If this is a compute-intensive component and you know that it can run without data dependencies on other components in the StreamBase application, you may be able to improve performance by enabling this option.

Run in parallel threads

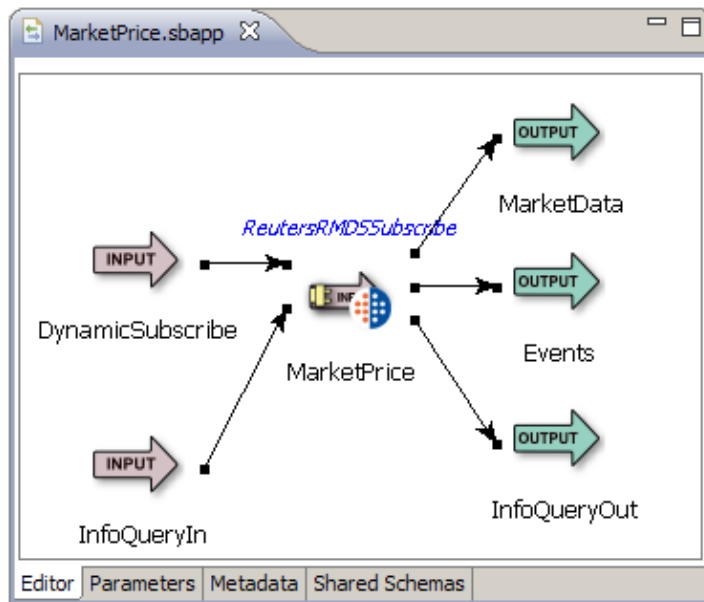
If you checked the separate thread option, you can also choose the parallel threads option, which causes the server to run multiple instances of this component, each in its own thread. Specify the Number of threads to run, and specify a Key expression, which must evaluate to an int. At run time, tuples are dispatched to particular thread instances based on the Key Expression value.

Using the Adapter in a StreamBase Application

This section demonstrates how to use the Reuters RMDS subscribing adapter within a StreamBase application and describes the use of the adapter's ports. As shown in the EventFlow diagram below (from this adapter's sample application), the adapter uses two input ports and three output ports to communicate with the surrounding application. As with other StreamBase adapters and operators, you can optionally enable an Error Output Port, as described in [Using Error Ports and Error Streams](#).

Note

In this adapter's sample application, the adapter's input and output ports are connected directly to externally-visible input and output streams. However, in more complex applications these ports will typically be connected to internal StreamBase operators.



Description of This Adapter's Ports

The Reuters RMDS subscribing adapter's ports are used as follows:

- **DynamicSubscribe** (input): Tuples enqueued on this port cause the adapter to subscribe to, or unsubscribe from, items after the adapter has started. Initial subscriptions (those processed during startup from entries in the initial subscriptions file) can later be unsubscribed from using this port. The schema of the DynamicSubscribe port is derived from the upstream operator or stream and must have

Adapters Guide

the following fields:

- Subject, string, MarketFeed only: specifies the Reuters 4-part subject name to subscribe to or unsubscribe from. When subscribing by 4-part subject, leave the service and item fields null.
- Service, string: specifies the Reuters service to subscribe to or unsubscribe from.
- Item, string: specifies the item to subscribe to or unsubscribe from.
- Subscribe, bool: if true, results in a new subscription. Note that the adapter treats bool fields containing null as false. If both Subscribe and Snapshot are false, unsubscribes.
- Snapshot, bool: if true and Subscribe is false, requests an image with no subsequent updates.
- **InfoQueryIn** (input): Tuples enqueued on this port request information from the adapter. The adapter provides the requested information by emitting tuples on its InfoQueryOut port. The InfoQueryIn port has the following fields:
 - Command, string: accepts a command specifying the type of metadata being requested. Valid commands include:
 - ◆ ListSubscriptions: returns the current set of active subscriptions.
 - ◆ DumpDictionary: for RMDS 6 only, returns the contents of the loaded field dictionary. If multiple dictionaries are loaded, use the Param field to specify the Reuters service name used to load the dictionary, or null to dump the dictionary loaded from the local file system.
 - Param, string: accepts a command-specific parameter value. Currently, only the DumpDictionary command uses a parameter, which provides the service name used to download the dictionary, or null to dump a locally-loaded dictionary.
 - Tag, string: this value is echoed in each tuple emitted from the information query output port in response to a command, allowing responses to be matched with commands.
- **MarketData** (output): This is the adapter's primary output port. The adapter emits tuples on this port when it receives Reuters market data messages. This section describes the schema usage for the market data port, and describes how and when the adapter emits market data tuples.

