Adapters Guide

SAP Sybase Event Stream Processor 5.1 SP03

DOCUMENT ID: DC01615-01-0513-01
LAST REVISED: August 2013

Contents

CHAPTER 1 Introduction

SAP® Sybase® Event Stream Processor includes an extensive set of input and output adapters that you can use to subscribe to and publish data. Additional specialized adapters for Event Stream Processor are available from SAP as optional add-ons. Event Stream Processor also provides an adapter toolkit, several SDKs, and an internal adapter API that enable you to write a custom adapter.

See the *Building Custom Adapters* guide for detailed information on writing your own adapter.

The supplied adapters support numerous datatypes, as well as providing datatype mapping for unsupported types.

Input and Output Adapters

Input and output adapters enable Event Stream Processor to send and receive messages from dynamic and static external sources and destinations.

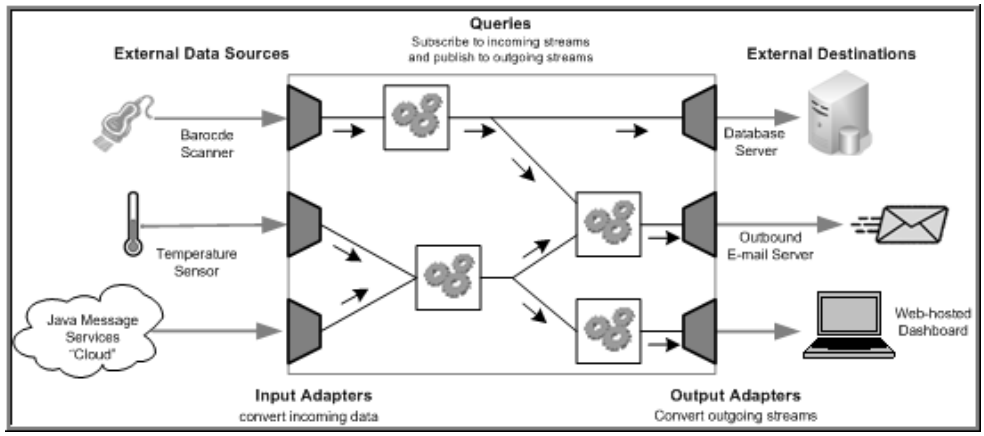
External sources or destinations can include:

- Data feeds
- Sensor devices
- Messaging systems
- Radio frequency identification (RFID) readers
- E-mail servers
- Relational databases

Input adapters connect to an external datasource and translate incoming messages from the external sources into a format that is accepted by the ESP Server. Output adapters translate rows published by Event Stream Processor into message formats that are compatible with external destinations and send those messages downstream.

The following illustration shows a series of input adapters that translate messages from a temperature sensor, bar code scanner, and a Java Message Service (JMS) cloud into formats compatible with Event Stream Processor. After the data is processed using various queries within Event Stream Processor, output adapters convert the result rows into updates that are sent to an external database server, e-mail server, and Web services dashboard.

Figure 1: Adapters in Event Stream Processor



Subscribing to Data with Input Adapters

Attach an input adapter to the ESP Server using an **ATTACH ADAPTER** statement and subscribe to data from an external datasource. If you wish to configure, start, and stop an adapter independently of the ESP Server (known as an unmanaged adapter), do not reference the adapter in the **ATTACH ADAPTER** statement.

When specifying paths, use forward slashes for both Windows and UNIX. If you use back slashes in a CCL file, an error displays because a back slash indicates a control character.

1. Assess the input data. Determine which sets or subsets of data you want to pull into Event Stream Processor.
2. Choose an input adapter suited for this task.

   If the datasource uses datatypes that are not supported by the ESP Server, the Server maps the data to an accepted datatype. Review the associated mapping description for your adapter in the *Adapters Guide*.
3. Create an input stream or window.
4. Use the **CREATE SCHEMA** statement to define the structure for incoming data within this stream or window.
5. (Skip this step if using an unmanaged adapter) Use the **ATTACH ADAPTER** statement to attach your adapter to the newly created stream or window, and specify values for the adapter properties.

   To declare default parameters for your adapter properties, use the **DECLARE** block and **parameters** qualifier to define default parameter values before you attach your adapter. Once you create the **ATTACH ADAPTER** statement, you can set the adapter properties to the parameter values you declared.

Note: You can bind declared parameters to a new value only when a module or project is loaded.

6. Start the ESP project. If you are using an unmanaged adapter, start the adapter manually.

Next

For detailed information on configuring individual Event Stream Processor-supplied adapters, datatype mappings, and schema discovery, see the *Adapters Guide*. For detailed information on CCL queries and statements, such as the **ATTACH ADAPTER**,**CREATE SCHEMA**, and **DECLARE** statements, see the *Programmers Reference Guide*.

Publishing Data with Output Adapters

Attach an output adapter to the ESP Server using an **ATTACH ADAPTER** statement and publish data to an external datasource. If you wish to configure, start, and stop an adapter independently of the ESP Server (known as an unmanaged adapter), do not reference the adapter in the **ATTACH ADAPTER** statement.

When specifying paths, use forward slashes for both Windows and UNIX. If you use back slashes in a CCL file, an error displays because a back slash indicates a control character.

1. Assess the output data. Determine which sets or subsets of data you want to send to an external datasource.
2. Choose an output adapter suited for this task.

   If the output destination uses datatypes that are not supported by the ESP Server, the Server maps the data to an accepted datatype. Review the associated mapping description for your adapter in the *Adapters Guide* to ensure that the resulting datatype is permitted by the external data destination.
3. Create an output stream or window.
4. Use the **CREATE SCHEMA** statement to define the structure for outgoing data within this stream or window.
5. (Skip this step if using an unmanaged adapter) Use the **ATTACH ADAPTER** statement to attach your adapter to the output stream or window, and set values for the adapter properties.

   To declare default parameters for your adapter properties, use the **DECLARE** block and **parameters** qualifier to define default parameter values before you attach your adapter. Once you create the **ATTACH ADAPTER** statement, you can set the adapter properties to the parameter values you declared.

   Note: You can bind declared parameters to a new value only when a module or project is loaded.
6. Start the ESP project. If you are using an unmanaged adapter, start the adapter manually.

Next

For detailed information on configuring individual Event Stream Processor-supplied adapters, datatype mappings, and schema discovery, see the *Adapters Guide*. For detailed information on CCL queries and statements, such as the **ATTACH ADAPTER**,**CREATE SCHEMA**, and **DECLARE** statements, see the *Programmers Reference Guide*.

Editing Adapter Property Sets

Use the CCR Project Configuration editor in Studio to configure adapter property sets, which are reusable groups of properties that are stored in the project configuration file.

Property sets appear in a tree format, and individual property definitions are shown as children to property sets.

1. In the CCR Project Configuration editor, select the **Adapter Properties** tab.
2. (Optional) To create a list of adapter property sets that correspond to the Attach Adapter statements in the main CCL file for the project, click **Add from CCL**.
3. To create a new adapter property node, click **Add**.
4. In the Property Set Details pane, define a name for the property node.
5. To add a new property to a property set, right-click the set and select **New** > **Property**.

   Note: You can add as many property items to a property set as required.
6. To configure a property:
   a) In the Property Details pane, define a name for the property.
   b) Enter a value for the property.
7. (Optional) To encrypt the property value:
   a) Select the property value and click **Encrypt**.
   b) Enter the required fields, including Cluster URI and credential fields.
   c) Click **Encrypt**.
      The value, and related fields are filled with randomized encryption characters.

      Note: To reset the encryption, click **Encrypt** beside the appropriate field. Change the values, as appropriate, then click **Reset**.
8. To remove items from the All Adapter Properties list:
   - Right-click a property set and select **Remove**, or
   - Right-click a property and select **Delete**.

Adapter Logging Configuration

Specific adapters currently available from SAP use the `log4j` API to log errors, warnings, and debugging messages.

You can modify the logging levels of the `log4j.properties` configuration file which is located in the `%ESP_HOME%\adapters\<adapter-name>\config` directory. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
```

```
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Internal and External Adapters

An adapter that runs as part of the Event Stream Processor Server is called an internal adapter. An adapter that runs as a separate process is called an external adapter.

Internal adapters are started by the Server when it starts the corresponding project (query module). The adapter is recognized by SAP Sybase Event Stream Processor Studio, and from inside Studio, you can attach the adapter to a stream by selecting the adapter from a menu of adapters. A disadvantage of internal adapters is that if the adapter fails, it may also cause the Server to fail.

External adapters have more flexibility than internal adapters and can run on a different machine than the Server. They can be either "managed" or "unmanaged." Managed external adapters provide an adapter configuration file (`.cnxml`) that can be configured using Studio, referenced in a CCL **ATTACH ADAPTER** statement, and can be started and stopped by the Server, behaving very similarly to an internal adapter. Unmanaged external adapters are not referenced in a CCL **ATTACH ADAPTER** statement, and are not managed by the Server. You start, stop, and configure these adapters independently.

CHAPTER 2 Adapters Currently Available from SAP

SAP provides a variety of internal and external adapters.

Unless otherwise noted, these adapters support the same platforms and operating systems as the Server and Studio. See the *Event Stream Processor Installation Guide*.

Adapter Summary

Summary of adapters currently available from SAP.

Some adapters do not come installed with the product and require additional licenses. For a list of these adapters, see the *Installation Guide*.

Adapter	Type	Managed	Studio Configurable	Supports Guaranteed Delivery
Adaptive Server® Enterprise Output	Internal	Yes	Yes	No
AtomReader Input	Internal	Yes	Yes	No
Database Input	Internal	Yes	Yes	No
Database Output	Internal	Yes	Yes	No
ESP Add-In for Microsoft Excel	External	No	No	No
ESP Web Services Provider	External	No	Partially	No
File CSV Input	External	Yes	Yes	No
File CSV Output	External	Yes	Yes	No
File FIX Input	Internal	Yes	Yes	No
File FIX Output	Internal	Yes	Yes	No
File JSON Input	External	Yes	Yes	No
File JSON Output	External	Yes	Yes	No

Adapter	Type	Managed	Studio Configurable	Supports Guaranteed Delivery
File XML Document Input	External	Yes	Yes	No
File XML Document Output	External	Yes	Yes	No
File XML Record Input	External	Yes	Yes	No
File XML Record Output	External	Yes	Yes	No
FIX Input	External	Yes	Yes	No
Flex Output	External	No	No	No
FTP CSV Input	External	Yes	Yes	No
FTP CSV Output	External	Yes	Yes	No
FTP XML Input	External	Yes	Yes	No
FTP XML Output	External	Yes	Yes	No
HTTP Output	External	Yes	Yes	No
JDBC Input	External	Yes	Yes	No
JDBC Output	External	Yes	Yes	No
JMS CSV Input	External	Yes	Yes	No
JMS CSV Output	External	Yes	Yes	No
JMS FIX Input	Internal	Yes	Yes	Yes
JMS FIX Output	Internal	Yes	Yes	Yes
JMS Object Array Input	External	Yes	Yes	No
JMS Object Array Output	External	Yes	Yes	No
JMS XML Input	External	Yes	Yes	No
JMS XML Output	External	Yes	Yes	No
KDB Input	External	Yes	Yes	No
KDB Output	External	Yes	Yes	No
Log File Input	External	No	No	No
NYSE Technologies Input	External	Yes	Yes	No

Adapter	Type	Managed	Studio Configurable	Supports Guaranteed Delivery
Open Input	External	No	No	No
Open Output	External	No	No	No
Random Tuples Generator Input	Internal	Yes	Yes	No
RAP Output	External	No	No	No
Replication Server® Input	External	Yes	Yes	Yes
Reuters Marketfeed Input	External	No	No	No
Reuters Marketfeed Output	External	No	No	No
Reuters OMM Input	External	No	No	No
Reuters OMM Output	External	No	No	No
RTView Output	External	No	No	No
Sample Input	External	Yes	Yes	No
Sample Output	External	Yes	Yes	No
SAP HANA Output	Internal	Yes	Yes	No
SAP RFC Input	External	Yes	Yes	No
SAP RFC Output	External	Yes	Yes	Yes
SAP Sybase IQ Output	Internal	Yes	Yes	No
SMTP Output	Internal	Yes	Yes	No
Socket FIX Input	Internal	Yes	Yes	No
Socket FIX Output	Internal	Yes	Yes	No
Socket CSV Input	External	Yes	Yes	No
Socket CSV Output	External	Yes	Yes	No
Socket JSON Input	External	Yes	Yes	No
Socket JSON Output	External	Yes	Yes	No
Socket XML Input	External	Yes	Yes	No

Adapter	Type	Managed	Studio Configurable	Supports Guaranteed Delivery
Socket XML Output	External	Yes	Yes	No
Tibco Rendezvous Input	External	Yes	No	Yes
Tibco Rendezvous Output	External	Yes	No	Yes
Web Services (SOAP) Input	External	Yes	Yes	Yes
Web Services (SOAP) Output	External	Yes	Yes	Yes
WebSphere MQ Input	Internal	Yes	Yes	Yes
WebSphere MQ Output	Internal	Yes	Yes	Yes

Adaptive Server Enterprise Output Adapter

Adapter type: SAP_ase_out. The Adaptive Server Enterprise (ASE) Output adapter reads data from an Event Stream Processor stream and writes it to the Adaptive Server.

The ASE Output Adapter is statically linked to the Open Client™ libraries, it uses the bulk insert API from the Open Client Bulk-Library to optimally load data into the Adaptive Server database. It supports parallel load for tables that are partitioned using round-robin partitioning, but does not support guaranteed delivery. Under normal operation of the system, all Event Stream Processor messages are stored in the database. In a production environment, the Adaptive Server database server runs on a separate machine than the Event Stream Processor engine. Network bandwidth can affect message flow throughput.

The stream or window attached to the ASE Output adapter might have a schema where the order or names of the columns are different from the order or names of the columns in the target Adaptive Server Enterprise table. In such a case, specify the optional **permutation** adapter parameter, which describes the mapping between the stream or window columns and the columns in the database. All non-nullable columns that do not have a default value defined in the target Adaptive Server database table must have a mapping in the **permutation** parameter. For nullable columns that do not have a mapping provided, a default value is used, if one is defined in the database; otherwise, a null value is used.

When the adapter gets a request to shutdown, it first processes the data it received before that request. The adapter gets the shutdown signal either when the project is stopping, the ESP server is shutting down, or the adapter itself is stopped. Because the amount of time it takes for the adapter to shutdown depends on the amount of data being processed, the adapter may appear to be slow shutting down.

By default, the adapter performs inserts, updates, and deletes. To perform only inserts, or to change all updates to inserts, use the optional **dataWarehouseMode** adapter parameter.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Here is a sample service entry for the ASE Output adapter to connect to the Adaptive Server database using an Open Client connection:

```
<Service  Name="ASE"  Type="DB">
    <Parameter Name="DriverLibrary">esp_db_ocs_lib</Parameter>
    <Parameter Name="Host">raprh5dev</Parameter>
    <Parameter Name="Port">15000</Parameter>
    <Parameter Name="User">sa</Parameter>
    <Parameter Name="Password"  />
    <Parameter Name="AppName">ASEOutputAdapter</Parameter>
   </Service>
```

Property Label	Description
DB Service Name	Property ID: **service** Type: `string` (Required for adapter operation and schema discovery) The name of the service entry with connection information for the Adaptive Server database into which information is loaded. See above for a sample service entry for the adapter. Service entries are specified in the Event Stream Processor `service.xml` file. This adapter uses only Open Client drivers to connect to the Adaptive Server database. See the *Administrators Guide* for more information. No default value.
Target DB Table Name	Property ID: **table** Type: `tables` (Required for adapter operation; optional only if you are using schema discovery) A string value representing the name of the Adaptive Server database table to load data into. No default value.
Include Base Content	Property ID: **outputBase** Type: `boolean` (Optional) Determines whether to process the base data of the window or stream to which the adapter is connected. The adapter processes the base data of the window or stream to which it is attached. Default value is false.

Property Label	Description
Only Base Content	Property ID: **onlyBase** Type: `boolean` (Advanced) If true, the adapter processes only the base data of the window or stream to which it is attached. No further message flow is processed. Default value is false.
Field Mapping	Property ID: **permutation** Type: `permutation` (Advanced) The permutation list maps the Event Stream Processor column names to the database schema column names in the specified table. If you do not specify a permutation, ensure that the Event Stream Processor stream or window columns exactly match the database schema of the destination table. For example, both must have the same order, same number of columns, and compatible datatypes. If the Data Warehouse Mode property is OFF, and you want to specify a permutation, include a mapping for at least one column of the primary key of the ESP window attached to the adapter. Without mapping at least one primary key column, the adapter fails to start. The format for this property is: <esp_columnname>=<database_columnname>:<esp_columnname>=<database_columnname> Use a colon to separate mappings. No default value.
Bulk Batch Size	Property ID: **bulkBatchSize** Type: `int` (Advanced) Specifies the number of rows that can be inserted in a table partition before a commit occurs. Value must be a multiple of the **bulkInsertArraySize** parameter. Default value is 10000.
Bulk Insert Array Size	Property ID: **bulkInsertArraySize** Type: `int` (Advanced) Specifies the number of rows that are simultaneously inserted into a table partition. Default value is 1000.

Property Label	Description
Idle Buffer Write Delay in Msec	Property ID: **idleBufferWriteDelayMSec** Type: `int` (Advanced) Specifies the number of milliseconds that a database table partition may sit idle with uncommitted data available for insert. Default value is 1000.
Buffer Age Limit in Msec	Property ID: **bufferAgeLimitMSec** Type: `int` (Advanced) Forces the loading of any data that has been in existence longer than the time limit. Specify a value in milliseconds between 1 and 65535. Default value is 10000.
Data Warehouse Mode	Property ID: **dataWarehouseMode** Type: `choice` (Advanced) Specifies the type of data warehousing mode the adapter uses. Valid values are: • • **ON –** updates are converted to inserts, and deletes are ignored. • • **INSERTONLY –** only inserts are processed, and updates and deletes are ignored. • • **OFF –** (default) all inserts, updates and deletes are processed as such. If you want to specify a field mapping, or permutation, map at least one column of the primary key of the ESP window attached to the adapter. Without mapping of at least one primary key column, the adapter fails to start.
Timestamp Column Name	Property ID: **timestampColumnName** Type:`string` (Advanced) If an Adaptive Server table column name is provided, the time at which the record is added to the bulk array is stored in that column of the Adaptive Server record. If this property is empty, there is no timestamp stored. The timestamp is always in UTC rather than the ESP Server's local time zone. No default value.

Property Label	Description
Timezone For Statistics	Property ID: **timezoneForStats** Type: `string` (Advanced) Controls the time zone in which midnight statistics for the adapter roll over. For example, if the adapter is left at its default value of GMT, midnight statistics reset at midnight GMT. This setting does not affect any other dates processed or returned by this adapter. Default value is GMT.

Custom Statistics

The Adaptive Server Enterprise Output adapter maintains custom statistics to show its status and to track its loading activities. Enable the time granularity option in the project configuration (`CCR`) file to get these statistics reported by the _ESP_Adapter_Statistics metadata stream:

Rows	Bytes	Average Rows
• Received in the last hour • Received since midnight • Loaded in the last hour • Loaded since midnight • Skipped* in the last hour • Skipped* since midnight	• Received since midnight • Loaded since midnight • Skipped since midnight	• Per second received over the last minute • Per second received over the last hour

* A row is skipped when the opcode does not match the adapter's current warehousing mode. For example, if the data warehousing mode is INSERTONLY, and the adapter passes in a delete or an update, this results in a skipped row.

The "since midnight" statistics reset after midnight in the time zone specified by the **timezoneForStats** adapter property. If no value is set, the default behavior is reset at midnight GMT.

See also

- *Adapter Support for Schema Discovery* on page 1005

Datatype Mapping for the ASE Output Adapter

Event Stream Processorr datatypes map to Adaptive Server datatypes.

Event Stream Processor does not have datatypes equivalent to these Adaptive Server datatypes:

- `tinyint`
- `smallint`
- `unsigned smallint`
- `unsigned int`
- `unsigned bigint`
- `smallmoney`

To use existing Adaptive Server schemas containing these datatypes, SAP supports mapping of `integer`, `long`, `money` and `money(1-15)` Event Stream Processor datatypes to those Adaptive Server Enterprise datatypes.

During message flow, the adapter verifies that incoming values can fit into the specified Adaptive Server Enterprise columns without losing precision, and rejects the rows if they cannot. Any ASE column datatypes that are marked with an asterisk have data dependent limitations on the mapping, and only these mappings are verified for each record flowing through the adapter. Due to the performance impact of runtime checks on each record, SAP recommends you do not use these mappings unless you cannot make changes to existing schemas.

Event Stream Processor Datatypes	Adaptive Server Enterprise Datatypes	Notes
`integer`	`tinyint*, smallint*, unsigned smallint*, int, bigint, money, numeric` and `decimal` datatypes with precision minus scale equals or is greater than 10	
`long`	`unsigned int*, bigint, unsigned bigint*, numeric` and `decimal` datatypes with precision minus scale equals or is greater than 19	

Event Stream Processor Datatypes	Adaptive Server Enterprise Datatypes	Notes
`float`	`double precision, numeric, decimal`	There is no required precision or scale for `float` to `numeric` or `decimal` mapping during adapter initialization. However, during message flow, any rows that generate an error occur while data is converting from Event Stream Processor `float` to Adaptive Server Enterprise `numeric` or `decimal` is rejected. This generally occurs due to insufficient precision of the target `numeric` and `decimal` datatype.
`date`	`date, smalldatetime, datetime, bigdatetime`	
`timestamp`	`bigdatetime`	
`bigdatetime`	`time, bigtime, bigdatetime, datetime`	
`string`	`char(n), varchar(n)`	If the destination column is not large enough to store the `string` value, the value is truncated to fit the column and a warning is written in the ESP Server logs.

Event Stream Processor Datatypes	Adaptive Server Enterprise Datatypes	Notes
`boolean`	`bit, tinyint*, smallint*, int, bigint, unsigned smallint*, unsigned int*, unsigned bigint*, varchar(5)`	For `boolean` values that map to `varchar(5)` columns, the adapter inserts the text "true" or "false" into the database table. For boolean values that map to various integer database columns, the adapter inserts values 1 or 0.
`money`	`smallmoney*, money, numeric*` and `decimal*` datatypes with scale >= 4	
`money(n)`	`numeric(p,s)*` and `decimal(p,s)*` datatypes with scale >= n	
`interval`	`bigint`	
`binary`	`binary(n), varbinary(n)`	

Error Handling for the Adaptive Server Enterprise Output Adapter

Various scenarios that write error messages in the ESP Server log.

Scenario	Result
• The database service specified by the **service** parameter is not for an Adaptive Server Enterprise database. • The connection information for the database is invalid. • The database table specified by the **table** parameter does not exist. • The **permutation** parameter is specified but one of the columns does not exist in the database table. • The database column datatype is not supported for a given Event Stream Processordatatype. • You did not specify the **permutation** parameter, and the stream or window attached to the Adaptive Server Enteprise Output adapter has a schema where the order, names, or count of the columns are different from the order, names, or count of the columns in the target database table. • The database contains a column that is not nullable and has no default value defined in the database, and a column mapping does not exist in the permutation for that column.	The Adaptive Server Enterprise Output adapter fails to initialize and logs an error message to the Server log file.
• A row contains `integer` or `long` or `money` values that are too large to store in a given Adaptive Server Enterprise column datatype. • A row has a `float` value that fails to convert to the target `numeric` or `decimal` value.	During message flow: • The row is rejected. • The bad rows statistic is incremented. • An error message is logged to the Server log file.

Scenario	Result
• An Event Stream Processor `string` value is longer than the `varchar` or `char` database column into which it is to be stored.	During message flow: • The adapter truncates the string value. • A warning message is logged to the Server log. • The warning messages in the Server log file do not contain any data from the rows truncated during message flow.
• An Event Stream Processor `binary` value is longer than the `varbinary` or `binary` database column into which it is to be stored.	During message flow: • The adapter rejects the row. • An error message is logged to the Server log. • The error messages in the Server log file do not contain any data from the rows rejected during message flow.
• A row contains a duplicate primary key of a row that is already in the database.	The given row and all other rows in the current batch are rejected and are not inserted into the database.
• A batch contains a bad row.	During message flow: • The individual row is discarded. • The bad rows statistic is incremented. • An error message is logged to the Server log file.

AtomReader Input Adapter

Adapter type: atomreader_in. The AtomReader Input adapter allows you to receive information from ATOM datasources.

ATOM datasources enable connections through URLs and transmit their information in a specialized XML format. Information the adapter receives from an ATOM datasource is inserted into an Event Stream Processor stream.

Ensure that incoming XML information includes:

- feed_title
- feed_link
- feed_author_name
- entry_title
- entry_link
- entry_content

Note: The adapter ignores any additional fields in the XML file.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Source URL	Property ID: **URL** Type: `string` (Required) The URL of the ATOM datasource.
Refresh Interval	Property ID: **refreshInterval** Type: `interval` (Optional) Determines how often the specified URL is queried for data. The adapter measures the interval in microseconds unless qualified with interval formatting. Default value is 60000 milliseconds.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) The format string for parsing timestamp values. Default value is %Y-%m-%dT%H:%M:%S.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) The format string for parsing date values. Default value is %Y-%m-%dT%H:%M:%S.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Database Adapter

Event Stream Processor provides an input and output database adapter. The Database Input adapter receives data from a database table, and the Database Output adapter sends data to a database table.

You can use several different JDBC™ and ODBC drivers with the database adapter. To use the ODBC drivers, ensure you have a driver manager installed.

On UNIX systems, Event Stream Processor expects your ODBC driver manager library to be called `libodbc.so.1`. Ensure that your driver manager library has this name or create a symbolic link from `libodbc.so.1` to your ODBC driver manager library.

If you are connecting to a database other than SAP HANA, SAP recommends upgrading to version 2.3.0 or later of unixODBC. If you are using a version lower than 2.3.0, set a parameter for the driver that instructs the database manager not to synchronize database access. To do this, add a line that says "Threading = 0" for your driver in the `odbcinst.ini` file.

If you are connecting to an SAP HANA database, use unixODBC version 2.3.0 or later - do not use an earlier version. In addition:

- If you are using version 2.3.0, add "Threading=0" in the `odbcinst.ini` file to ensure optimal adapter performance.
- If you are using version 2.3.1, create a symbolic link under `<2.3.1 installation folder>/lib` as follows:

  ```
  ln -s libodbc.so.2.0.0 libodbc.so.1
  ```

  As stated previously, Event Stream Processor expects your ODBC driver manager library to be called `libodbc.so.1`, which unixODBC 2.3.1 has renamed `libodbc.so.2`.

See also

- *Adapter Support for Schema Discovery* on page 1005

Database Input Adapter

Adapter type: db_in. The Database Input adapter receives data from a database table.

You can use the adapter to periodically poll the table and receive updates. The required properties depend on the database type you are connecting to. The supported databases for JDBC are Adaptive Server Enterprise, Microsoft SQL Server, IBM DB2, Oracle, and KDB. The supported databases for ODBC are Adaptive Server Enterprise, Microsoft SQL Server, IBM DB2, Oracle, SAP Sybase IQ, SQL Anywhere®, TimesTen, MySQL 5.x, and PostgreSQL.

The `service.xml` file contains service definitions and the properties required for a database connection. For the service definition name, consult the person responsible for

configuring and maintaining the `service.xml` file. See the *Adminstrators Guide* for information on configuring database connections using the `service.xml` file.

Use the **query** property to override the table selection and get data from an arbitrary query. This adapter supports schema discovery.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Important: For an Adaptive Server Enterprise database, enable the **ddl in tran** option on the temporary database (tempdb) to discover all tables when using schema discovery. Then update the Server by performing a checkpoint on tempdb or restarting the database instance. For more information on the **ddl in tran** option, consult your Adaptive Server documentation.

Property Label	Description
Database Service	Property ID: **service** Type: `string` (Required for adapter operation and schema discovery) Name of database service as defined in the `service.xml` file. No default value.
Database Query	Property ID: **query** Type: `string` (Optional) The SQL query to be executed by the adapter. No default value. The adapter definition requires either **query** or **table** to be defined. If both parameters are defined, the **query** parameter is used.
Input Table Name	Property ID: **table** Type: `tables` (Optional) A string value representing the name of the table to read. No default value.
Poll Period (in seconds)	Property ID: **pollperiod** Type: `uint` (Advanced) Period for polling for new contents, in seconds. Default value is 0, which means no polling.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Format string for parsing date values. Default value is `%Y-%m-%dT%H:%M:%S`.

Property Label	Description
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Format string for parsing timestamp values. Default value is `%Y-%m-%dT%H:%M:%S`.

Field Mapping	Property ID: **permutation**
	Type: `permutation`
	Mapping between Event Stream Processor and external fields, for example:
	<esp_columnname>=<database_columnname>:<esp_columnname>=<database_columnname>. No default value.
	For Oracle 11g, DB2 9.7, and SAP HANA, the metadata services return results in uppercase, so ensure the database column key in the permutation is in uppercase. For example, in CCL, this command does not work:
	`permutation= 'Subject=subject:c_string=c_string'`
	but this one does:
	`permutation= 'Subject=SUBJECT:c_string=C_STRING'`
	For Adaptive Server Enterprise 15.5 and Microsoft SQL Server 2008, the metadata results are the same as the case in the defined column name, unless the name is modified in the **SELECT** statement.

PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

The Database Input adapter has these limitations:

- Ensure, when polling, that this is the only adapter.
- Any data updates received from any other source are undone on the next poll.

Database Output Adapter

Adapter type: db_out. The Database Output adapter sends data to a database table.

You can truncate the table when the adapter starts using the **truncateTable** property. The required properties depend on the database type you are connecting to. The supported databases for JDBC are Adaptive Server Enterprise, Microsoft SQL Server, IBM DB2, Oracle, and KDB. The supported databases for ODBC are Adaptive Server Enterprise, Microsoft SQL Server, IBM DB2, Oracle, SAP Sybase IQ, SQL Anywhere, TimesTen, MySQL 5.x, and PostgreSQL.

The `service.xml` file contains service definitions and the properties required for a database connection. For the service definition name, consult the person responsible for configuring and maintaining the `service.xml` file. See the *Administrators Guide* for more information on configuring database connections using the `service.xml` file.

Attention: The Oracle ODBC driver does not support SQL_C_SBIGINT/SQL_C_UBIGINT parameters, causing errors when the Database Output adapter tries to write `long` and `interval` Event Stream Processor types to `bigint` type columns. To successfully use the Oracle and the TimesTen ODBC drivers with the Database Output adapter, add this parameter "`<Parameter Name ="WriteBigIntAsChar" > true < /Parameter >`" to the `service.xml` file.

An example of specifying a different date format is when inserting a date column into an Oracle Date column. The default Oracle date format is: `04-Apr-1964 17:12:00`, so you specify that the **dateFormat** parameter is `d-%b-%Y %H:%M:%S`.

Important: Enable the "Server side prepare" option in the ODBC configuration to ensure that the Database Output adapter writes successfully to the PostgreSQL database using the ODBC driver. For more information on this option, consult your ODBC documentation.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Important: For an Adaptive Server Enterprise database, enable the **ddl in tran** option on the temporary database (tempdb) to discover all tables when using schema discovery. Then update the Server by performing a checkpoint on tempdb or restarting the database instance. For more information on the **ddl in tran** option, consult your Adaptive Server documentation.

Property Label	Description
Database Service	Property ID: **service** Type: `string` (Required for adapter operation and schema discovery) Name of database service as defined in the `service.xml` file. No default value.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Format string for parsing date values. Default value is `%Y-%m-%dT%H:%M:%S`.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Format string for parsing timestamp values. Default value is `%Y-%m-%dT%H:%M:%S`.

Field Mapping	Property ID: **permutation**
	Type: `permutation`
	(Advanced) Maps the internal ESP fields to the external application or display fields. No default value.
	For Oracle 11g, DB2 9.7, and SAP HANA, the metadata services return results in uppercase, so ensure the database column key in the permutation is in uppercase. For example, in CCL, this command does not work:
	`permutation= 'Subject=sub-ject:c_string=c_string'`
	but this one does:
	`permutation= 'Subject=SUB-JECT:c_string=C_STRING'`
	For Adaptive Server Enterprise 15.5 and Microsoft SQL Server 2008, the metadata results are the same as the case in the defined column name, unless the name is modified in the **SELECT** statement.

Property Label	Description
Only Base Content	Property ID: **onlyBase** Type: `boolean` (Advanced) Send only the initial contents of the stream, once. Default value is false.
Batch Limit	Property ID: **batchLimit** Type: `uint` (Advanced) Number of records to process as a batch. Default value is 1. Using UPSERT with batch processing may negatively impact performance since this process may be terminated if the adapter receives a delete. The resolution of an UPSERT to either INSERT or UPDATE based on stream content gives you less control over the grouping of these operations. However, frequently changing between operations (INSERT, UPDATE, DELETE, and UPSET) reduces the optimization of using batch processing.
Data Location	Property ID: **datalocation** Type: `string` (Advanced) Looks up properties in the project configuration. No default value.
Output Table Name (runtime)	Property ID: **table** Type: `tables` (Optional) A string value representing the name of the table to push data to. No default value.
Include Base Content	Property ID: **outputBase** Type: `boolean` (Optional) Output initial stream contents in addition to stream updates. Default value is false.
Truncate the Database Table	Property ID: **truncateTable** Type: `boolean` (Optional) Truncate the database table, then populate with streaming data. Default value is false.

Property Label	Description
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

The Database Output adapter has these limitations:

- The output table must exist.
- Each row translates to an SQL statement, therefore updates are slow.
- If you are using a memory store, you can perform only UPSERT, UPDATE, and DELETE on data that is in the stream.
- If a batch contains a bad row, the entire batch is discarded, the bad rows statistic is incremented, and an error message is logged to the Server log file.

Datatype Mapping for the Database Adapter

Mapping between Event Stream Processor datatypes and SAP Adaptive Service Enterprise, Microsoft SQL Server, IBM DB2, Oracle, and KDB datatypes.

Datatype Mapping: SAP HANA

Mapping between Event Stream Processor datatypes and SAP HANA® datatypes.

Event Stream Processor Datatype	SAP HANA Datatype
`integer`	`int`
`long`	`bigint`
`float`	`double`
`date`	`seconddate`
`string`	`varchar(n)`
`money`	`decimal(19,4)`
`timestamp`	`timestamp`

Event Stream Processor Datatype	SAP HANA Datatype
boolean	tinyint (supports null values)
money1	decimal(19,1)
money2	decimal(19,2)
money3	decimal(19,3)
money4	decimal(19,4)
money5	decimal(19,5)
money6	decimal(19,6)
money7	decimal(19,7)
money8	decimal(19,8)
money9	decimal(19,9)
money10	decimal(19,10)
money11	decimal(19,11)
money12	decimal(19,12)
money13	decimal(19,13)
money14	decimal(19,14)
money15	decimal(19,15)
interval	bigint
bigdatetime	timestamp
binary	binary

Datatype Mapping: SAP Sybase Adaptive Server Enterprise

Mapping between Event Stream Processor datatypes and Adaptive Server Enterprise 15.5 datatypes.

Event Stream Processor Datatype	Adaptive Server Enterprise Datatype
integer	int
long	bigint
float	float

Event Stream Processor Datatype	Adaptive Server Enterprise Datatype
date	datetime
string	varchar(n)
money	money
timestamp	bigdatetime
boolean	smallint or bit (Does not support null values)
money1	numeric(19,1)
money2	numeric(19,2)
money3	numeric(19,3)
money4	numeric(19,4)
money5	numeric(19,5)
money6	numeric(19,6)
money7	numeric(19,7)
money8	numeric(19,8)
money9	numeric(19,9)
money10	numeric(19,10)
money11	numeric(19,11)
money12	numeric(19,12)
money13	numeric(19,13)
money14	numeric(19,14)
money15	numeric(19,15)
interval	bigint
bigdatetime	bigdatetime
binary	varbinary(n)

Datatype Mapping: Microsoft SQL Server Database

Mapping between Event Stream Processor datatypes and Microsoft SQL Server 2008 R2 datatypes.

Event Stream Processor Datatype	SQL Datatype
integer	int
long	bigint
float	float
date	datetime
string	varchar(n)
money	money
timestamp	datetime2
boolean	smallint or bit (Does not support null values)
money1	numeric(19,1)
money2	numeric(19,2)
money3	numeric(19,3)
money4	numeric(19,4)
money5	numeric(19,5)
money6	numeric(19,6)
money7	numeric(19,7)
money8	numeric(19,8)
money9	numeric(19,9)
money10	numeric(19,10)
money11	numeric(19,11)
money12	numeric(19,12)
money13	numeric(19,13)

Event Stream Processor Datatype	SQL Datatype
money14	numeric(19,14)
money15	numeric(19,15)
interval	bigint
bigdatetime	datetime2
binary	varbinary(n)

Datatype Mapping: IBM DB2 Database

Mapping between Event Stream Processor datatypes and IBM DB2 9.7 datatypes.