The market data port's schema consists of both market data and metadata fields. Market data fields are entered via Studio's Edit Schema tab, and their names must match those in the data dictionary. The adapter adds the following metadata fields to the market data output schema:

- MessageType, string: contains IMAGE (MarketFeed only), REFRESH (non-MarketFeed only), UPDATE, UNSOLICITED_IMAGE, CORRECTION, CLOSING_RUN, or STATUS, depending upon the type of Reuters message that generated the tuple.
- Subject, string, MarketFeed only: contains the Reuters 4-part subject name.
- Service, string: contains the Reuters service name.
- Item, string: contains the item name.
- RefreshComplete, boolean, MarketByOrder, MarketByPrice, and MarketMaker only: set to true in an item's last refresh tuple, false in all other refresh tuples, and null for non-refresh tuples.
- Action, string, MarketByOrder, MarketByPrice, and MarketMaker only: contains ADD, DELETE, or UPDATE to indicate the action occurring for the specific entry in the order book, market depth, or market maker table.
- Key, string, for MarketByOrder, MarketByPrice, and MarketMaker only: contains a unique identifier for the entry in the order book, market depth, or market maker table.
- State, string, MarketFeed only: for status and image messages, contains one of the following values to convey the item's state:

Adapters Guide

- ◆ OK
 - ◆ CLOSED
 - ◆ CLOSED_RECOVER
 - ◆ STALE
 - ◆ NO_CHANGE
 - StreamState, string, non-MarketFeed only: for status and refresh messages, contains one of the following values to convey the item's current stream state:
 - ◆ UNSPECIFIED
 - ◆ OPEN
 - ◆ NONSTREAMING
 - ◆ CLOSED_RECOVER
 - ◆ CLOSED
 - ◆ REDIRECT
 - DataState, string, non-MarketFeed only: for status and refresh messages, contains one of the following values to convey the item's current data state:
 - ◆ UNSPECIFIED
 - ◆ OK
 - ◆ SUSPECT
 - StatusCode, string: contains one of the following values to convey the item's current status:
 - ◆ 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
 - StatusText, string: contains a human-readable description of a item's current status.
- When configured to send thin tuples, the adapter adds the following fields to the market data output schema:
- Fid, int: contains the FID of the Reuters message field.
 - FidName, string(30): contains the FID acronym (e.g., "BID") of the Reuters message field.
 - ValueType, string(10): contains the StreamBase data type used to hold the field value: int, double, string, timestamp, or blob.
 - IntValue, int: contains the field value for Reuters fields of type INTEGER. In addition, Reuters ENUMERATED field values are conveyed in this field when the adapter's "Send Enum Field Display Values" property is disabled.
 - DoubleValue, double: contains the field value for Reuters fields of type PRICE.

Adapters Guide

- **StringValue**, double: contains the field value for Reuters fields of type ALPHANUMERIC. In addition, Reuters DATE, TIME, and TIME_SECONDS field values are conveyed in this field if the adapter is unable convert the data/time value to a StreamBase timestamp. Finally, Reuters ENUMERATED field values are conveyed in this field when the adapter's "Send Enum Field Display Values" property is enabled.
- **TimestampValue**, timestamp: contains the field value for Reuters fields of type DATE, TIME, or TIME_SECONDS.
- **BlobValue**, blob: contains the field value for Reuters fields of type BINARY.