Event Stream Processor Datatype	IBM DB2 Datatype
integer	int
long	bigint
float	float
date	timestamp
string	varchar(n)
money	decimal(19,5)
timestamp	timestamp
boolean	smallint or bit (Does not support null values)
money1	decimal(19,1)
money2	decimal(19,2)
money3	decimal(19,3)
money4	decimal(19,4)
money5	decimal(19,5)
money6	decimal(19,6)
money7	decimal(19,7)

Event Stream Processor Datatype	IBM DB2 Datatype
money8	decimal(19,8)
money9	decimal(19,9)
money10	decimal(19,10)
money11	decimal(19,11)
money12	decimal(19,12)
money13	decimal(19,13)
money14	decimal(19,14)
money15	decimal(19,15)
interval	bigint
bigdatetime	timestamp
binary	blob

Datatype Mapping: Oracle Database

Mapping between Event Stream Processor datatypes and Oracle 11g datatypes.

Event Stream Processor Datatype	Oracle Datatype
integer	int
long	number(19)
float	float
date	date
string	varchar2(n)
money	number(19,4)
timestamp	timestamp
boolean	smallint or bit (Does not support null values)
money1	number(19,1)

Event Stream Processor Datatype	Oracle Datatype
money2	number(19,2)
money3	number(19,3)
money4	number(19,4)
money5	number(19,5)
money6	number(19,6)
money7	number(19,7)
money8	number(19,8)
money9	number(19,9)
money10	number(19,10)
money11	number(19,11)
money12	number(19,12)
money13	number(19,13)
money14	number(19,14)
money15	number(19,15)
interval	number(19)
bigdatetime	timestamp
binary	blob

Datatype Mapping: KDB Database

Mapping between Event Stream Processor datatypes and KDB datatypes.

Event Stream Processor Datatype	KDB Datatype
integer	int
long	long
float	float
date	datetime
string	symbol

Event Stream Processor Datatype	KDB Datatype
money	–
timestamp	datetime
boolean	boolean (Does not support null)
money1	–
money2	–
money3	–
money4	–
money5	–
money6	–
money7	–
money8	–
money9	–
money10	–
money11	–
money12	–
money13	–
money14	–
money15	–
interval	long
bigdatetime	–
binary	–

Note: Do not include Event Stream Processor datatypes in your output that do not have an equivalent KDB datatype.

ESP Add-In for Microsoft Excel

The SAP ESP Add-in for Microsoft Excel is a real-time data add-in for Microsoft Excel that lets you view and retrieve records from one or more running Event Stream Processor projects, as well as publish records to them.

The ESP Add-in for Microsoft Excel does not support Linux or Solaris platforms. You can run it on 32-bit and 64-bit editions of Windows, and with the 32-bit editions of Microsoft Excel 2007 and 2010.

On the display side, you can use the ESP Add-in for Microsoft Excel to select streams and view the columns within those streams. You can also filter records based on data values, and view the most recent “N” records, or the most recent “N” records that match a specified filter (where “N” is the specified number of records).

On the publication side, you can use the ESP Add-in for Microsoft Excel to automatically publish data whenever data changes in a range of cells. You can also manually publish data to Event Stream Processor by selecting a range of cells and using the Publication wizard.

Connection Wizard

The Connection wizard lets you simultaneously connect to one or more instances of Event Stream Processor.

The connection information for an Excel workbook is saved with the workbook.

Component	Description
Connections	Enter the name of a new connection, or select from a list of previously defined connections. When you select a connection, all the information associated with that connection appears.
Host	(Required) Enter the host name for the Event Stream Processor cluster manager to connect to.
Port	(Required) Enter the port of the Event Stream Processor cluster manager.
Workspace	Enter the workspace that the project is part of.
Project	Enter the project to connect to.
User	Enter the user name to connect to the cluster manager.
Password	(Optional) Enter the password associated with the user name.
SSL	Connect to an Event Stream Processor cluster manager that was started with SSL mode enabled.

Component	Description
Authentication Type	Select an authentication type. The authentication type selected must match the mode that was used when starting Event Stream Processor.
RSA Key File	If RSA authentication is selected, enter the RSA key file. You can either type the location and name of the key file, or click the button next to the field to browse and to choose the file.
Save	Saves the connection information in a hidden worksheet associated with the active workbook. Connection when the active workbook is retrieved and shown.
Connect	After providing all the information, click this button to connect to the server. If the connection is successful, only the Disconnect button is available for this connection. This also saves the connection information for future use.
Disconnect	Drop the connection to Event Stream Processor. On a successful disconnect, this button is disabled and the Connect and Delete buttons are enabled. Any queries that are actively using this connection are stopped, after user confirmation, before disconnecting.
Delete	Delete a connection.
Hide	Hide the window while preserving all information. To redisplay the screen, click the SAP SybaseRT button on the Excel toolbar.

Enabling Kerberos Authentication for the ESP Add-In for Microsoft Excel

Enable Kerberos Authentication for the ESP Add-In for Microsoft Excel by setting the necessary environment variables and specifying the **Authentication Type** and **User** parameters.

1. Set the following environment variables on Windows before starting the Excel application:
   a) Set the ESP_GSSAPI_LIB environment variable to point to the shared library provided by the Kerberos install. The library contains the GSSAPI function implementations.
   b) Set the PATH environment variable to include any library dependencies of the Kerberos Dynamic-Link Library (DLL).
   c) Set the ESP_SERVICE_NAME environment variable to set the service principal name.
   d) Set the KRB5_CONFIG environment variable to point to the configuration file used by the Kerberos library.

e) Set the KRB5CCNAME environment variable to point to the ticket cache.

2. Specify the following parameters under the SAP RT Wizard Connection Wizard Tab:
   - Set **Authentication Type** parameter to Kerberos.
   - Set **User** parameter.

Subscription Wizard

The Subscription wizard pane enables you to define and control one or more subscription queries, the results of which appear in the Excel worksheet.

The ESP Add-in for Microsoft Excel keeps track of the locations of your queries even if its defined cells are shifted horizontally or vertically.

Component	Description
Subscription Queries	Enter the name of a new query, or choose a previously defined query. When you select a previously saved query, you see all information associated with that query.
Connection Name	Select the connection associated with the query. You can run a query only if the associated connection is active.
Start Cell	Enter the location in the Excel worksheet in which to start inserting the real-time data formulas. Specify the location in "A1" notation; for example, a value of B5 tells the ESP Add-in for Microsoft Excel to insert the formulas as a grid starting at column B, row 5.
Max Rows	Enter the number of records (maximum value 65536) to appear in the Excel worksheet. When there are more rows to be shown than the number specified, the oldest records are discarded.
Get Base Transactions	Retrieve base records from a stream before new transactions. Leave this field unselected to retrieve only new transactions. For small tables with relatively few new transactions select this option, so you can see query data. Otherwise, the data for the query does not appear. However, for dynamic tables with high transaction activity, leave this option unselected: otherwise, Excel tries to potentially load millions of records every time it starts.

Component	Description
Lossy Subscription	To turn on the this option, click on the box to put a check in it. To leave it turned off, leave the box unchecked. This option is typically used when the network connection between the client and Event Stream Processor is slow. If this option is on, the client may not get all transactions if it cannot keep up with the Event Stream Processor. If this option is off, the client receives all transactions at the expense of potentially slowing down the Event Stream Processor, especially if the network is slow and the subscription buffer is filled up.
Streams	Displays all the streams available in the server with which the selected connection is associated. These streams automatically appear when you select a connection.
Columns	When one of the streams is selected, this area displays each column, along with its datatype and a check box to indicate whether or not it is a key column. You can choose a different key column for the stream than the specified one.
SQL Statement	To customize data retrieved from Event Stream Processor, specify a SQL statement. The statement cannot include joins, GROUP BY, and ORDER BY clauses as the SQL is applied to individual transaction logs for the stream, not the data in the stream. See the *Utilities Guide* for information about **esp_query** supported SQL syntax. This option is available only if SQL Statement is selected as the stream with which the connection is associated.
Parse SQL	Parse the contents of the SQL Statement field. If the SQL parses successfully, the column names and corresponding datatypes appear in the Columns field. By default, while none of the columns are marked as key fields, the appropriate key columns must be selected before the real-time data query is applied.
Apply	Apply the real-time data formulas in the Excel worksheet after configuring a new subscription or modifying an existing subscription. Once the formulas have been applied, you can start the query.
Reset	Display the properties of the Subscription Query when it was last saved. Select this option if changes have been made to the query that need to be completely reversed.
Delete	Delete a previously saved query.
Start	Start the query. The data appears in the Excel worksheet.

Component	Description
Stop	Stop the running query. No new data appears in the Excel worksheet. However, any data that appears continues to show until you close and reopen the worksheet, or restart the query.

Publication Wizard

The Publication wizard lets you manually publish data to a stream and graphically construct publication formulas meant for automatic publishing.

Components	Description
Connection Name	The name of the connection to use for publishing. Only active connections appear. When you click a connection, the streams the connection object is connected to appear in the Streams field, and the columns and the datatypes for the stream appear in the table named Columns.
Operation Code	If a record exists, select UPDATE, DELETE, or UPSERT (the default). Otherwise, select INSERT.
Data Range	Specify the range of cells in the Excel worksheet that contain the data to publish. You cannot edit this field directly: select the cells in the worksheet to publish, then click the blue button next to this field to populate it. You cannot simultaneously publish multiple noncontiguous areas.
WorkBook Name	A read-only field that shows the workbook in which the selected range is located.
WorkSheet Name	A read-only field that shows the worksheet in which the selected range is located.
First Row Has Columns	Indicate that the first row in the selected range has column names. If it does not, leave this field unselected. When you specify data columns, they can be in any order, and only values for the desired fields must be supplied. The rest of the columns are automatically filled with NULL. However, if you do not provide column names, the ESP Add-in for Microsoft Excel expects all the columns in the streams to be in the exact same order as defined in Event Stream Processor.
Transpose Rows To Columns	Select this option if the data columns for a record are provided vertically in a single column instead of the horizontally across multiple columns (the normal way of representing records).

Components	Description
Streams	Select the stream for which a publication should be made.
Columns	Shows the columns and the corresponding datatypes for the selected stream. You cannot select the columns in this table; however, you can copy the names of the columns and paste them in Excel.
Log File	(Optional) Specify the path and file name to which publication errors are written, either by entering it directly into the field, or by browsing to and selecting it. The errors written to this file in also appear in the Result field. Use a forward slash for both UNIX and Windows paths.
Result	Read-only results of the publication.
Publish Data	Publish the data to Event Stream Processor. The result of the publication appears in the Result field. The Event Stream Processor acknowledges only whether the data has been received. To find out whether the record was rejected for some reason, such as a duplicate insert or bad data, either subscribe to the stream or submit a SQL query.
Show Formula	Graphically create the formula. This provides a convenient way to create a formula for automatic publishing. If there are no errors, the formula appears in the Results field. You can then copy the formula and place it in the Excel worksheet to start automatically publishing the data.
Clear Results	Clear the Result field if there are too many entries.

Automatic Publishing

Use the **SybaseRTP** add-in function when the data in a cell changes

SybaseRTP is a wrapper function around the underlying Excel real-time data mechanism used for publishing data. The syntax for this formula is:

```
=SybaseRTP("ConnectionName","StreamName","OperationCode",DataRange,
[[ColumnRange],[TransposeRows],["LogFile"],[InstanceNo],
[NoResults])]
```

where:

Parameter	Description
ConnectionName	Name of the connection to use for publishing. You must establish the connection before you can publish successfully.

Parameter	Description
StreamName	Name of the stream where data is published.
OperationCode	The opcode for publishing INSERT, UPDATE, DELETE, or UPSERT.
DataRange	The address or name of the Excel range containing the data to publish. Do not enclose the DataRange object in double quotes.
[ColumnRange]	(Optional) The Excel range address or range name containing the stream column names. Do not enclose this parameter in quotes.
[TransposeRows]	(Optional) Whether the data record is specified in a column instead of a row. It can be either true or false (the default).
[LogFile]	(Optional) The name and location of the log file to which any errors are logged. If not provided, no logging is done.
[InstanceNo]	Internal use only; leave empty.
[NoResults]	Internal use only; set to false or leave empty.

For example:

```
=SybaseRTP("Connection1","Trades","INSERT",A2:E10, A1:E1,False,"C:
\logs\log1.log",,)
```

You can place this formula in any sheet in the workbook. When constructing the formula, tell Excel the workbook or worksheet to which the address refers either by selecting the appropriate cells in the desired worksheet or using the [Workbook]Worksheet!A1:E5 format.

Once the formula is in Excel, any changes made to any of the cells publishes the entire range. To publish only when certain cells are changed, use a call inside a custom wrapper that encapsulates the business logic that dictates when to call this function.

The return value for this function is an array that is formatted as a string using an Excel-style location: {{val11,val12},{val21,22}....}. You can then convert this formula into an Excel-style array object. The string contains one or more array of elements, and each subelement contains two subitems. The array string contains only one element when there are errors passed in values. Otherwise, it contains one more element than the number of rows to publish.

The first element in the array string is a summary that indicates whether errors are detected when parsing the formula, a one-element array of the form is returned; for example:

```
{{"1","Some error message."}}
```

If there are errors during record validation, or if the process is completed successfully, there is one more array element than the number of rows to publish. For example, if there are two rows to publish and both the records have been successfully published, the array string looks like the following example:

```
{{"0",""},{"0",""},{"0",""}}
```

If only one record was published successfully, and another failed, then the return array string looks like this:

```
{{"1","An error message"},{"0",""},{"1","row level error message"}}
```

Subscription Queries

You can save subscription queries permanently by saving the Excel worksheet containing the formula associated with the query. The next time the Excel Worksheet is opened, the query appears in the Subscription wizard.

Applying a Query

Apply and start a query from the Subscription Wizard pane to populate the Excel worksheet.

Prerequisites

Define a query in the Subscription Wizard pane.

Task

1. From the **Subscription Queries** menu, select the desired query.
2. Click **Apply**.
   - The ESP Add-in for Microsoft Excel first verifies that the supplied subscription query name has not already been used then verifies that the provided Start Cell is a valid Excel cell address. If either condition is false, resolve the problem.
   - Next, the ESP Add-in for Microsoft Excel constructs Excel real-time data formulas based on the specified subscription query, and inserts one formula per cell into the active worksheet. Depending on the query, hundreds of formulas may be inserted. The ESP Add-in for Microsoft Excel uses this logic to insert formulas:
     - Formulas are always inserted as a grid, starting at the specified Start Cell location. Each selected column appears in separate but contiguous columns in the Excel worksheet. The value of Max Rows controls the number of rows to which the filter is applied.
     - Soon after the first formula is inserted into the active worksheet, Excel recognizes the real-time data formula and makes a call to the ESP Add-in for Microsoft Excel server that passes the query information for the first filter. The real-time server looks at the information passed, recognizes it as a new query, and spawns a query object. The real-time data server also stores the passed-in information for future use.
     - This process is repeated for every formula of the query, except the real-time server recognizes that the formula is part of the previously seen query, and therefore, it does not create a new query object. Rather, it stores the information so that it can return the data corresponding to the formula.

3. Click **Start**.
   - The ESP Add-in for Microsoft Excel verifies that the connection to Event Stream Processor is active and that the specified query is still valid. If either of these conditions are false, then it returns to the formula.
   - Next, the ESP Add-in for Microsoft Excel spawns a new read thread to read the transaction log data from Event Stream Processor, and stores it in an internal buffer.
   - Every tenth of a second, the ESP Add-in for Microsoft Excel reads the transaction logs from the internal buffer, and decides whether to insert, update, or delete records in a display buffer, based on the user-specified key fields. When there is an insert into the display buffer and the number of records in the buffer is equal to the specified Max Rows, the oldest record in the buffer is deleted, the rest of the records are moved up, and the record is inserted at end. When a record needs to be updated, an in-place update is performed. This insert and update mechanism results in a more stable view of the data in the Excel worksheet, and makes it easier to create charts based on the subscribed data.
   - Once the display buffer has been populated, the ESP Add-in for Microsoft Excel notifies Excel that new data is available. When it receives a request for the data, it sends the data in a format that Excel can understand and shows it the appropriate location in the worksheet.

Known Issues and Limitations

The ESP Add-in for Microsoft Excel has some known issues and limitations.

- Performance degrades when Max Rows is set to a large value, for example, several thousand rows or more. The machine becomes very busy as it attempts to process and complete the request.
- When Event Stream Processor stops or the connection is lost due to network failure, the ESP Add-in for Microsoft Excel screen is not automatically refreshed to reflect the current state of the query and connection. Refresh the screen by selecting either the connection or the query to show the current state of the selected object.
- You cannot use more than one worksheet containing ESP Add-in for Microsoft Excel connection or subscription information within the same instance of Excel.

ESP Web Services Provider

The ESP Web Services Provider receives data from a Web service client and publishes it to an ESP stream or window within an ESP project.

For installation details, see the section on *Performing a Custom Installation* in the *SAP Sybase Event Stream Processor Installation Guide*. If you enable the ESP Web Services Provider using the installer, the configuration is complete and the provider is ready for use. If you did not enable the provider using the installer, see *Configuring the ESP Web Services Provider* in the *SAP Sybase Event Stream Processor Configuration and Administration Guide* for instructions on manually configuring the provider.

For instructions on exposing a project as a Web service, see *Exposing a Project as a Web Service Using the ESP Web Services Provider* in the *SAP Sybase Event Stream Processor Studio Users Guide*.

File CSV Input and Output Adapter

The File CSV Input adapter obtains CSV data from files on a local hard disk and publishes it to Event Stream Processor. The File CSV Output adapter reads rows from Event Stream Processor and writes this data into a specified CSV file.

File CSV Input Adapter Configuration

Configure the File CSV Input adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

Transporter Module: File Input Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
BufferMaxSize	Type: `integer` (Advanced) Specify the capacity of the buffer queue between the transporter module and the next module. The default value is 10240.
Parameters	(Required) Section containing the **FileInputTransporterParameters** element.
FileInputTransporterParameters	(Required) Section containng parameters for the File Input transporter.
Dir	Type: `string` (Required) Specify the absolute path to the data files which you want the adapter to read. For example, `<username>/<foldername>`. No default value. Use a forward slash for both UNIX and Windows paths.
File	Type: `string` (Required) Specify the file which you want the adapter to read or the regex pattern to filter the files on a given directory. See the **DynamicMode** parameter. No default value.