The market data fields present in the market data schema, and the frequency and method with which the adapter emits tuples on this port, vary with the message model the adapter is configured for.

- **MarketFeed** is used by RMDS 5 Reuters servers to provide access to trades and quotes from exchanges. The data is structured as field-value pairs, and the adapter emits a single tuple for each MarketFeed message it receives. The first tuple emitted for an item contains the item's initial image (`MessageType=IMAGE`). The adapter subsequently emits update tuples (`MessageType=UPDATE`) as market information for that item changes. The adapter can be configured to set unchanged fields in update tuples to `null` or to the field's previous value.
- **MarketPrice** is used by RMDS 6 Reuters servers to provide access to trades and quotes from exchanges. The data is structured as field-value pairs, and the adapter emits a single tuple for each OMM message it receives. The first tuple emitted for an item contains the item's initial image (`MessageType=REFRESH`). The adapter subsequently emits update tuples (`MessageType=UPDATE`) as market information for that item changes. The adapter can be configured to set unchanged fields in update tuples to `null` or to the field's previous value.
- **MarketByOrder** provides access to full order books. An order book consists of summary information, such as currency, which applies to all orders in the book, along with a set of zero or more orders. When an application subscribes to a new item, the adapter retrieves the initial order book contents through a set of OMM refresh messages. Upon retrieving the entire order book, the adapter emits a set of refresh tuples (`MessageType=REFRESH`), one for each order in the book. The adapter then processes OMM messages containing changes to orders in the book, each of which results in an update tuple (`MessageType=UPDATE`). The specific action — `ADD`, `DELETE`, or `UPDATE` — is conveyed in the `Action` metadata field. An order's unique identifier is conveyed in the `Key` metadata field. Summary information appears in every refresh and update tuple.

The MarketByOrder data fields typically available are:

- ◆ Permission information (`PROD_PERM`)
- ◆ Currency of the orders (`CURRENCY`)
- ◆ Trade Units for the precision for which order prices are set (`TRD_UNITS`)
- ◆ Market State (`MKT_ST_IND`)
- ◆ Exchange Identifier on which the orders were placed (`RDN_EXCHD2`)
- ◆ Price Ranking Rules (`PR_RNK_RUL`)
- ◆ Order Ranking Rules (`OR_RNK_RUL`)
- ◆ Order Price and Side (`BID` and `ASK`, or `ORDER_PRC` and `ORDER_SIDE`)
- ◆ Order Size (`BIDSIZE`, `ASKSIZE`, or `ORDER_SIZE`)
- ◆ Price Qualifiers (`PRC_QL_CD`, `PRC_QL2`)
- ◆ Market Maker Identifier (`MKT_MKR_ID` or `MMID`)
- ◆ Quote Time (`QUOTIM_MS`)
- **MarketByPrice** provides access to market depth information — that is, to Level II data that is aggregated by price and side. The mechanics of tuple emission is identical to that for MarketByOrder. The adapter processes a set of OMM refresh messages to retrieve the full

Adapters Guide

market depth information for an item, emits a refresh tuple (`MessageType=REFRESH`) for each order, then emits update tuples (`MessageType=UPDATE`) as market depth changes for that item. The `Action` and `Key` metadata fields work identically as for the `MarketByOrder` message model, and summary information appears in all tuples.