Parameter	Description
AccessMode	Type: `string` (Required) Specify an access mode. The adapter supports two modes: • **rowBased –** the adapter reads one text line at a time • **Streaming –** the adapter reads a pre-configured size of bytes into a buffer No default value.
DynamicMode	Type: `string` (Advanced) Specify a dynamic mode. The adapter supports three modes: • **Static –** the adapter reads the file specified in the **Dir** and **File** parameters • **dynamicFile –** the adapter reads the file specified in the **Dir** and **File** parameters and keeps polling the new appended content. The polling period is specified in the **PollingPeriod** parameter. • **dynamicPath –** the adapter polls all the new files under the **Dir** parameter. Also, the **File** parameter will act as a regex pattern and will filter out the necessary files. The default value is Static. If the DynamicMode has been set to dynamicPath and you leave the **File** parameter empty, the adapter reads all the files under the specified directory. An example regex pattern is ".*\.txt". This filters to only files that end with ".txt". In regex patterns, an escape character, "\", is necessary prior to the meta chars if you wish to include them in the pattern string.

Parameter	Description
PollingPeriod	Type: `integer` (Advanced) Define the period, in seconds, to poll the specified file or directory. Set this parameter only if the value of the **DynamicMode** parameter is set to dynamicFile or dynamicPath. A value of <=0 turns off polling. The default value is 0.
RemoveAfterProcess	Type: `boolean` (Optional) If this property is set to true, the file is removed from the directory after the adapter processes it. If the value of the **DynamicMode** parameter is set to dynamicFile or dynamicPath, the file is not removed after the adapter processes it. The default value is false.
ScanDepth	Type: `integer` (Optional) Specify the depth of the schema discovery. The adapter reads the number of rows specified by this parameter value when discovering the input data schema. The default value is three.

Formatter Module: CSV String to ESP Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **CsvStringToEspFormatterParameters** parameter.
CsvStringToEspFormatterParameters	(Required) Section containing the CSV String to ESP formatter parameters.
ExpectStreamNameOpcode	Type: `boolean` (Required) If set to true, the adapter interprets the first two fields of the incoming CSV line as stream name and opcode respectively. The adapter discards messages with unmatched stream names. The accepted opcodes are: • i or I: INSERT • d or D: DELETE • u or U: UPDATE • p or P: UPSERT • s or S: SAFEDELETE The default value is false.
Delimiter	Type: `string` (Advanced) The symbols used to separate the column. The default value is a comma (,).

Parameter	Description
HasHeader	Type: `boolean` (Advanced) Determines whether the first line of the file contains the description of the fields. Default value is false.
DateFormat	Type: `string` (Advanced) The format string for parsing date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.
TimestampFormat	Type: `string` (Advanced) Format string for parsing timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

ESP Connector Module: ESP Publisher

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
BufferMaxSize	Type: `integer` (Advanced) Specify the capacity of the buffer queue between the transporter module and the next module. The default value is 10240.
Parameters	(Required) Section containing the **EspPublisherParameters** parameter.

Parameter	Description
EspPublisherParameters	(Required) Section containing parameters for the ESP publisher.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream to which the adapter publishes data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
MaxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.

Parameter	Description
UseTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the File CSV Input adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.

Parameter	Description
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

See also

- *Adapter Support for Schema Discovery* on page 1005

File CSV Input Adapter Studio Properties

Adapter type: toolkit_file_csv_input. Set these properties for the File CSV Input adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Column Delimiter	Property ID: **csvDelimiter** Type: `string` (Advanced) Specify the symbol used to separate the columns.

Property Label	Description
Date Format	Propterty ID: **csvDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **csvTimestampFormat** Type: `string` (Advanced) Specify the format for parsing time-stamp values.
Has Header	Property ID: **csvHasHeader** Type: `boolean` (Advanced) Indicate whether the first line of the file contains the description of the CSV fields.
Steam name, opcode expected	Property ID: **csvExpectStreamNameOpcode** Type: `boolean` (Optional) If set to true, the adapter interprets the first two fields of the incoming CSV line as stream name and opcode. The adapter discards messages with unmatched values.
Directory	Property ID: **dir** Type: `directory` (Required) Specify the location of the data files.
File (in Directory)	Property ID: **file** Type: `string` (Required) Specify the file to read. Wildcards are allowed.
Dynamic Loading Mode	Property ID: **dynamicMode** Type: `string` (Optional) Set dynamic mode for reading files. Valid values: static, dynamicFile, dynamicPath.

Property Label	Description
Poll Period (seconds)	Property ID: **pollingPeriod** Type: `int` (Advanced) Specify the poll period when **dynamicMode** is dynamicFile or dynamicPath.
Remove Files After Processing	Property ID: **removeAfterProcess** Type: `boolean` (Optional) Removes files after they have been processed.
Scan Depth	Property ID: **scanDepth** Type: `int` (Advanced) Specify the depth of the schema discovery. The adapter reads the number of rows specified by this parameter value when discovering the input data scheme.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File for the File CSV Input Adapter

Sample adapter configuration file for the File CSV Input adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
  <Name>file_csv_input</Name>
  <Description>An adapter which gets csv data from files on local
hard disk, transforms to ESP data format, and publishes to ESP
stream.</Description>
  <Log4jProperty>./log4j.properties</Log4jProperty>
  <Modules>
```

```
    <Module type="transporter">
      <InstanceName>FileInputTransporter</InstanceName>
      <Name>FileInputTransporter</Name>
      <Next>CsvStringToEspFormatter</Next>
      <BufferMaxSize>10240</BufferMaxSize>
      <Parameters>
          <FileInputTransporterParameters>
              <Dir>./data</Dir>
              <File>input.csv</File>
              <AccessMode>rowBased</AccessMode>
              <RemoveAfterProcess>false</RemoveAfterProcess>
              <ScanDepth>5</ScanDepth>
          </FileInputTransporterParameters>
      </Parameters>
    </Module>

    <Module type="formatter">
      <InstanceName>CsvStringToEspFormatter</InstanceName>
      <Name>CsvStringToEspFormatter</Name>
      <Next>MyInStream_Publisher</Next>
      <Parallel>true</Parallel>
      <Parameters>
           <CsvStringToEspFormatterParameters>
               <ExpectStreamNameOpcode>true</ExpectStreamNameOpcode>
           </CsvStringToEspFormatterParameters>
      </Parameters>
    </Module>

    <Module type="espconnector">
      <InstanceName>MyInStream_Publisher</InstanceName>
      <Name>EspPublisher</Name>
      <Parameters>
        <EspPublisherParameters>
          <ProjectName>EspProject1</ProjectName>
          <StreamName>BaseInput</StreamName>
          <MaxPubPoolSize>1</MaxPubPoolSize>
          <UseTransactions>false</UseTransactions>
          <SafeOps>true</SafeOps>
          <SkipDels>true</SkipDels>
        </EspPublisherParameters>
      </Parameters>
      <BufferMaxSize>10240</BufferMaxSize>
    </Module>

  </Modules>

  <EspProjects>
    <EspProject>
      <Name>EspProject1</Name>
      <Uri>esp://localhost:19011/sample_workspace/file_csv_input</
Uri>
      <Security>
        <User></User>
        <Password encrypted="false"></Password>
        <AuthType>user_password</AuthType>
      </Security>
```

```
    </EspProject>
  </EspProjects>
  <GlobalParameters></GlobalParameters>
</Adapter>
```

File CSV Output Adapter Configuration

Configure the File CSV Output adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

ESPConnector Module: ESP Subscriber

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, espconnector.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.

Parameter	Description
Parameters	(Required) Section containing the **EspSubscriberParameters** parameter.
EspSubscriberParameters	(Required) Section containing parameters for the ESP subscriber.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Specifies the unique project tag of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream from which the adapter subscribes to data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.

Formatter Module: ESP to CSV String Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.

Parameter	Description
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **EspToCsvStringFormatterParameters** parameter.
EspToCsvStringFormatterParameters	(Required) Section containing parameters for the ESP to CSV String formatter parameters.
PrependStreamNameOpcode	Type: `boolean` (Optional) If set to true, the adapter prepends the stream name and the opcode in each row of data that is generated. The default value is false.
Delimiter	Type: `string` (Advanced) The symbols used to separate the column. The default value is a comma (,).
HasHeader	Type: `boolean` (Advanced) Determines whether the first line of the file contains the description of the fields. The default value is false.
DateFormat	Type: `string` (Advanced) The format string for date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.

Parameter	Description
TimestampFormat	Type: `string` (Advanced) Format string for timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

Transporter Module: File Output Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **FileOutputTransporterParameters** parameter.
FileOutputTransporterParameters	(Required) Section containing the parameters for the File Output transporter.
Dir	Type: `string` (Required) Specify the absolute path to the data files that you want the adapter to write to. For example, `<username>/<foldername>`. The default value is ".", meaning the current directory in which the adapter is running. Use a forward slash for both UNIX and Windows paths.

Parameter	Description
File	Type: `string` (Required) Specify the file to which you want the adapter to write.
AccessMode	Type: `string` (Required) Specify an access mode. The adapter supports two modes: • **rowBased –** the adapter writes one text line at a time into the file • **Streaming –** the adapter writes the raw data in ByteBuffer into the file No default value.
AppendMode	Type: `boolean` (Optional) If set to true, the adapter appends the data into the existing file. If set to false, the adapter overwrites existing content in the file. Default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the File CSV Output adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.

Parameter	Description
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.

Parameter	Description
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

File CSV Output Adapter Studio Properties

Adapter type: toolkit_file_csv_output. Set these properties for the File CSV Output adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Directory	Property ID: **dir** Type: `directory` (Required) Specify the location of the data files.
File (in Directory)	Property ID: **file** Type: `string` (Required) Specify the file to write to.
Append Mode	Property ID: **appendMode** Type: `boolean` (Advanced) Whether data should be appended to an existing file.
Column Delimiter	Property ID: **csvDelimiter** Type: `string` (Advanced) Specify the symbol used to separate the columns.
Date Format	Propterty ID: **csvDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **csvTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.

Property Label	Description
Has Header	Property ID: **csvHasHeader** Type: `boolean` (Advanced) Indicate whether the first line of the file contains the description of the CSV fields.
Prepend stream name, opcode	Property ID: **csvPrependStreamNameOpcode** Type: `boolean` (Optional) If set to true, the adapter prepends the stream name and the opcode in each row of data that is generated.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File for the File CSV Output Adapter

Sample adapter configuration file for the File CSV Output adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>file_xmllist_output</Name>
   <Description>An adapter which transforms ESP data to csv format,
and save to file on local hard disk.</Description>
    <Log4jProperty>./log4j.properties</Log4jProperty>
    <Modules>
        <Module type="espconnector">
            <InstanceName>MyOutStream_Subscriber</InstanceName>
            <Name>EspSubscriber</Name>
            <Next>EspToCsvStringFormatter</Next>
            <Parameters>
                <EspSubscriberParameters>
                    <ProjectName>EspProject2</ProjectName>
                    <StreamName>BaseOutput</StreamName>
                </EspSubscriberParameters>
            </Parameters>
```

```
        </Module>

        <Module type="formatter">
            <InstanceName>EspToCsvStringFormatter</InstanceName>
            <Name>EspToCsvStringFormatter</Name>
            <Next>FileOutputTransporter</Next>
            <Parallel>true</Parallel>
            <Parameters>
                <EspToCsvStringFormatterParameters>
                    <PrependStreamNameOpcode>true</
PrependStreamNameOpcode>
                </EspToCsvStringFormatterParameters>
            </Parameters>
        </Module>

        <Module type="transporter">
            <InstanceName>FileOutputTransporter</InstanceName>
            <Name>FileOutputTransporter</Name>
            <Parameters>
                <FileOutputTransporterParameters>
                    <Dir>./data</Dir>
                    <File>output.csv</File>
                    <AccessMode>rowBased</AccessMode>
                    <AppendMode>false</AppendMode>
                </FileOutputTransporterParameters>
            </Parameters>
        </Module>
    </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject2</Name>
            <Uri>esp://localhost:19011/sample_workspace/
file_csv_output</Uri>
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Adapter Controller Parameters

The Adapter Controller port listens for commands. The `controller.xml` file is located in `%ESP_HOME%/adapters/framework/config` directory. This file is shared among all the adapter within the `%ESP_HOME%/adapters/framework/instances` directory.

Parameter	Description
ControlPort	(Required) Section containing the **MinPort** and **MaxPort** parameters.
MinPort	Type: `int` (Required) The minimum port number that can act as a control port number. The adapter framework allocates this number automatically. A recommended value is 19082.
MaxPort	Type: `int` (Optional) The maximum port number that can act as a control port number. The adapter framework allocates this number automatically. The default value is 65535.
Statistics_Update_Interval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework publishes statistics data to the ESP project to which the adapter is connected. A recommended value is 5000.
MonitorInterval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework checks the status of internal resource allocation. The framework logs a message in the log file if a resource is about to be exhausted. A recommended value is 3000.

Logging

The File CSV Input and Output adapters use the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing

the logging configuration is located in the `%ESP_HOME%\adapters\framework\config` directory.

Specify the location of the logging file you wish to use in the **Log4jProperty** parameter within the adapter configuration file. You can modify the logging levels within this file or the `$ESP_HOME/adapters/framework/config/log4j.properties`, which is used by default. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
```

```
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Starting the File CSV Adapter

To start the File CSV adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

   ```
   cd %ESP_HOME%\cluster\nodes\node1
     %ESP_HOME%\bin\esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   %ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

   UNIX:

a. Start the example cluster.

```
cd $ESP_HOME/cluster/nodes/node1
  $ESP_HOME/bin/esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
$ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

2. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: For the File CSV Input adapter: `cd $ESP_HOME/adapters/framework/instances/file_csv_input` For the File CSV Output adapter: `cd $ESP_HOME/adapters/framework/instances/file_csv_output` `./start_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the File CSV Input adapter: `cd %ESP_HOME%/adapters/framework/instances/file_csv_input` For the File CSV Output adapter: `cd %ESP_HOME%/adapters/framework/instances/file_csv_output` `start_adapter.bat <adapter configuration file name>`

Stopping the File CSV Adapter

To stop the File CSV adapter from the command line, execute the **stop** command.

Operating System	Step
UNIX	Open a terminal window and enter: For the File CSV Input adapter: `cd $ESP_HOME/adapters/framework/instances/file_csv_input` For the File CSV Output adapter: `cd $ESP_HOME/adapters/framework/instances/file_csv_output` `./stop_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the File CSV Input adapter: `cd %ESP_HOME%/adapters/framework/instances/file_csv_input` For the File CSV Output adapter: `cd %ESP_HOME%/adapters/framework/instances/file_csv_output` `stop_adapter.bat <adapter configuration file name>`

File FIX Input Adapter

Adapter type: fixfile_in. The File FIX Input adapter reads FIX messages from a file and writes them as stream records.

Each stream hosts FIX messages of a certain type. The adapter ignores messages of any other FIX type. It writes all FIX fields, except the following, in the same order in stream columns:

- BeginString
- BodyLength
- MsgType
- CheckSum

Ensure the names of the stream columns correspond to the FIX protocol specification.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
FIX Version	Property ID: **fixVersion** Type: `choice` (Required) Version of the FIX protocol. Default value is 4.2.
FIX Message Type	Property ID: **fixMessageType** Type: `string` (Required) Type of messages hosted by the stream. No default value.
File	Property ID: **fileName** Type: `filename` (Required) Path to the input file containing FIX messages. No default value. Use a forward slash for both UNIX and Windows paths.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Date format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Timestamp format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Known limitations:

- This adapter is not a full FIX engine.
- Supports only FIX versions 4.2, 4.3, 4.4, and 5.0.
- Does not support repeating groups and components.
- Supports only INSERT opcode.

See also

- *FIX Input Adapter* on page 162

Datatype Mapping for the File FIX Input Adapter

Event Stream Processor datatypes map to FIX datatypes.

Event Stream Processor Datatype	QuickFix Datatype
`integer`	`boolean`
`string`	`byte[]`
`string`	`char`
`string`	`string`
`date`	`date`
`float`	`float`
`integer`	`integer`
`date` or `timestamp`	`UTCDateOnly`
`date` or `timestamp`	`UTCTimeOnly`
`date` or `timestamp`	`UTCTimeStamp`

File FIX Output Adapter

Adapter type: fixfile_out. The File FIX Output adapter writes stream data as FIX messages to a file.

Each stream hosts FIX messages of a certain type. The adapter writes messages to file in an adjoining manner, with no line feeds. It generates the following FIX fields:

- BeginString
- BodyLength
- MsgType
- CheckSum

Write the remaining fields in appropriate order in stream columns. Ensure the names of the stream columns correspond to the FIX protocol specification.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
FIX Version	Property ID: **fixVersion** Type: `choice` (Required) Version of the FIX protocol. Default value is 4.2.
FIX Message Type	Property ID: **fixMessageType** Type: `string` (Required) Type of messages hosted by the stream. No default value.
File	Property ID: **fileName** Type: `filename` (Required) Path to the input file containing FIX messages. No default value. Use a forward slash for both UNIX and Windows paths.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Date format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Timestamp format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.

Property Label	Description
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Known limitations:

- This adapter is not a full FIX engine.
- Supports only FIX versions 4.2, 4.3, 4.4, and 5.0.
- Does not support repeating groups and components.
- Does not support schema discovery.
- Supports only INSERT opcode.

See also

- *FIX Input Adapter* on page 162

Datatype Mapping for the File FIX Output Adapter

Event Stream Processor datatypes map to FIX datatypes.

Event Stream Processor Datatype	QuickFix Datatype
`integer`	`boolean`
`string`	`byte[]`
`string`	`char`
`string`	`string`
`date`	`date`
`float`	`float`
`integer`	`integer`
`date` or `timestamp`	`UTCDateOnly`

Event Stream Processor Datatype	QuickFix Datatype
`date` or `timestamp`	`UTCTimeOnly`
`date` or `timestamp`	`UTCTimeStamp`

File JSON Input and Output Adapter

The File JSON Input adapter takes JSON messages from JSON files, and publishes them to Event Stream Processor. The File JSON Output adapter takes data from Event Stream Processor, formats it into JSON format, and sends it to a JSON file.

File JSON Input Adapter Configuration

Configure the File JSON Input adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

Transporter Module: File Input Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
BufferMaxSize	Type: `integer` (Advanced) Specify the capacity of the buffer queue between the transporter module and the next module. The default value is 10240.
Parameters	(Required) Section containing the **FileInputTransporterParameters** element.
FileInputTransporterParameters	(Required) Section containng parameters for the File Input transporter.
Dir	Type: `string` (Required) Specify the absolute path to the data files which you want the adapter to read. For example, `<username>/<foldername>`. No default value. Use a forward slash for both UNIX and Windows paths.
File	Type: `string` (Required) Specify the file which you want the adapter to read or the regex pattern to filter the files on a given directory. See the **DynamicMode** parameter. No default value.

Parameter	Description
AccessMode	Type: `string` (Required) Specify an access mode. The adapter supports two modes: • **rowBased –** the adapter reads one text line at a time • **Streaming –** the adapter reads a pre-configured size of bytes into a buffer No default value.
DynamicMode	Type: `string` (Advanced) Specify a dynamic mode. The adapter supports three modes: • **Static –** the adapter reads the file specified in the **Dir** and **File** parameters • **dynamicFile –** the adapter reads the file specified in the **Dir** and **File** parameters and keeps polling the new appended content. The polling period is specified in the **PollingPeriod** parameter. • **dynamicPath –** the adapter polls all the new files under the **Dir** parameter. Also, the **File** parameter will act as a regex pattern and will filter out the necessary files. The default value is Static. If the DynamicMode has been set to dynamicPath and you leave the **File** parameter empty, the adapter reads all the files under the specified directory. An example regex pattern is ".*\.txt". This filters to only files that end with ".txt". In regex patterns, an escape character, "\", is necessary prior to the meta chars if you wish to include them in the pattern string.

Parameter	Description
PollingPeriod	Type: `integer` (Advanced) Define the period, in seconds, to poll the specified file or directory. Set this parameter only if the value of the **DynamicMode** parameter is set to dynamicFile or dynamicPath. A value of <=0 turns off polling. The default value is 0.
RemoveAfterProcess	Type: `boolean` (Optional) If this property is set to true, the file is removed from the directory after the adapter processes it. If the value of the **DynamicMode** parameter is set to dynamicFile or dynamicPath, the file is not removed after the adapter processes it. The default value is false.
ScanDepth	Type: `integer` (Optional) Specify the depth of the schema discovery. The adapter reads the number of rows specified by this parameter value when discovering the input data schema. The default value is three.

Formatter Module: Streaming JSON to JSON String Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **JsonStreamToJsonStringFormatterParameters** parameter.
JsonStreamToJsonStringFormatterParameters	(Required) Section containing parameters for the Streaming JSON to JSON String formatter.
CharsetName	Type: `string` (Optional) Specify the name of a supported charset. The default value is US-ASCII.

Formatter Module: JSON String to ESP Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.

Parameter	Description
Parameters	(Required) Section containing the **JsonString-ToEspFormatterParameters** parameter.
JsonStringToEspFormatterParameters	(Required) Section containing the JSON String to ESP formatter parameters.
ColumnMappings	(Required) Section containing the **ColsMapping** parameter.
ColsMapping	(Required) Section containing the **Column** parameter.
Column	Type: `string` (Required) Specify a value for the JSON data that you wish to map to ESP columns. This value is matched by a pattern path expression. For example, [<Column>JSONPath expression</Column>]. The first <Column/> is mapped to the first column of an ESP row, the second <Column/> is mapped to the second column of an ESP row, and so on. This parameter has two attributes: • **streamname –** Specify which stream to publish. • **rootpath –** Specify a rootpath for the JSON data. You can use one rootpath to publish more than one ESP row from one JSON data record.
DateFormat	Type: `string` (Advanced) The format string for parsing date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.
TimestampFormat	Type: `string` (Advanced) Format string for parsing timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

ESP Connector Module: ESP Publisher

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **EspPublisherParameters** parameter.
EspPublisherParameters	(Required) Section containing parameters for the ESP publisher.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.

Parameter	Description
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream to which the adapter publishes data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
MaxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.
UseTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the File JSON Input adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.

Parameter	Description
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.

Parameter	Description
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using key-store • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

File JSON Input Adapter Studio Properties

Adapter type: toolkit_file_json_input. Set these properties for the File JSON Input adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
JSON Column Mapping List	Property ID: **jsonColsMappingList** Type: `string` (Required) The JSON expression list is separated by commas. The first separated part is mapped to the first column of an ESP row, the second separated part is mapped to the second column of an ESP row, and so on.
JSON Root Path	Property ID: **jsonRootpath** Type: `string` (Required) Specify a root path for the JSON data.

Property Label	Description
Date Format	Propterty ID: **jsonDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **jsonTimestampFormat** Type: `string` (Advanced) Specify the format for parsing time-stamp values.
Directory	Property ID: **dir** Type: `directory` (Required) Specify the location of the data files.
File (in Directory)	Property ID: **file** Type: `string` (Required) Specify the file to read. Wildcards are allowed.
Dynamic Loading Mode	Property ID: **dynamicMode** Type: `string` (Optional) Set dynamic mode for reading files. Valid values: static, dynamicFile, dynamicPath.
Poll Period (seconds)	Property ID: **pollingPeriod** Type: `int` (Advanced) Specify the poll period when **dynamicMode** is dynamicFile or dynamicPath.
Remove Files After Processing	Property ID: **removeAfterProcess** Type: `boolean` (Optional) Removes files after they have been processed.

Property Label	Description
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File for the File JSON Input Adapter

Sample adapter configuration file for the File JSON Input adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>file_json_input</Name>
    <Description>An adapter which receives JSON message from json
file,transforms to ESP data format, and publishes to ESP stream.</
Description>
    <Log4jProperty>./log4j.properties</Log4jProperty>
    <Modules>
        <Module type="transporter">
            <InstanceName>FileInputTransporter</InstanceName>
            <Name>FileInputTransporter</Name>
            <Next>MyJsonStreamToJsonStringFormatter</Next>
            <BufferMaxSize>10240</BufferMaxSize>
            <Parameters>
                <FileInputTransporterParameters>
                    <Dir>./data</Dir>
                    <File>article_1.json</File>
                    <AccessMode>Streaming</AccessMode>
                    <RemoveAfterProcess>false</RemoveAfterProcess>
                    <ScanDepth>5</ScanDepth>
                </FileInputTransporterParameters>
            </Parameters>
        </Module>

        <Module type="formatter">
            <InstanceName>MyJsonStreamToJsonStringFormatter</
InstanceName>
            <Name>JsonStreamToJsonStringFormatter</Name>
            <Next>MyJsonInFormatter</Next>
            <Parameters />
        </Module>
```

```
        <Module type="formatter">
            <InstanceName>MyJsonInFormatter</InstanceName>
            <Name>JsonStringToEspFormatter</Name>
            <Next>MyInStream_Publisher</Next>
            <Parameters>
                <JsonStringToEspFormatterParameters>
                    <DateFormat>yyyy-MM-dd HH:mm:ss.SSS</DateFormat>
                    <TimestampFormat>yyyy/MM/dd HH:mm:ss</
TimestampFormat>
                    <ColumnMappings>
                        <ColsMapping streamname="EntityStream"
rootpath="entities">
                            <Column>display_text</Column>
                            <Column>domain_role</Column>
                            <Column>offset</Column>
                            <Column>length</Column>
                        </ColsMapping>
                    </ColumnMappings>
                </JsonStringToEspFormatterParameters>
            </Parameters>
        </Module>

        <Module type="espconnector">
            <InstanceName>MyInStream_Publisher</InstanceName>
            <Name>EspPublisher</Name>
            <Parameters>
                <EspPublisherParameters>
                    <ProjectName>EspProject1</ProjectName>
                    <StreamName>EntityStream</StreamName>
                    <MaxPubPoolSize>1</MaxPubPoolSize>
                    <UseTransactions>false</UseTransactions>
                    <SafeOps>true</SafeOps>
                    <SkipDels>true</SkipDels>
                </EspPublisherParameters>
            </Parameters>
        </Module>

    </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject1</Name>
            <Uri>esp://localhost:19011/sample_workspace/
file_json_input</Uri>
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
             <!-- <RSAKeyStore>/keystore/keystore.jks</RSAKeyStore>
<RSAKeyStorePassword>Sybase123</RSAKeyStorePassword> -->
                <!-- <KerberosKDC>KDC</KerberosKDC>
<KerberosRealm>REALM</KerberosRealm>
                 <KerberosService>service/instance</KerberosService>
<KerberosTicketCache>/tmp/krb5cc_user</KerberosTicketCache> -->
                <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
            </Security>
```

```
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

File JSON Output Adapter Configuration

Configure the File JSON Output adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

ESPConnector Module: ESP Subscriber

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, espconnector.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.

Parameter	Description
Parameters	(Required) Section containing the **EspSubscriberParameters** parameter.
EspSubscriberParameters	(Required) Section containing parameters for the ESP subscriber.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Specifies the unique project tag of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream from which the adapter subscribes to data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.

Formatter Module: ESP to JSON Stream Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.

Parameter	Description
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **EsptoJsonStreamFormatterParameters** parameter.
EsptoJsonStreamFormatterParameters	(Required) Section containing the ESP to JSON Stream formatter parameters.
ColsMapping	(Required) Section containing the **Column** parameter.
Column	Type: `complextype` (Required) Specify which columns of an ESP row you wish to map to JSON data. These values are matched by a pattern path expression. For example, [<Column>JSONPath expression</Column>]+. The first <Column/> is mapped to the first column of an ESP row, the second <Column/> is mapped to the second column of an ESP row, and so on.
DateFormat	Type: `string` (Advanced) The format string for date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.

Parameter	Description
TimestampFormat	Type: `string` (Advanced) Format string for timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

Transporter Module: File Output Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **FileOutputTransporterParameters** parameter.
FileOutputTransporterParameters	(Required) Section containing the parameters for the File Output transporter.
Dir	Type: `string` (Required) Specify the absolute path to the data files that you want the adapter to write to. For example, `<username>/<foldername>`. The default value is ".", meaning the current directory in which the adapter is running. Use a forward slash for both UNIX and Windows paths.

Parameter	Description
File	Type: `string` (Required) Specify the file to which you want the adapter to write.
AccessMode	Type: `string` (Required) Specify an access mode. The adapter supports two modes: • **rowBased –** the adapter writes one text line at a time into the file • **Streaming –** the adapter writes the raw data in ByteBuffer into the file No default value.
AppendMode	Type: `boolean` (Optional) If set to true, the adapter appends the data into the existing file. If set to false, the adapter overwrites existing content in the file. Default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the File JSON Output adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.

Parameter	Description
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.

Parameter	Description
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

File JSON Output Adapter Studio Properties

Adapter type: toolkit_file_json_output. Set these properties for the File JSON Output adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
JSON Column Mapping List	Property ID: **jsonColsMappingList** Type: `string` (Required) The JSON expression list is separated by commas. The first separated part is mapped to the first column of an ESP row, the second separated part is mapped to the second column of an ESP row, and so on.
Date Format	Propterty ID: **jsonDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **jsonTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.
Charset Name	Property ID: **charsetName** Type: `string` (Advanced) Specify the name of a supported charset.
Directory	Property ID: **dir** Type: `directory` (Required) Specify the location of the data files.
File (in Directory)	Property ID: **file** Type: `string` (Required) Specify the file to write to.

Property Label	Description
Append Mode	Property ID: **appendMode** Type: `boolean` (Advanced) Whether data should be appended to an existing file.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File for the File JSON Output Adapter

Sample adapter configuration file for the File JSON Output adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
  <Name>file_json_output</Name>
  <Description>The File JSON Output adapter takes data from Event
Stream Processor, formats it into JSON format, and sends it to a JSON
file.</Description>
  <Log4jProperty>./log4j.properties</Log4jProperty>
  <Modules>
    <Module type="espconnector">
      <InstanceName>MyOutStream_Subscriber</InstanceName>
      <Name>EspSubscriber</Name>
      <Next>MyJsonOutFormatter</Next>
      <Parameters>
        <EspSubscriberParameters>
          <ProjectName>EspProject1</ProjectName>
          <StreamName>EntityStream</StreamName>
        </EspSubscriberParameters>
      </Parameters>
    </Module>

    <Module type="formatter">
      <InstanceName>MyJsonOutFormatter</InstanceName>
      <Name>EspToJsonStringFormatter</Name>
      <Next>MyStringToStreamFormatter</Next>
      <Parameters>
```

```
        <EspToJsonStringFormatterParameters>
          <DateFormat>yyyy-MM-dd HH:mm:ss.SSS</DateFormat>
          <TimestampFormat>yyyy/MM/dd HH:mm:ss</TimestampFormat>
          <ColsMapping>
            <Column>published_at</Column>
            <Column>title</Column>
            <Column>lang</Column>
          </ColsMapping>
        </EspToJsonStringFormatterParameters>
      </Parameters>
    </Module>

    <Module type="formatter">
      <InstanceName>MyStringToStreamFormatter</InstanceName>
      <Name>StringToStreamFormatter</Name>
      <Next>FileOutputTransporter</Next>
      <Parameters>
        <StringToStreamFormatterParameters>
          <Delimiter>\n</Delimiter>
          <IncludeDelimiter>true</IncludeDelimiter>
          <AppendString>\n</AppendString>
          <AppendPosition>end</AppendPosition>
          <IgnoreSpace>true</IgnoreSpace>
         <CharsetName>US-ASCII</CharsetName>
       </StringToStreamFormatterParameters>
      </Parameters>
    </Module>

    <Module type="transporter">
    <InstanceName>FileOutputTransporter</InstanceName>
        <Name>FileOutputTransporter</Name>
        <Parameters>
            <FileOutputTransporterParameters>
                <Dir>./data</Dir>
                <File>output.json</File>
                <AccessMode>Streaming</AccessMode>
                <AppendMode>false</AppendMode>
            </FileOutputTransporterParameters>
        </Parameters>
    </Module>

  </Modules>

  <EspProjects>
    <EspProject>
      <Name>EspProject1</Name>
      <Uri>esp://localhost:19011/sample_workspace/file_json_output</
Uri>
      <Security>
        <User></User>
        <Password encrypted="false"></Password>
        <AuthType>user_password</AuthType>
        <!--
          <RSAKeyStore>/keystore/keystore.jks</RSAKeyStore>
          <RSAKeyStorePassword>Sybase123</RSAKeyStorePassword>
        -->
```

```
        <!--
         <KerberosKDC>KDC</KerberosKDC>
         <KerberosRealm>REALM</KerberosRealm>
          <KerberosService>service/instance</KerberosService>
         <KerberosTicketCache>/tmp/krb5cc_user</KerberosTicketCache>
        -->
        <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
      </Security>
    </EspProject>
  </EspProjects>
  <GlobalParameters></GlobalParameters>
</Adapter>
```

Adapter Controller Parameters

The Adapter Controller port listens for commands. The `controller.xml` file is located in `%ESP_HOME%/adapters/framework/config` directory. This file is shared among all the adapter within the `%ESP_HOME%/adapters/framework/instances` directory.

Parameter	Description
ControlPort	(Required) Section containing the **MinPort** and **MaxPort** parameters.
MinPort	Type: `int` (Required) The minimum port number that can act as a control port number. The adapter framework allocates this number automatically. A recommended value is 19082.
MaxPort	Type: `int` (Optional) The maximum port number that can act as a control port number. The adapter framework allocates this number automatically. The default value is 65535.
Statistics_Update_Interval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework publishes statistics data to the ESP project to which the adapter is connected. A recommended value is 5000.

Parameter	Description
MonitorInterval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework checks the status of internal resource allocation. The framework logs a message in the log file if a resource is about to be exhausted. A recommended value is 3000.

Logging

The File JSON Input and Output adapter use the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing the logging configuration is located in the `%ESP_HOME%\adapters\framework\config` directory.

Specify the location of the logging file you wish to use in the **Log4jProperty** parameter within the adapter configuration file. You can modify the logging levels within this file or the `$ESP_HOME/adapters/framework/config/log4j.properties`, which is used by default. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Starting the File JSON Adapter

To start the File JSON adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

a. Start the example cluster.

```
cd %ESP_HOME%\cluster\nodes\node1
  %ESP_HOME%\bin\esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
%ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
%ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
%ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

UNIX:

a. Start the example cluster.

```
cd $ESP_HOME/cluster/nodes/node1
  $ESP_HOME/bin/esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
$ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

2. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: For the File JSON Input adapter: `cd $ESP_HOME/adapters/framework/instances/file_json_input` For the File JSON Output adapter: `cd $ESP_HOME/adapters/framework/instances/file_json_output` `./start_adapter.sh <adapter configuration file name>`

Operating System	Step
Windows	Open a command window and enter: For the File JSON Input adapter: `cd %ESP_HOME%/adapters/framework/instances/file_json_input` For the File JSON Output adapter: `cd %ESP_HOME%/adapters/framework/instances/file_json_output` `start_adapter.bat <adapter configuration file name>`

Stopping the File JSON Adapter

To stop the File JSON adapter from the command line, execute the **stop** command.

Operating System	Step
UNIX	Open a terminal window and enter: For the File JSON Input adapter: `cd $ESP_HOME/adapters/framework/instances/file_json_input` For the File JSON Output adapter: `cd $ESP_HOME/adapters/framework/instances/file_json_output` `./stop_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the File JSON Input adapter: `cd %ESP_HOME%/adapters/framework/instances/file_json_input` For the File JSON Output adapter: `cd %ESP_HOME%/adapters/framework/instances/file_json_output` `stop_adapter.bat <adapter configuration file name>`

File XML Document Input and Output Adapter

The File XML Document Input adapter loads data from an XML document into a project in Event Stream Processor. The File XML Document Output adapter outputs data from a project in Event Stream Processor into an XML document.

File XML Document Input Adapter Configuration

Configure the File XML Document Input adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

Transporter Module: File Input Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.

Parameter	Description
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
BufferMaxSize	Type: `integer` (Advanced) Specify the capacity of the buffer queue between the transporter module and the next module. The default value is 10240.
Parameters	(Required) Section containing the **FileInputTransporterParameters** element.
FileInputTransporterParameters	(Required) Section containng parameters for the File Input transporter.
Dir	Type: `string` (Required) Specify the absolute path to the data files which you want the adapter to read. For example, `<username>/<foldername>`. No default value. Use a forward slash for both UNIX and Windows paths.
File	Type: `string` (Required) Specify the file which you want the adapter to read or the regex pattern to filter the files on a given directory. See the **DynamicMode** parameter. No default value.
AccessMode	Type: `string` (Required) Specify an access mode. The adapter supports two modes: • **rowBased –** the adapter reads one text line at a time • **Streaming –** the adapter reads a pre-configured size of bytes into a buffer No default value.

Parameter	Description
DynamicMode	Type: `string` (Advanced) Specify a dynamic mode. The adapter supports three modes: • **Static –** the adapter reads the file specified in the **Dir** and **File** parameters • **dynamicFile –** the adapter reads the file specified in the **Dir** and **File** parameters and keeps polling the new appended content. The polling period is specified in the **PollingPeriod** parameter. • **dynamicPath –** the adapter polls all the new files under the **Dir** parameter. Also, the **File** parameter will act as a regex pattern and will filter out the necessary files. The default value is Static. If the DynamicMode has been set to dynamicPath and you leave the **File** parameter empty, the adapter reads all the files under the specified directory. An example regex pattern is ".*\.txt". This filters to only files that end with ".txt". In regex patterns, an escape character, "\", is necessary prior to the meta chars if you wish to include them in the pattern string.
PollingPeriod	Type: `integer` (Advanced) Define the period, in seconds, to poll the specified file or directory. Set this parameter only if the value of the **DynamicMode** parameter is set to dynamicFile or dynamicPath. A value of <=0 turns off polling. The default value is 0.