The `MarketByPrice` data fields typically available are:

- ◆ Permission information (`PROD_PERM`)
- ◆ Currency of the orders (`CURRENCY`)
- ◆ Trade Units for the precision for which order prices are set (`TRD_UNITS`)
- ◆ Market State (`MKT_ST_IND`)
- ◆ Exchange Identifier on which the orders were placed (`RDN_EXCHD2`)
- ◆ Price Ranking Rules (`PR_RNK_RUL`)
- ◆ Order Price and Side (`BID` and `ASK`, or `ORDER_PRC` and `ORDER_SIDE`)
- ◆ Order Size (`BIDSIZE`, `ASKSIZE`, or `ORDER_SIZE`)
- ◆ Number of aggregated orders (`NO_ORD`)
- ◆ Quote Time (`QUOTIM_MS`)
- **MarketMaker** provides access to market maker quotes and trade information. The mechanics of tuple emission is identical to that for `MarketByOrder` and `MarketByPrice`.

The `MarketMaker` data fields typically available are:

- ◆ Permission information (`PROD_PERM`)
- ◆ Currency of the orders (`CURRENCY`)
- ◆ Trade Units for the precision for which order prices are set (`TRD_UNITS`)
- ◆ Market State (`MKT_ST_IND`)
- ◆ Exchange Identifier on which the orders were placed (`RDN_EXCHD2`)
- ◆ Price Ranking Rules (`PR_RNK_RUL`)
- ◆ Bid (`BID`)
- ◆ Ask (`ASK`)
- ◆ Bid Size (`BIDSIZE`)
- ◆ Ask Size (`ASKSIZE`)
- ◆ Market Source (`MKT_SOURCE`)
- ◆ Market Maker Name (`MKT_MKR_NM`)
- ◆ Price Qualifiers (`PRC_QL_CD` and `PRC_QL2`)
- ◆ Quote Time (`QUOTIM_MS`)
- **Symbol List** provides a mechanism to retrieve all the symbols available on an exchange. For example, you can subscribe to `0#ARCA` to retrieve the set of symbols available on the ARCA exchange.

The `Symbol List` data fields typically available are:

- ◆ Permission information (`PROD_PERM`)
- ◆ The symbol provided by the exchange (`PROV_SYMB`)
- **Events** (output): The adapter emits tuples from this port when significant events occur, such as when a login succeeds or fails, when a data dictionary is loaded, when Reuters connections and services go up and down, and when the adapter is told an item is closed. The schema for this port has the following fields:
 - `EventType`, string: returns one of the following values to convey the type of event:

Adapters Guide

- ◆ Connection
- ◆ Dictionary
- ◆ Directory
- ◆ Item
- ◆ Login
- ◆ Service
- ◆ Suspend/Resume
- ◆ UserInput
- Object, string: returns an event type-specific value, such as the connection or service name that went up or down, the dictionary file name that was loaded, or the user input that was rejected.
- ConnectionType, string, MarketFeed only: contains the type of RMDS 5 connection, such as SSLED, that went up or down.
- Action, string: returns an action associated with the event Type and Object, such as Loaded or Failed to load after a dictionary load attempt.
- State, string, MarketFeed only: returns the state of an RMDS 5 connection, service, or item. Refer to the description of the State field in the Market Data port for a list of the possible values.
- StreamState, string, non-MarketFeed only: returns the stream state returned in an OMM message. Refer to the description of the StreamState field in the Market Data port for a list of the possible values.
- DataState, string, non-MarketFeed only: returns the data state returned in an OMM message. Refer to the description of the DataState field in the Market Data port for a list of the possible values.
- StatusCode, string: returns a code to convey status about the event. Refer to the description of the StatusCode field in the Market Data port for a list of the possible values.
- StatusText, string: contains a human-readable description of a connection, service or item's current status.
- ItemEventType, string, MarketFeed only: returns the type of item event:
 - ◆ IMAGE
 - ◆ UNSOLICITED_IMAGE
 - ◆ UPDATE
 - ◆ CORRECTION
 - ◆ CLOSING_RUN
 - ◆ STATUS
 - ◆ RENAME
 - ◆ PERMISSION_DATA
 - ◆ GROUP_CHANGE
- ItemName, string: returns the item name associated with the event.
- SubjectName, string, MarketFeed only: returns the 4-part subject name associated with the event.
- NewItemName, string, (MarketFeed only): on item rename events, returns an item's new name.
- NewSubjectName, string, MarketFeed only: on item rename events, returns an item's new 4-part subject name.
- Info, string: Returns a human-readable description of the event.
- **InfoQueryOut** (output): The adapter emits tuples on this port in response to information query commands. The InfoQueryOut port has the following schema:

Adapters Guide

- Done, bool: Set to `false` for all but the last tuple emitted in response to a specific command. The final (marker) tuple has Done set to `true` and all other fields set to `null`.
- Command, string: returns the command value specified in the corresponding information query request tuple.
- Param, string: returns the parameter value specified in the corresponding information query request tuple.
- Tag, string: returns the tag value specified in the corresponding information query request tuple.
- Info1-Info5, string: returns one or more command-specific values. The description of the InfoQueryIn stream above lists the information returned for each command.

Adding the Adapter to an EventFlow Application

Add an instance of the adapter to a new StreamBase EventFlow application as follows:

1. Within StreamBase Studio, create a project, including an empty StreamBase EventFlow Application file, which will host the adapter.
2. Import into the project the sample RFA configuration file, `rfa-config.xml`, which is located in `streambase-install-dir/sample/adapter/embedded/reuters-rmds-sub`. Edit this file and change the P2PS (`serverList` and `portNumber`) or RTIC (`daemon`, `network`, and `service`) parameters to match the Reuters infrastructure at your site. Take note of the namespace and session names, as they will be used below. In the configuration file shipped with the sample, these values are `SBSubscribeNamespace` and `SBSubscribeRFA6Session` (`SBSubscribeSession` for MarketFeed), respectively.
3. From the Global Adapters drawer of the Palette view, drag a ReutersRMDSSubscribe component to the canvas.
4. Double-click the adapter icon, and in the Properties view, select the Adapter Settings tab.
5. Select a Reuters Message Model from the drop-down.
6. In the RFA Configuration File field's dropdown list, select the RFA Configuration File you imported into the project above.
7. In the RFA Session Name field, enter the namespace and session names, delimited by a double colon (`: :`) in the RFA Session Name text entry box. For example, enter `SBSubscribeNamespace::SBSubscribeSession`.
8. If you will be downloading data dictionaries from the Reuters server, mark the Download Data Dictionary checkbox (for MarketFeed, you must edit the RFA preferences file, `rfa-config.xml`, to enable dictionary downloading). If not downloading, enter the names of the Field and Enumeration dictionary files. Note that sample versions of these files are shipped with the adapter in `streambase-install-dir/sample/adapter/embedded/reuters-rmds-sub/RDMFieldDi` (`appendix_a` for MarketFeed) and `streambase-install-dir/sample/adapter/embedded/reuters-rmds-sub/enumtype-r` (`enumtype-rmds5.def` for MarketFeed), which you can import into your project and select from the drop-downs.
9. If you wish to have the adapter subscribe to one or more items when it starts, you can specify an Initial Subscriptions File Name in the drop-down. A sample file is available for import into your project in `streambase-install-dir/sample/adapter/embedded/reuters-rmds-sub/initial_subscriptions.txt`.
10. Enter Username, Position, and Application values as necessary for accessing the Reuters server at your site. Note that at many sites one or more of these parameters are optional and can be left blank.
11. By default, the adapter, in processing Market Price update messages, sends `null` in unchanged tuple fields. Uncheck Send Unchanged Fields as Null to have the adapter send the previous value of such

fields.

12. The last three adapter properties — Display OMM Messages Sent and Received, and OMM Message Display Filter — are used for debugging and should typically be left with their default, empty values.
13. Connect Input and Output Streams to the adapter's two input and three output ports.
14. Configure the schemas of the two input ports and add market data fields to the adapter's primary (MarketData) output port. (The schemas of the Events and InfoQueryOut output ports are set automatically by the adapter, as are the metadata fields of the MarketData output port.) The specific fields required in the two input schemas are specified above in the descriptions of the DynamicSubscribe and InfoQueryIn ports. The adapter guides you through this process, displaying Typecheck Errors when an expected field is missing or of the wrong type or length, or if an unexpected field is present in the schema.