Parameter	Description
RemoveAfterProcess	Type: `boolean` (Optional) If this property is set to true, the file is removed from the directory after the adapter processes it. If the value of the **DynamicMode** parameter is set to dynamicFile or dynamicPath, the file is not removed after the adapter processes it. The default value is false.
ScanDepth	Type: `integer` (Optional) Specify the depth of the schema discovery. The adapter reads the number of rows specified by this parameter value when discovering the input data schema. The default value is three.

Formatter Module: XMLDoc Stream to ESP Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.

Parameter	Description
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **XmlDocStreamToEspFormatterParameters** parameter.
XmlDocStreamToEspFormatterParameters	(Required) Section containing the XMLDoc Stream to ESP formatter parameters.
XmlElemMappingRowPattern	Type: `string` (Required) Specify a pattern to determine which XML elements in the XML doc are processed by the formatter. The matched elements are mapped to ESP rows, whose attributes and child elements are mapped as columns of an ESP row. The adapter ignores any XML elements that do not match this pattern. This pattern is a subset of the XPath expressions. The [/]?NCName[/NCName]* path expression is the only supported expression, where NCName (Non-Colonized Name) is the local name element without a prefix or namespace. If the elements in the path expression include a namespace URI (prefix), they belong to the same namespace. Provide the namespace in the **XmlElemNamespace** parameter. Here are some examples of valid path expressions: • /RootElement • ParentElement • ParentElement/ChildElement • /RootElement/ParentElement
XmlElemNamespace	Type: `string` (Required) Specify the namespace URI for elements that appear in the pattern path expression.
ColsMapping	(Required) Section containing the **Column** parameter.

Parameter	Description
Column	Type: `string` (Required) Specify which attributes or child elements of the XML elements, which are matched by pattern path expression, to map to columns of the ESP row. For example, [<Column>XPath expression</Column>]+. The XPath expression is any valid XPath expression specified by an XPath specification. The XPath expression can only begin from the last XML element that appears in the path pattern expression or its decedent elements. The first <Column/> is mapped to the first column of the ESP row, the second <Column/> is mapped to the second column of the ESP row, and so on.
DateFormat	Type: `string` (Optional) The format string for parsing date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.
TimestampFormat	Type: `string` (Optional) Format string for parsing timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

ESP Connector Module: ESP Publisher

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
BufferMaxSize	Type: `integer` (Advanced) Specify the capacity of the buffer queue between the transporter module and the next module. The default value is 10240.
Parameters	(Required) Section containing the **EspPublisherParameters** parameter.
EspPublisherParameters	(Required) Section containing parameters for the ESP publisher.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.

Parameter	Description
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream to which the adapter publishes data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
MaxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.
UseTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the File XML Document Input adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.

Parameter	Description
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.

Parameter	Description
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using key-store • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

File XML Document Input Adapter Studio Properties

Adapter type: toolkit_file_xmldoc_input. Set these properties for the File XML Document Input adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Directory	Property ID: **dir** Type: `directory` (Required) Specify the location of the data files.
File (in Directory)	Property ID: **file** Type: `string` (Required) Specify the file to read. Wildcards are allowed.
Row Expression	Property ID: **xmlElemMappingRowPattern** Type: `string` (Required) The XPath expression for mapping XML elements to ESP rows.

Property Label	Description
Namespace URI	Property ID: **xmlElemNamespaceURI** Type: `string` (Optional) The namespace URI for XML elements included in the pattern expression.
Column Expression List	Property ID: **espColumnPattern** Type: `string` (Required) Comma-separated list of XPath expressions for mapping XML elements to ESP columns. If the XML element used in an expression belongs to a namespace, specify the namespace in the expression.
Dynamic Loading Mode	Property ID: **dynamicMode** Type: `string` (Optional) Set dynamic mode for reading files. Valid values: static, dynamicFile, dynamicPath.
Poll Period (seconds)	Property ID: **pollingPeriod** Type: `int` (Advanced) Specify the poll period when **dynamicMode** is dynamicFile or dynamicPath.
Dynamic Loading Mode	Property ID: **dynamicMode** Type: `string` (Optional) Set dynamic mode for reading files. Valid values: static, dynamicFile, dynamicPath.
Poll Period (seconds)	Property ID: **pollingPeriod** Type: `int` (Advanced) Specify the poll period when **dynamicMode** is dynamicFile or dynamicPath.
Remove Files After Processing	Property ID: **removeAfterProcess** Type: `boolean` (Optional) Removes files after they have been processed.

Property Label	Description
Date Format	Propterty ID: **xmldocDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **xmldocTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File for the File XML Document Input Adapter

Two sample adapter configuration files for the File XML Document Input adapter.

The example below shows how the adapter parses datetime and timestamp data from an ESP project.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
  <Name>file_xmldoc_input_2</Name>
  <Description>An adapter which gets data from local xml file,
transforms to ESP data format, and publishes to ESP stream.</
Description>
  <Log4jProperty>./log4j.properties</Log4jProperty>
  <Modules>
    <Module type="transporter">
      <InstanceName>FileInputTransporter</InstanceName>
      <Name>FileInputTransporter</Name>
      <Next>MyXMLDocESPFORMATTER</Next>
      <BufferMaxSize>10240</BufferMaxSize>
      <Parameters>
          <FileInputTransporterParameters>
```

```
            <!-- Set the "Dir" to absolute path pointing to $ESP_HOME
\adapters\framework\exampleAdapters\XMLDocumentAdapters
\XMLDocumentInputAdapter\datetimeExample\data -->
              <Dir></Dir>
              <File>.*\.xml</File>
              <AccessMode>streaming</AccessMode>
              <DynamicMode>dynamicPath</DynamicMode>
              <PollingPeriod>3</PollingPeriod>
              <RemoveAfterProcess>false</RemoveAfterProcess>
          </FileInputTransporterParameters>
      </Parameters>
    </Module>

    <Module type="formatter">
      <InstanceName>MyXMLDocESPFORMATTER</InstanceName>
      <Name>XmlDocStreamToEspFormatter</Name>
      <Next>MyInStream_Publisher</Next>
      <Parallel>true</Parallel>
      <Parameters>
           <XmlDocStreamToEspFormatterParameters>
                <XmlElemMappingRowPattern>/orders/purchase-order</
XmlElemMappingRowPattern>
                <ColsMapping>
                     <Column>/purchase-order/@id</Column>
                     <Column>//name</Column>
                     <Column>date</Column>
                     <Column>datetime</Column>
                     <Column>totalPrice</Column>
                </ColsMapping>
                <DateFormat>yyyy-MM-dd'T'XXX</DateFormat>
                <TimestampFormat>yyyy-MM-dd'T'HH:mm:ss.SSSXXX</
TimestampFormat>
           </XmlDocStreamToEspFormatterParameters>
      </Parameters>
    </Module>

    <Module type="espconnector">
      <InstanceName>MyInStream_Publisher</InstanceName>
      <Name>EspPublisher</Name>
      <Parameters>
        <EspPublisherParameters>
          <ProjectName>EspProject1</ProjectName>
          <StreamName>MyInStream</StreamName>
          <MaxPubPoolSize>1</MaxPubPoolSize>
          <UseTransactions>false</UseTransactions>
        </EspPublisherParameters>
      </Parameters>
      <BufferMaxSize>10240</BufferMaxSize>
    </Module>

  </Modules>

  <EspProjects>
    <EspProject>
      <Name>EspProject1</Name>
      <Uri>esp://localhost:19011/sample_workspace/
```

```
file_xmldoc_input_2</Uri>
      <Security>
        <User></User>
        <Password encrypted="false"></Password>
        <AuthType>user_password</AuthType>
      </Security>
    </EspProject>
  </EspProjects>
  <GlobalParameters></GlobalParameters>
</Adapter>
```

This example shows how the adapter obtains data from an XML document with name space declaration.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
  <Name>file_xmldoc_input_1</Name>
  <Description>An adapter which gets data from local xml file,
transporms to ESP data format, and publishes to ESP stream.</
Description>
  <Log4jProperty>./log4j.properties</Log4jProperty>
  <Modules>
    <Module type="transporter">
      <InstanceName>FileInputTransporter</InstanceName>
      <Name>FileInputTransporter</Name>
      <Next>MyXMLDocESPFORMATTER</Next>
      <BufferMaxSize>10240</BufferMaxSize>
      <Parameters>
          <FileInputTransporterParameters>
            <!-- Set the "Dir" to absolute path pointing to $ESP_HOME
\adapters\framework\exampleAdapters\XMLDocumentAdapters
\XMLDocumentInputAdapter\namespaceExample\data -->
              <Dir></Dir>
              <File>.*\.xml</File>
              <AccessMode>streaming</AccessMode>
              <DynamicMode>dynamicPath</DynamicMode>
              <PollingPeriod>3</PollingPeriod>
              <RemoveAfterProcess>false</RemoveAfterProcess>
          </FileInputTransporterParameters>
      </Parameters>
    </Module>

    <Module type="formatter">
      <InstanceName>MyXMLDocESPFORMATTER</InstanceName>
      <Name>XmlDocStreamToEspFormatter</Name>
      <Next>MyInStream_Publisher</Next>
      <Parallel>true</Parallel>
      <Parameters>
           <XmlDocStreamToEspFormatterParameters>
                <XmlElemMappingRowPattern>books/book</
XmlElemMappingRowPattern>
                <XmlElemNamespaceURI>http://sap.com/xmlns/books</
XmlElemNamespaceURI>
                <ColsMapping>
                   <Column>/*[namespace-uri()='http://sap.com/xmlns/
books' and local-name()='book']/@*[namespace-uri()='http://sap.com/
```

```
xmlns/books' and local-name()='id']</Column>
                  <Column>/*[namespace-uri()='http://sap.com/xmlns/
books' and local-name()='book']/*[namespace-uri()='http://sap.com/
xmlns/books' and local-name()='name']</Column>
                  <Column>/*[namespace-uri()='http://sap.com/xmlns/
books' and local-name()='book']/*[namespace-uri()='http://sap.com/
xmlns/books' and local-name()='price']</Column>
               </ColsMapping>
            </XmlDocStreamToEspFormatterParameters>
      </Parameters>
    </Module>

    <Module type="espconnector">
      <InstanceName>MyInStream_Publisher</InstanceName>
      <Name>EspPublisher</Name>
      <Parameters>
        <EspPublisherParameters>
          <ProjectName>EspProject1</ProjectName>
          <StreamName>MyInStream</StreamName>
          <MaxPubPoolSize>1</MaxPubPoolSize>
          <UseTransactions>false</UseTransactions>
        </EspPublisherParameters>
      </Parameters>
      <BufferMaxSize>10240</BufferMaxSize>
    </Module>

  </Modules>

  <EspProjects>
    <EspProject>
      <Name>EspProject1</Name>
      <Uri>esp://localhost:19011/sample_workspace/
file_xmldoc_input_1</Uri>
      <Security>
        <User></User>
        <Password encrypted="false"></Password>
        <AuthType>user_password</AuthType>
      </Security>
    </EspProject>
  </EspProjects>
  <GlobalParameters></GlobalParameters>
</Adapter>
```

File XML Document Output Adapter Configuration

Configure the File XML Document Output adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

ESPConnector Module: ESP Subscriber

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, espconnector.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **EspSubscriberParameters** parameter.
EspSubscriberParameters	(Required) Section containing parameters for the ESP subscriber.

Parameter	Description
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Specifies the unique project tag of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream from which the adapter subscribes to data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.

Formatter Module: ESP to XMLDoc String Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **EspToXmlDocStringFormatterParameters** parameter.
EspToXmlDocStringFormatterParameters	(Required) Section containing parameters for the ESP to XMLDoc String formatter parameters.
XMLSchemaFilePath	Type: `string` (Required) The path to the XML schema file which the XML output document builds against. No default value.
GlobalElementLocalName	Type: `string` (Required) Specify a global element that you wish to use as the root element in the generated XML document. No default value.
ColsMapping	(Required) Section containing the **Column** parameter.

Parameter	Description
Column	Type: `string` (Required) Specify which attributes or child elements, that are generated by the global element, to match by a pattern path expression and map to columns of an ESP row. For example, [<Column>XPath expression</Column>]+. The XPath expression is any valid XPath expression specified by an XPath specification. The first <Column/> is mapped to the first column of an ESP row, the second <Column/> is mapped to the second column of an ESP row, and so on.

Transporter Module: File Output Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **FileOutputTransporterParameters** parameter.
FileOutputTransporterParameters	(Required) Section containing the parameters for the File Output transporter.

Parameter	Description
Dir	Type: `string` (Required) Specify the absolute path to the data files that you want the adapter to write to. For example, `<username>/<foldername>`. The default value is ".", meaning the current directory in which the adapter is running. Use a forward slash for both UNIX and Windows paths.
File	Type: `string` (Required) Specify the file to which you want the adapter to write.
AccessMode	Type: `string` (Required) Specify an access mode. The adapter supports two modes: • **rowBased –** the adapter writes one text line at a time into the file • **Streaming –** the adapter writes the raw data in ByteBuffer into the file No default value.
AppendMode	Type: `boolean` (Optional) If set to true, the adapter appends the data into the existing file. If set to false, the adapter overwrites existing content in the file. Default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the XMLDocument Output adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.

Parameter	Description
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.

Parameter	Description
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using key-store • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

File XML Document Output Adapter Studio Properties

Adapter type: toolkit_file_xmldoc_output. Set these properties for the File XML Document Output adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Directory	Property ID: **dir** Type: `directory` (Required) Specify the location of the data files.
File (in Directory)	Property ID: **file** Type: `string` (Required) Specify the file to write to.
Append Mode	Property ID: **appendMode** Type: `boolean` (Advanced) Whether data should be appended to an existing file.

Property Label	Description
XML Schema Document	Property ID: **xmlSchemaFilePath** Type: `filename` (Required) The XML schema document, on which the generated XML document is based.
Local Name of XSD Global Element	Property ID: **globalElementLocalName** Type: `string` (Required) The local name of a global element in the XML schema document.
Column Expression List	Property ID: **espColumnPattern** Type: `string` (Required) Comma-separated list of XPath expressions for mapping ESP columns to XML elements. If the XML element used in an expression belongs to a namespace, specify the namespace in the expression.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File for the File XML Document Output Adapter

Sample adapter configuration file for the File XML Document Output adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>file_xmldoc_output</Name>
    <Description>An adapter which transforms ESP data to String and
save to xml file on local hard disk.</Description>
    <Log4jProperty>./log4j.properties</Log4jProperty>
    <Modules>
```

```
        <Module type="espconnector">
            <InstanceName>MyOutStream_Subscriber</InstanceName>
            <Name>EspSubscriber</Name>
            <Next>XmlDocOutputFormatter</Next>
            <Parameters>
                <EspSubscriberParameters>
                    <ProjectName>XMLDOCOUTROOTELEM</ProjectName>
                    <StreamName>BaseOutput</StreamName>
                </EspSubscriberParameters>
            </Parameters>
        </Module>

        <Module type="formatter">
            <InstanceName>XmlDocOutputFormatter</InstanceName>
            <Name>EspToXmlDocStringFormatter</Name>
            <Next>FileOutputTransporter</Next>
            <Parallel>true</Parallel>
            <Parameters>
                <EspToXmlDocStringFormatterParameters>
                <XMLSchemaFilePath>outschema.xsd</XMLSchemaFilePath>
                <GlobalElementLocalName>purchase-order</
GlobalElementLocalName>
                <ColsMapping>
                    <Column>/*[namespace-uri()='http://openuri.org/
easypo' and local-name()='purchase-order']/@*[namespace-
uri()='http://openuri.org/easypo' and local-name()='id']</Column>
                    <Column>/*[namespace-uri()='http://openuri.org/
easypo' and local-name()='purchase-order']/*[namespace-
uri()='http://openuri.org/easypo' and local-name()='name']</Column>
                    <Column>/*[namespace-uri()='http://openuri.org/
easypo' and local-name()='purchase-order']/*[namespace-
uri()='http://openuri.org/easypo' and local-name()='price']</Column>
                </ColsMapping>
                </EspToXmlDocStringFormatterParameters>
            </Parameters>
        </Module>

        <Module type="transporter">
            <InstanceName>FileOutputTransporter</InstanceName>
            <Name>FileOutputTransporter</Name>
            <Parameters>
                <FileOutputTransporterParameters>
                    <Dir>./data</Dir>
                    <File>output.xml</File>
                    <AccessMode>rowBased</AccessMode>
                    <AppendMode>false</AppendMode>
                </FileOutputTransporterParameters>
            </Parameters>
        </Module>
    </Modules>

    <EspProjects>
        <EspProject>
            <Name>XMLDOCOUTROOTELEM</Name>
            <Uri>esp://localhost:19011/sample_workspace/
file_xmldoc_output</Uri>
```

```
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Adapter Controller Parameters

The Adapter Controller port listens for commands. The `controller.xml` file is located in `%ESP_HOME%/adapters/framework/config` directory. This file is shared among all the adapter within the `%ESP_HOME%/adapters/framework/instances` directory.

Parameter	Description
ControlPort	(Required) Section containing the **MinPort** and **MaxPort** parameters.
MinPort	Type: `int` (Required) The minimum port number that can act as a control port number. The adapter framework allocates this number automatically. A recommended value is 19082.
MaxPort	Type: `int` (Optional) The maximum port number that can act as a control port number. The adapter framework allocates this number automatically. The default value is 65535.
Statistics_Update_Interval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework publishes statistics data to the ESP project to which the adapter is connected. A recommended value is 5000.

Parameter	Description
MonitorInterval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework checks the status of internal resource allocation. The framework logs a message in the log file if a resource is about to be exhausted. A recommended value is 3000.

Logging

The File XML Document Input and Output adapter use the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing the logging configuration is located in the `%ESP_HOME%\adapters\framework\config` directory.

Specify the location of the logging file you wish to use in the **Log4jProperty** parameter within the adapter configuration file. You can modify the logging levels within this file or the `$ESP_HOME/adapters/framework/config/log4j.properties`, which is used by default. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Starting the File XML Document Adapter

To start the File XML Document adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

a. Start the example cluster.

```
cd %ESP_HOME%\cluster\nodes\node1
  %ESP_HOME%\bin\esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
%ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
%ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
%ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

UNIX:

a. Start the example cluster.