Typechecking and Error Handling

The Reuters RMDS subscribing adapter uses typecheck messages to help you configure the adapter within your StreamBase application. In particular, the adapter generates typecheck messages for the following reasons:

- A Reuters Message Model has not been selected from the drop-down.
- An invalid RFA configuration file name has been entered.
- An invalid RFA session name has been entered. Session names must be of the form `namespace::session`.
- Dictionary downloading is disabled and no field or enumeration dictionary file name has been provided.
- An invalid initial subscription file name has been entered.
- One or more required fields in the two input schemas or primary output schema is missing or is of the wrong type or size.

The adapter generates warning messages during runtime under various conditions, including:

- The adapter fails to log in to the Reuters server.
- The adapter is unable to read a local data dictionary file.
- An unexpected or malformed Reuters message is received.
- In processing an Reuters message, a string is truncated to fit in an output tuple.
- The data type for a field in the market data schema is incompatible with the data type in the Reuters message.
- A field in the market data output schema is not present in the data dictionary.
- A string field in the market data output schema is shorter than the field length specified in the data dictionary.
- A field in an Reuters message is not present in the data dictionary.
- A field in the market data output schema is not present in a Reuters refresh (image) message.
- The adapter is configured with an invalid Reuters Message Display Filter.
- An invalid line is detected in the initial subscription file.
- A dynamic subscription input tuple contains a null or empty server or item name.
- A request is made to unsubscribe from an item that is not currently subscribed to.
- An information query input tuple contains a null or empty command or tag value or an unrecognized command.
- An information query input tuple requests a dump of the data dictionary before it has been loaded.

Suspend/Resume Behavior

When suspended, the adapter continues to receive and process Reuters messages, but no longer emits tuples on its primary output port.

When resumed, the adapter once again starts emitting tuples on its primary output port.

Related Topics

- [Using Embedded Adapters](#)
- [Copyright](#) © 2004-2010 StreamBase Systems, Inc. All Rights Reserved.
- [Contact Us](#)



- [Site Map](#)
- [Topic Index](#)

[Home](#)[Install](#)[Migrate](#)[Start](#)[Authoring](#)[Stream](#)[SQL](#)[Test/Debug](#)[Adapters](#)[API](#)[Admin](#)[Samples](#)[Studio](#)[Reference](#)

Current Location: [Home](#) > [Adapters Guide](#) > StreamBase Embedded Adapters

StreamBase Embedded Adapters

Embedded Adapters

Embedded adapters are those that run in the same StreamBase Server process that is hosting and processing your StreamBase application. In contrast to external adapters, embedded adapters do not incur a client-server communication overhead. Embedded adapters start and stop automatically with the StreamBase application, which simplifies many administration tasks.

Input Adapters

The following embedded input adapters are included with the base StreamBase installation:

- [ActivFeed Input Adapter](#)
- [Alpha Trading Systems EMAPI Market Data Feed Adapter](#)
- [Alpha Trading Systems EMAPI Order Entry Adapter](#)
- [Binary File Reader Input Adapter](#)
- [Comstock \(PlusFeed\) Input Adapter](#)
- [CSV File Reader Input Adapter](#)
- [CSV Socket Reader Input Adapter](#)
- [Currenex Adapter](#)
- [EBS Adapter](#)
- [Deutsche Bank AutobahnFX Trading System Adapter](#)
- [FIX Adapter](#)
- [Goldman Sachs Electronic Trading FX Adapter](#)
- [HA Heartbeat Input Adapter](#)
- [HTTP Reader Input Adapter](#)
- [Hotspot FX Trading System Adapter](#)
- [InfoReach TMS Adapter](#)
- [IRC Reader Input Adapter](#)
- [Lime Citrius Quote Input Adapter](#)
- [Once Input Adapter](#)
- [Opentick Input Adapter](#)
- [POP3 Reader Input Adapter](#)
- [Regular Expression File Reader Input Adapter](#)
- [Regular Expression Socket Reader Input Adapter](#)
- [Reuters RMDS Subscribing Input Adapter](#)

Adapters Guide

- [Reuters Subscribing Input Adapter \(Deprecated\)](#)
- [RSS Reader Input Adapter](#)
- [SMTP Reader Input Adapter](#)
- [StreamBase to StreamBase Input Adapter](#)
- [TIBCO Rendezvous Subscribing Input Adapter](#)
- [UBS FX2B FIX Adapter](#)
- [Wombat MAMA Input Adapter](#)

StreamBase Systems provides a separate installation kit for the JMS embedded input adapters, which are bundled with the JMS external adapters. Contact your StreamBase Systems representative about ordering adapter kits.

- [JMS Input and Output Adapters](#)

Output Adapters

The following embedded output adapters are included with the base StreamBase installation:

- [Adobe Flex Output Adapter](#)
- [Alpha Trading Systems EMAPI Order Entry Adapter](#)
- [Binary File Writer Output Adapter](#)
- [CSV File Writer Output Adapter](#)
- [CSV Socket Writer Output Adapter](#)
- [Deutsche Bank AutobahnFX Trading System Adapter](#)
- [E-mail Sender Output Adapter](#)
- [Currenex Adapter](#)
- [EBS Adapter](#)
- [FIX Adapter](#)
- [Goldman Sachs Electronic Trading FX Adapter](#)
- [Hotspot FX Trading System Adapter](#)
- [Log Output Adapter](#)
- [Reuters RMDS Publishing Output Adapter](#)
- [StreamBase to StreamBase Output Adapter](#)
- [TIBCO Rendezvous Publishing Output Adapter](#)
- [XML File Writer Output Adapter](#)
- [XML over HTTP Writer Output Adapter](#)
- [UBS FX2B FIX Adapter](#)

Global Repository

The JAR files that implement embedded adapters reside in a *Global Repository*. The Global Repository is created in the following default locations:

On Windows

```
C:\Program Files\StreamBase Systems\StreamBase.n.m\lib\adapter\
```

On Linux

```
/opt/streambase/lib/adapter/
```

Related Topics

- [StreamBase Embedded Adapters](#) in the *Authoring Guide* describes how to include embedded adapters in your application
- [Using the Embedded Adapter API](#) in the *API Guide*
- [Copyright](#) © 2004-2010 StreamBase Systems, Inc. All Rights Reserved.
- [Contact Us](#)



- [Site Map](#)
- [Topic Index](#)

[Home](#)[Install](#)[Migrate](#)[Start](#)[Authoring](#)[Stream](#)[SQL](#)[Test/Debug](#)[Adapters](#)[API](#)[Admin](#)[Samples](#)[Studio](#)[Reference](#)

Current Location: [Home](#) > [Adapters Guide](#) > StreamBase External Adapters

StreamBase External Adapters

External Adapters

External adapters are those that run in independent processes separate from the StreamBase Server process. External adapters are provided in separate installation kits.

- [Excel External Adapter](#) (Windows only)
- [JDBC External Adapter](#)
- [JMS External Adapters](#)
- [Managed Publishing Adapter for Reuters RFA](#) (Windows only)
- [Reuters Messaging External Adapter](#) (Windows only)
- [Reuters SFC External Adapter](#)
- [TIBCO Rendezvous External Adapter](#)

When you install an external adapter kit, its individual documentation file is installed in *streambase-install-dir/doc/adapter*, as well as in this guide.

Related Topic

- [Supported Configurations](#) in the Installation Guide
- [Copyright](#) © 2004-2010 StreamBase Systems, Inc. All Rights Reserved.
- [Contact Us](#)