```
cd $ESP_HOME/cluster/nodes/node1
  $ESP_HOME/bin/esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
$ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

2. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: For the File XML Document Input adapter: `cd $ESP_HOME/adapters/framework/instances/file_xmldoc_input` For the File XML Document Output adapter: `cd $ESP_HOME/adapters/framework/instances/file_xmldoc_output` `./start_adapter.sh <adapter configuration file name>`

Operating System	Step
Windows	Open a command window and enter: For the File XML Document Input adapter: `cd %ESP_HOME%/adapters/framework/instances/file_xmldoc_input` For the File XML Document Output adapter: `cd %ESP_HOME%/adapters/framework/instances/file_xmldoc_output` `start_adapter.bat <adapter configuration file name>`

Stopping the File XML Document Adapter

To stop the File XML Document adapter from the command line, execute the **stop** command.

Operating System	Step
UNIX	Open a terminal window and enter: For the File XML Document Input adapter: `cd $ESP_HOME/adapters/framework/instances/file_xmldoc_input` For the File XML Document Output adapter: `cd $ESP_HOME/adapters/framework/instances/file_xmldoc_output` `./stop_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the File XML Document Input adapter: `cd %ESP_HOME%/adapters/framework/instances/file_xmldoc_input` For the File XML Document Output adapter: `cd %ESP_HOME%/adapters/framework/instances/file_xmldoc_output` `stop_adapter.bat <adapter configuration file name>`

File XML Record Input and Output Adapter

The File XML Record Input adapter reads XML list text files and inputs this data into Event Stream Processor. The File XML Record Output adapter reads rows from Event Stream Processor ands writes this data into XML list files.

File XML Record Input Adapter Configuration

Configure the File XML Record Input adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

Transporter Module: File Input Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.

Parameter	Description
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
BufferMaxSize	Type: `integer` (Advanced) Specify the capacity of the buffer queue between the transporter module and the next module. The default value is 10240.
Parameters	(Required) Section containing the **FileInputTransporterParameters** element.
FileInputTransporterParameters	(Required) Section containing parameters for the File Input transporter.
Dir	Type: `string` (Required) Specify the absolute path to the data files which you want the adapter to read. For example, `<username>/<foldername>`. No default value. Use a forward slash for both UNIX and Windows paths.
File	Type: `string` (Required) Specify the file which you want the adapter to read or the regex pattern to filter the files on a given directory. See the **DynamicMode** parameter. No default value.
AccessMode	Type: `string` (Required) Specify an access mode. The adapter supports two modes: • **rowBased –** the adapter reads one text line at a time • **Streaming –** the adapter reads a pre-configured size of bytes into a buffer No default value.

Parameter	Description
DynamicMode	Type: `string` (Advanced) Specify a dynamic mode. The adapter supports three modes: • **Static –** the adapter reads the file specified in the **Dir** and **File** parameters • **dynamicFile –** the adapter reads the file specified in the **Dir** and **File** parameters and keeps polling the new appended content. The polling period is specified in the **PollingPeriod** parameter. • **dynamicPath –** the adapter polls all the new files under the **Dir** parameter. Also, the **File** parameter will act as a regex pattern and will filter out the necessary files. The default value is Static. If the DynamicMode has been set to dynamicPath and you leave the **File** parameter empty, the adapter reads all the files under the specified directory. An example regex pattern is ".*\.txt". This filters to only files that end with ".txt". In regex patterns, an escape character, "\", is necessary prior to the meta chars if you wish to include them in the pattern string.
PollingPeriod	Type: `integer` (Advanced) Define the period, in seconds, to poll the specified file or directory. Set this parameter only if the value of the **DynamicMode** parameter is set to dynamicFile or dynamicPath. A value of <=0 turns off polling. The default value is 0.

Parameter	Description
RemoveAfterProcess	Type: `boolean` (Optional) If this property is set to true, the file is removed from the directory after the adapter processes it. If the value of the **DynamicMode** parameter is set to dynamicFile or dynamicPath, the file is not removed after the adapter processes it. The default value is false.
ScanDepth	Type: `integer` (Optional) Specify the depth of the schema discovery. The adapter reads the number of rows specified by this parameter value when discovering the input data schema. The default value is three.

Formatter Module: XML String to ESP Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.

Parameter	Description
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **XmlString-ToEspFormatterParameters** parameter.
XmlStringToEspFormatterParameters	(Required) Section containing the XML String to ESP formatter parameters.
DateFormat	Type: `string` (Optional) The format string for parsing date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.
TimestampFormat	Type: `string` (Optional) Format string for parsing timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

ESP Connector Module: ESP Publisher

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **EspPublisherParameters** parameter.

Parameter	Description
EspPublisherParameters	(Required) Section containing parameters for the ESP publisher.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream to which the adapter publishes data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
MaxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.

Parameter	Description
MaxPubPoolTime	Type: `positive integer` (Optional) Specifies the maximum period of time, in milliseconds, for which records are pooled before being published. If not set, pooling time is unlimited and the pooling strategy is governed by **maxPubPoolSize**. No default value.
UseTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.
SafeOps	Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT, and converts DELETE to SAFEDELETE. The default value is false.
SkipDels	Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. The default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the File XML Record Input adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.

Parameter	Description
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.

Parameter	Description
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

See also

- *Adapter Support for Schema Discovery* on page 1005

File XML Record Input Adapter Studio Properties

Adapter type: toolkit_file_xmllist_input. Set these properties for the File XML Record Input adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Directory	Property ID: **dir** Type: `directory` (Required) Specify the location of the data files.
File (in Directory)	Property ID: **file** Type: `string` (Required) Specify the file to read. Wildcards are allowed.
Date Format	Propterty ID: **xmllistDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **xmllistTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.
Dynamic Loading Mode	Property ID: **dynamicMode** Type: `string` (Optional) Set dynamic mode for reading files. Valid values: static, dynamicFile, dynamicPath.
Poll Period (seconds)	Property ID: **pollingPeriod** Type: `int` (Advanced) Specify the poll period when **dynamicMode** is dynamicFile or dynamicPath.

Property Label	Description
Remove Files After Processing	Property ID: **removeAfterProcess** Type: `boolean` (Optional) Removes files after they have been processed.
Scan Depth	Property ID: **scanDepth** Type: `int` (Advanced) Specify the depth of the schema discovery. The adapter reads the number of rows specified by this parameter value when discovering the input data scheme.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: File XML Record Input Adapter

Sample adapter configuration file for the File XML Record Input adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
  <Name>file_xmllist_input</Name>
  <Description>An adapter which gets xml list data from files on
local hard disk, transforms to ESP data format, and publishes to ESP
stream.</Description>
  <Log4jProperty>./log4j.properties</Log4jProperty>
  <Modules>
    <Module type="transporter">
      <InstanceName>FileInputTransporter</InstanceName>
      <Name>FileInputTransporter</Name>
      <Next>XmlListInputFormatter</Next>
      <BufferMaxSize>10240</BufferMaxSize>
      <Parameters>
          <FileInputTransporterParameters>
```

```
              <Dir>./data</Dir>
              <File>input.xml</File>
              <AccessMode>rowBased</AccessMode>
              <RemoveAfterProcess>false</RemoveAfterProcess>
              <ScanDepth>5</ScanDepth>
          </FileInputTransporterParameters>
      </Parameters>
    </Module>

    <Module type="formatter">
      <InstanceName>XmlListInputFormatter</InstanceName>
      <Name>XmlStringToEspFormatter</Name>
      <Next>MyInStream_Publisher</Next>
      <Parallel>true</Parallel>
      <Parameters>
      </Parameters>
    </Module>

    <Module type="espconnector">
      <InstanceName>MyInStream_Publisher</InstanceName>
      <Name>EspPublisher</Name>
      <Parameters>
        <EspPublisherParameters>
          <ProjectName>EspProject1</ProjectName>
          <StreamName>BaseInput</StreamName>
          <MaxPubPoolSize>1</MaxPubPoolSize>
          <UseTransactions>false</UseTransactions>
          <SafeOps>true</SafeOps>
          <SkipDels>true</SkipDels>
        </EspPublisherParameters>
      </Parameters>
      <BufferMaxSize>10240</BufferMaxSize>
    </Module>

  </Modules>

  <EspProjects>
    <EspProject>
      <Name>EspProject1</Name>
      <Uri>esp://localhost:19011/sample_workspace/
file_xmllist_input</Uri>
      <Security>
        <User></User>
        <Password encrypted="false"></Password>
        <AuthType>user_password</AuthType>
      </Security>
    </EspProject>
  </EspProjects>
  <GlobalParameters></GlobalParameters>
</Adapter>
```

Here is a sample input file which contains a stream name and opcode value:

```
<BaseInput ESP_OPS="i" stringCol="aaa" int32Col="11" int64Col="111"
doubleCol="1.1" dateCol="2008-03-13T08:19:30" moneyCol="111.1111"
timestampCol="2008-03-13T08:19:30.123" booleanCol="false"
```

```
binaryCol="FF00FE05FF" bigdatetimeCol="2008-03-13T08:19:30.123456"
intervalCol="64000" money1Col="922.0"
money15Col="337.000000000000000" />
<BaseInput ESP_OPS="u" stringCol="new" int32Col="22" int64Col="222"
doubleCol="2.2" dateCol="2008-03-13T08:19:30" moneyCol="222.2222"
timestampCol="2008-03-13T08:19:30.123" booleanCol="true"
binaryCol="FF00FE05FF" bigdatetimeCol="2008-03-13T08:19:30.123456"
intervalCol="64000" money1Col="922.0"
money15Col="337.000000000000000" />
<BaseInput ESP_OPS="i" stringCol="bbb" int32Col="33" int64Col="333"
doubleCol="3.3" dateCol="2008-03-13T08:19:30" moneyCol="111.1111"
timestampCol="2008-03-13T08:19:30.123" booleanCol="true"
binaryCol="FF00FE05FF" bigdatetimeCol="2008-03-13T08:19:30.123456"
intervalCol="64000" money1Col="922.0"
money15Col="337.000000000000000" />
<BaseInput ESP_OPS="d" stringCol="bbb" int32Col="33" int64Col="333"
doubleCol="3.3" dateCol="2008-03-13T08:19:30" moneyCol="111.1111"
timestampCol="2008-03-13T08:19:30.123" booleanCol="true"
binaryCol="FF00FE05FF" bigdatetimeCol="2008-03-13T08:19:30.123456"
intervalCol="64000" money1Col="922.0"
money15Col="337.000000000000000" />
```

The accepted opcodes are:

- i or I: INSERT
- d or D: DELETE
- u or U: UPDATE
- p or P: UPSERT
- s or S: SAFEDELETE

Here is a sample input file which does not specify an opcode value. By default, the opcode UPSERT is used.

```
<BaseInput stringCol="aaa" int32Col="11" int64Col="111"
doubleCol="1.1"
    dateCol="2008-03-13T08:19:30" moneyCol="111.1111"
timestampCol="2008-03-13T08:19:30.123"
    booleanCol="false" binaryCol="FF00FE05FF"
bigdatetimeCol="2008-03-13T08:19:30.123456"
    intervalCol="64000"  money1Col="922.0"
money15Col="337.000000000000000" /><BaseInput
    stringCol="new" int32Col="22" int64Col="222" doubleCol="2.2"
    dateCol="2008-03-13T08:19:30" moneyCol="222.2222"
timestampCol="2008-03-13T08:19:30.123"
    booleanCol="true" binaryCol="FF00FE05FF"
bigdatetimeCol="2008-03-13T08:19:30.123456"
    intervalCol="64000"  money1Col="922.0"
money15Col="337.000000000000000" /><BaseInput
    stringCol="bbb" int32Col="33" int64Col="333" doubleCol="3.3"
    dateCol="2008-03-13T08:19:30" moneyCol="111.1111"
timestampCol="2008-03-13T08:19:30.123"
    booleanCol="true" binaryCol="FF00FE05FF"
bigdatetimeCol="2008-03-13T08:19:30.123456"
    intervalCol="64000"  money1Col="922.0"
money15Col="337.000000000000000" />
```

File XML Record Output Adapter Configuration

Configure the File XML Record Output adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

ESPConnector Module: ESP Subscriber

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, espconnector.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	Section containing the **EspSubscriberParameters** parameter.
EspSubscriberParameters	(Required) Section containing parameters for the ESP subscriber.

Parameter	Description
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Specifies the unique project tag of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream from which the adapter subscribes to data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.

Formatter Module: ESP to XML String Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **EspToXmlStringFormatterParameters** parameter.
EspToXmlStringFormatterParameters	(Required) Section containing the ESP to XML String formatter parameters.
DateFormat	Type: `string` (Optional) The format string for date values from an ESP project. For example, `yyyy-MM-dd'T'HH:mm:ss.`
TimestampFormat	Type: `string` (Optional) Format string for timestamp values from an ESP project. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

Transporter Module: File Output Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.

Parameter	Description
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **FileOutputTransporterParameters** parameter.
FileOutputTransporterParameters	(Required) Section containing the File Output transporter parameters.
Dir	Type: `string` (Required) Specify the absolute path to the data files that you want the adapter to write to. For example, `<username>/<foldername>`. The default value is ".", meaning the current directory in which the adapter is running. Use a forward slash for both UNIX and Windows paths.
File	Type: `string` (Required) Specify the file to which you want the adapter to write.
AccessMode	Type: `string` (Required) Specify an access mode. The adapter supports two modes: • **rowBased –** the adapter writes one text line at a time into the file • **Streaming –** the adapter writes the raw data in ByteBuffer into the file No default value.

Parameter	Description
AppendMode	Type: `boolean` (Optional) If set to true, the adapter appends the data into the existing file. If set to false, the adapter overwrites existing content in the file. Default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the File XML Record Output adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.

Parameter	Description
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

File XML Record Output Adapter Studio Properties

Adapter type: toolkit_file_xmllist_output. Set these properties for the File XML Record Output adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Directory	Property ID: **dir** Type: `directory` (Required) Specify the location of the data files.
File (in Directory)	Property ID: **file** Type: `string` (Required) Specify the file to write to.

Property Label	Description
Append Mode	Property ID: **appendMode** Type: `boolean` (Advanced) Whether data should be appended to an existing file.
Date Format	Propterty ID: **xmllistDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **xmllistTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: File XML Record Output Adapter

Sample adapter configuration file for the File XML Record Output adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>file_xmllist_output</Name>
    <Description>An adapter which transforms ESP data to xml list
format, and save to file on local hard disk.</Description>
    <Log4jProperty>./log4j.properties</Log4jProperty>
    <Modules>
        <Module type="espconnector">
            <InstanceName>MyOutStream_Subscriber</InstanceName>
            <Name>EspSubscriber</Name>
            <Next>XmlListOutputFormatter</Next>
```

```
            <Parameters>
                <EspSubscriberParameters>
                    <ProjectName>EspProject2</ProjectName>
                    <StreamName>BaseOutput</StreamName>
                </EspSubscriberParameters>
            </Parameters>
        </Module>

        <Module type="formatter">
            <InstanceName>XmlListOutputFormatter</InstanceName>
            <Name>EspToXmlStringFormatter</Name>
            <Next>FileOutputTransporter</Next>
            <Parallel>true</Parallel>
            <Parameters>
            </Parameters>
        </Module>

        <Module type="transporter">
            <InstanceName>FileOutputTransporter</InstanceName>
            <Name>FileOutputTransporter</Name>
            <Parameters>
                <FileOutputTransporterParameters>
                    <Dir>./data</Dir>
                    <File>output.xml</File>
                    <AccessMode>rowBased</AccessMode>
                    <AppendMode>false</AppendMode>
                </FileOutputTransporterParameters>
            </Parameters>
        </Module>
    </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject2</Name>
            <Uri>esp://localhost:19011/sample_workspace/
file_xmllist_output</Uri>
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Adapter Controller Parameters

The Adapter Controller port listens for commands. The `controller.xml` file is located in `%ESP_HOME%/adapters/framework/config` directory. This file is shared among all the adapter within the `%ESP_HOME%/adapters/framework/instances` directory.

Parameter	Description
ControlPort	(Required) Section containing the **MinPort** and **MaxPort** parameters.
MinPort	Type: `int` (Required) The minimum port number that can act as a control port number. The adapter framework allocates this number automatically. A recommended value is 19082.
MaxPort	Type: `int` (Optional) The maximum port number that can act as a control port number. The adapter framework allocates this number automatically. The default value is 65535.
Statistics_Update_Interval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework publishes statistics data to the ESP project to which the adapter is connected. A recommended value is 5000.
MonitorInterval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework checks the status of internal resource allocation. The framework logs a message in the log file if a resource is about to be exhausted. A recommended value is 3000.

Logging

The File XML Record Input and Output adapter use the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file

containing the logging configuration is located in the `%ESP_HOME%\adapters\framework\config` directory.

Specify the location of the logging file you wish to use in the **Log4jProperty** parameter within the adapter configuration file. You can modify the logging levels within this file or the `$ESP_HOME/adapters/framework/config/log4j.properties`, which is used by default. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
```

```
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Starting the File XML Record Adapter

To start the File XML Record adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

   ```
   cd %ESP_HOME%\cluster\nodes\node1
     %ESP_HOME%\bin\esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   %ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

   UNIX:

a. Start the example cluster.

```
cd $ESP_HOME/cluster/nodes/node1
  $ESP_HOME/bin/esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
$ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

2. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: For the File XML Record Input adapter: `cd $ESP_HOME/adapters/framework/instances/file_xmllist_input` For the File XML Record Output adapter: `cd $ESP_HOME/adapters/framework/instances/file_xmllist_output` `./start_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the File XML Record Input adapter: `cd %ESP_HOME%/adapters/framework/instances/file_xmllist_input` For the File XML Record Output adapter: `cd %ESP_HOME%/adapters/framework/instances/file_xmllist_output` `start_adapter.bat <adapter configuration file name>`

Stopping the File XML Record Adapter

To stop the File XML Record adapter from the command line, execute the **stop** command.

Operating System	Step
UNIX	Open a terminal window and enter: For the File XML Record Input adapter: `cd $ESP_HOME/adapters/framework/instances/file_xmllist_input` For the File XML Record Output adapter: `cd $ESP_HOME/adapters/framework/instances/file_xmllist_output` `./stop_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the File XML Record Input adapter: `cd %ESP_HOME%/adapters/framework/instances/file_xmllist_input` For the File XML Record Output adapter: `cd %ESP_HOME%/adapters/framework/instances/file_xmllist_output` `stop_adapter.bat <adapter configuration file name>`

FIX Input Adapter

Adapter type: fixplugin. The SAP Sybase Event Stream Processor FIX adapter is an implementation of the opensource QuickFIX engine that has been integrated with the SAP Sybase Event Stream Processor API.

The FIX adapter:

- Engages in and manages FIX sessions with well-behaved FIX engines
- Receives and sends FIX messages via connectors and FIX sessions
- Validates inbound FIX messages
- Translates FIX messages into Event Stream Processor records
- Translates Event Stream Processor records into FIX messages

Note: The FIX adapter supports customization of the FIX dictionary.

The FIX adapter requires a separately purchased license that you can obtain from the SAP Product Download Site. It supports the standard SySAM grace period, which means you can run it unlicensed for 30 days. After this period, you cannot run the adapter without a valid license.

If you purchased your product from SAP or an authorized SAP reseller, go to the secure SAP Product Download Center (SPDC) at *https://sybase.subscribenet.com* and log in to generate

license keys. The license generation process may vary slightly, depending on whether you ordered directly from SAP or from an SAP reseller.

If you ordered your product under an SAP contract and were directed to download from SAP Service Marketplace (SMP), you can use SMP at *http://service.sap.com/licensekeys* to generate license keys for SAP products that use SySAM 2-based licenses.

See also

- *File FIX Input Adapter* on page 71
- *File FIX Output Adapter* on page 73
- *Socket FIX Input Adapter* on page 857
- *Socket FIX Output Adapter* on page 860

Supported FIX Versions

FIX protocol versions supported by the FIX adapter.

The FIX adapter supports FIX protocol versions 4.0 through 5.0.

Note: FIXML is not supported.

Control Flow

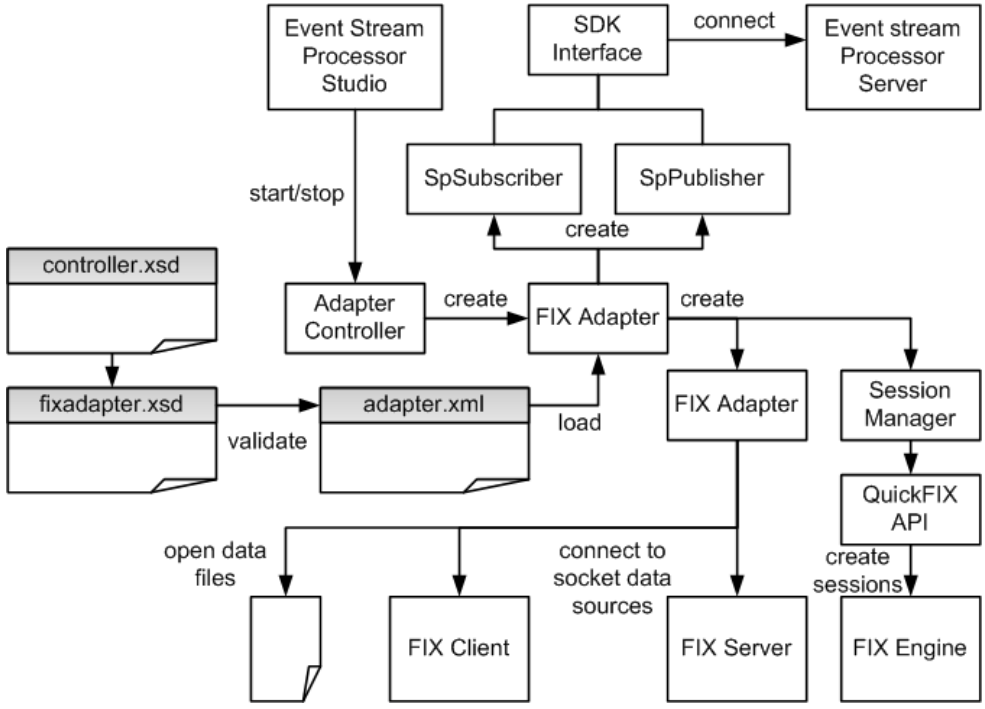
The adapter loads its configuration from a file (for example, `adapter.xml`) and validates it against the adapter schema (`fixadapter.xsd`), which includes the API-wide controller schema (`controller.xsd`).

You cannot edit schemas.

The FIX adapter control flow includes loading different configuration files, and various commands and components.

The Adapter Controller creates an instance of the adapters, and receives and executes user commands. It can execute **start**, **stop**, and **status** commands.

Figure 2: Control Flow



Start Command

The **start** command starts the FIX adapter, configures and starts the command and control interface, loads the FIX dictionary and the SpPublisher and SpSubscriber components, and then connects to the Event Stream Processor via the API interfaces.

The Message Distributor prepares to publish and subscribe to data streams. Data streams are organized into hierarchies named stream clusters. A stream cluster is a set of streams that is capable of hosting FIX messages of a certain type.

The Connector Manager opens FIX data files and socket connections to client and server sources of FIX data, and the Session Manager uses the QuickFIX API to create and log on to sessions with well-behaved FIX engines. The SpSubscriber and SpPublisher components connect to Event Stream Processor via the API interfaces. SpSubscriber starts listening to output streams, and SpPublisher is ready to publish data to input streams.

The adapter ignores the **start** command if it is executed when there is a running instance of the adapter, and sends a warning.

See also

- *Data Streams* on page 165
- *Message Flow* on page 169

- *Starting the FIX Adapter* on page 194

Stop Command

The **stop** command causes the Data Source Handler to close the session and disconnect from the datasource, the Adapter Controller to stop listening to user commands, and the adapter process to terminate.

The Connector Manager closes any open data files and socket connections to client and server datasources, and the Session Manager logs out of existing sessions.

The adapter ignores the **stop** command if it is executed when there is no running instance of the adapter, and a warning is sent.

See also

- *Stopping the FIX Adapter* on page 196

Status Command

The **status** command reports the FIX adapter status, and the Adapter Controller prints out its status: either running or stopped.

See also

- *Checking the FIX Adapter Status* on page 195

Data Streams

Input FIX messages are stored as stream records that are organized into stream clusters.

The FIX adapter stores individual messages in multiple records that belong to a stream hierarchy named stream cluster. The top stream in the stream cluster is called the main stream, and it stores fields that belong to the FIX message. All the other streams in the stream cluster store fields that belong to nested groups.

Note: Messages of the same type can be stored in more than one stream cluster. These clusters do not have to share a common structure.

Store inbound messages in source streams only and outbound messages in any kind of stream.

The FIX adapter ensures proper indexing of records related to inbound messages. Proper indexing of outbound records is the responsibility of the person creating the model.

The adapter `templates` directory contains generated models for all FIX message types. You can use these automatically generated, exhaustive projects to create stream clusters that serve specific business purposes.

See also

- *Message Flow* on page 169
- *Start Command* on page 164

Example: FIX Input Adapter Data Stream

Sample of a FIX Input adapter data stream.

This is a `Quote` type FIX message:

8=FIX.4.4 | 9=204 | 35=S | 49=COUNTERPARTYA | 55=AASymbol | 117=AAQuoteID | 133=31.1 | 453=2 | 448=AAPartyID1 | 447=B | 452=1 | 802=2 | 523=AAPartySubID11 | 803=1 | 523=AAPartySubID12 | 803=2 | 448=AAPartyID2 | 447=C | 452=2 | 802=1 | 523=AAPartySubID21 | 803=3 | 10=107 |

That contains these fields:

- SenderCompID=COUNTERPARTYA (tag 49)
- QuoteID=AAQuoteID (tag 117)
- Symbol=AASymbol (tag 55)
- OfferPx=31.1 (tag 133)
- NoPartyIDs=2 (tag 453)

This is the main stream that stores the FIX message:

```
CREATE  MEMORY  STORE FixStore PROPERTIES  INDEXTYPE ='tree',
INDEXSIZEHINT =8;


CREATE  INPUT  WINDOW MyQuotes
SCHEMA (SenderCompID STRING, QuoteID STRING, NoPartyIDs INTEGER,
Symbol STRING, OfferPx FLOAT, FixMsgId LONG)
PRIMARY KEY (FixMsgId)
 STORE FixStore;
```

The message contains two groups of type NoPartyIDs:

Group 1:

- `PartyID=AAPartyID1 (tag 448)`
- `PartyIDSource=B (tag 447)`
- `PartyRole=1 (Executing Firm, tag 452)`
- `NoPartySubIDs=2 (tag 802)`

Group 2:

- `PartyID=AAPartyID1 (tag 448)`
- `PartyIDSource=C (tag 447)`
- `PartyRole=2 (Broker of Credit, tag 452)`
- `NoPartySubIDs=1 (tag 802)`

Groups 1 and 2 are stored in this stream:

```
CREATE  INPUT  WINDOW MyQuotes_NoPartyIDs
SCHEMA (PartyID STRING, PartyIDSource STRING, PartyRole INTEGER,
NoPartySubIDs INTEGER, FixMsgId LONG, NoPartyIDs_Num LONG)
```

```
PRIMARY KEY (FixMsgId, NoPartyIDs_Num)
 STORE FixStore;
```

Group 1 and Group 2 contain their own groups of type NoPartySubIDs. Groups 11 and 12 below are part of Group 1:

Group 11:

- `PartySubID=AAPartySubID11 (tag 523)`
- `PartySubIDType=1 (Firm, tag 803)`

Group 12:

- `PartySubID=AAPartySubID12 (tag 523)`
- `PartySubIDType=2 (Person, tag 803)`

Group 21 is part of Group 2:

- `PartySubID=AAPartySubID21 (tag 523)`
- `PartySubIDType=3 (System, tag 803)`

Groups 11, 12, and 21 are stored in this stream:

```
CREATE  INPUT  WINDOW MyQuotes_NoPartyIDs_NoPartySubIDs
SCHEMA (PartySubID STRING, PartySubIDType INTEGER, FixMsgId LONG,
NoPartyIDs_Num LONG, NoPartySubIDs_Num LONG)
PRIMARY KEY (FixMsgId, NoPartyIDs_Num, NoPartySubIDs_Num)
 STORE FixStore;
```

Stream and Column Names

Ensure that the field names of the stream columns correspond to FIX fields. The order of columns does not have to follow the order of fields in the FIX dictionary.

Note: Columns unrelated to hosted FIX messages are not allowed.

The names of main streams can be chosen arbitrarily.

Ensure that descendant streams follow a strict naming convention. Since each descendant stream has a parent stream and corresponds to a repeating group, ensure that its name follows this form:

```
<parent stream name>_<name of the repeating group>
```

Header and Trailer Fields

You can add or update header and trailer fields to create a valid FIX message. Some columns may correspond to header or trailer fields. Output connectors keep all fields in the message body intact, as stored in stream columns.

Record Indexing

FIX messages are stored in a hierarchy of records, and cross-referenced using index columns.

Index columns have a `long` type, and are located at the end of the stream.

Records in the main stream have only one index. Child records have two indexes, the first of which must have the same value as the parent record. Descendant records at the next level have three indexes, the first two of which must have the same value as their parent record, and so on.

Adapters and Sessions

The FIX adapter exchanges FIX messages with datasources such as files and socket connections, as well as other FIX engines.

Files and socket connections are handled by the Connection Manager. Sessions with other FIX engines are handled by the Session Manager.

Note: The adapter can work simultaneously with any number of different datasources.

Adapters are one-directional. You can use each adapter to receive messages from (or send them to) a single source of FIX data, such as a file, a client socket connection, or a server socket connection. By default, file and socket adapters validate the checksum and body length tags of all input messages. You can turn validation off in the adapter configuration file.

Sessions with other FIX engines are two-directional. Input messages received via sessions are always validated. You cannot turn off validation for sessions. Session management (login, logout, message sequence numbers, resending of messages, and so on), as well as message validation, is performed by the QuickFIX API.

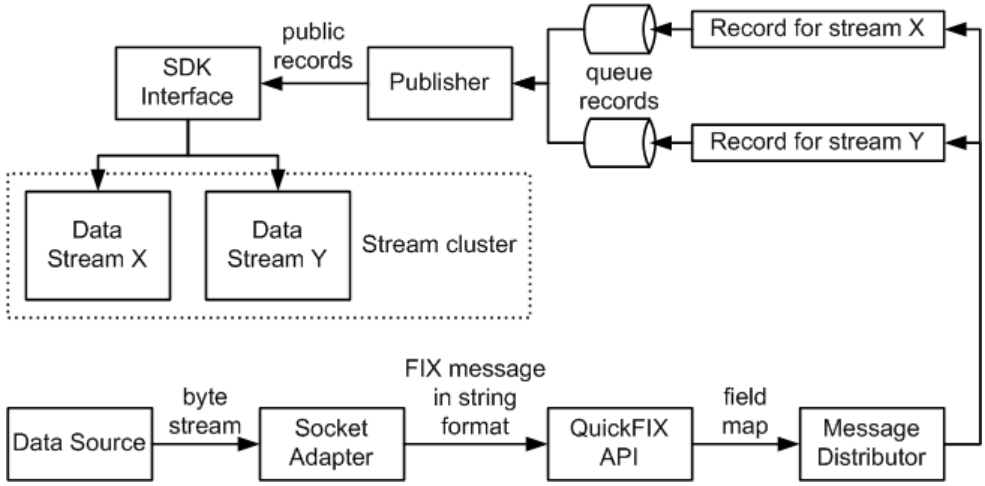
Invalid input messages are discarded and any errors are logged. Otherwise, messages are parsed and published to stream clusters even if their checksum or body length tags are absent or have incorrect values.

Before sending an output message, the adapter recalculates its checksum and body length and updates the appropriate tags.

Message Flow

The message flow through the adapter is initiated by the **start** command.

Figure 3: Inbound Message Flow Through a Socket Connector



The connector receives FIX data as a byte stream. FIX messages are preparsed into string objects. The QuickFIX API parses the strings into field maps, and those field maps are passed on to the Message Distributor.

The Message Distributor converts each field map into a number of records targeting a stream cluster. The records are now ready to be published to Event Stream Processor. However, they are not published immediately. Records are queued, then picked up by the Publisher object on separate threads, one thread for each record queue. You can configure the queue capacity. A larger queue is less likely to overflow in the event of a message burst. When the queue becomes three-quarters full, a warning is logged. Another warning is logged when the queue returns to three-quarters empty. If the queue is full, the adapter waits until room becomes available before placing the next record.

Records are published asynchronously. The adapter receives no feedback from Event Stream Processor.

If you are using the adapter with Event Stream Processor, in the event of a failover, the SDK interface switches to the spare Event Stream Processor instance without message loss.

See also

- *Data Streams* on page 165
- *Start Command* on page 164

Datatype Mapping for the FIX Adapter

Event Stream Processor datatypes map to FIX datatypes.

Event Stream Processor Datatype	QuickFix Datatype
`integer`	`boolean`
`string`	`byte[]`
`string`	`char`
`string`	`string`
`date`	`date`
`float`	`float`
`integer`	`integer`
`date` or `timestamp`	`UTCDateOnly`
`date` or `timestamp`	`UTCTimeOnly`
`date` or `timestamp`	`UTCTimeStamp`

Setting the JAVA_HOME Environment Variable

Set the JAVA_HOME environment variable to point to the Java directory.

Prerequisites

Install Java Runtime Environment version 1.7.0_1 or higher. To see if you have a suitable version of Java installed, go to *http://www.java.com/en/download/installed.jsp*.

Task

Set the JAVA_HOME environment variable to the directory path where Java Runtime Environment 1.7.0_1 or higher is installed.

Next

Verify that the ESP_HOME environment variable is set correctly.

Configuration

Configuration information for the FIX adapter.

FIX Adapter Directory

The adapter directory contains all files, such as configuration files, templates, examples, and JAR files, relating to the adapter.

```
README.txt    Documentation note
ReleaseNotes.txt        Release notes

bin/
    start_adapter.bat             Standalone adapter startup script
    start_adapter.sh              Standalone adapter startup script
    adapter-plugin.bat       Plug-in connector startup script
    adapter-plugin.sh        Plug-in connector startup script

config/
       controller.xsd        Controller schema
       fixadapter.xsd        Adapter schema
       log4j.properties      Sample logging configuration
       login.config          Authentication configuration

dictionary/
           FIX40.xml         FIX 4.0 dictionary
           FIX41.xml         FIX 4.1 dictionary
           FIX42.xml         FIX 4.2 dictionary
           FIX43.xml         FIX 4.3 dictionary
           FIX44.xml         FIX 4.4 dictionary

examples/                              Working examples
         AllInOne/
         ClientSocketConnectors/
         FileConnectors/
         ServerSocketConnectors/

libj/

    commons-codec-1.3.jar Required by SDK API
    commons-collections-3.2.1.jar
    commons-configuration-1.6.jar
    commons-lang-2.6.jar
    commons-logging-1.1.jar Logging library
    esp_adapter_api.jar Adapter API code
    esp_adapter_fix.jar FIX Adapter code
    esp_i18n.jar
    esp_license.jar
    esp_sdk.jar ESP SDK library
    log4j-1.2.16.jar Logging library
    mina-core-1.1.0.jar
    postgresql.jar
    quickfixj-all-1.5.0.jar
    slf4j-api-1.6.1.jar
    slf4j-log4j12-1.6.1.jar
    sylapi.jar
    ws-commons-util-1.0.2.jar Required by ESP SDK
    xerces-impl-2.9.1.jar XML parser library
    xmlrpc-client-3.1.3.jar Required by ESP SDK
```

```
    xmlrpc-common-3.1.3.jar Required by ESP SDK
templates/            Sample stream description
utils/
```

Schema and Configuration File

The adapter configuration is loaded from a file and validated against the adapter schema.

Ensure that the FIX adapter configuration file is placed into the `$ESP_HOME/adapters/fix/config` before you start the adapter, and that the adapter configuration validates against the schema.

The `$ESP_HOME/adapters/fix/examples` folder contains sample adapter configuration files. You can edit any of these files or write a new one.

Note: The adapter manager looks for either `<sp>` or `<sdk>` node in the configuration file. An `<sp>` node indicates a connection to Event Stream Processor.

Adapter Controller Parameters

The Adapter Controller port listens for commands.

Parameter Name	Description
controllerPort	Type: `positive integer` (Required) Specifies the adapter command and control port. User commands are sent to this port on localhost. Ensure that each adapter instance has its own dedicated port.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the FIX adapter.

These parameters are defined in the `controller.xsd` file in the `config` directory.

Parameter Name	Description
espAuthType	Type: `string` (Required) Specifies the method used to authenticate to Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **espAuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
espUser	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **espAuthType**). No default value.
espPassword	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **espPassword** value is encrypted. Default value is false. If set to true, the password value is decrypted using **espRSAKeyStore** and **espRSAKeyStorePassword**.
espProjectUri	Type: `string` (Required) Specifies the total project URI to connect to Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
pulseInterval	Type: `non-negative integer` (Optional) Specifies the time interval, in seconds, during which outbound record changes are coalesced by Event Stream Processor, then received by the adapter as a single event. If not set or set to 0, record changes are received individually as they occur.

Parameter Name	Description
espHeartbeatPeriod	Type: `positive integer` (Optional) Specifies the length of time, in seconds, that the adapter waits before sending the next heartbeat to Event Stream Processor. If Event Stream Processor fails to receive two consecutive heartbeats, all records published by the adapter are marked stale. Default value is 10.
recordQueueCapacity	Type: `positive integer` (Optional) Specifies capacity of the record queues. Default value is 4096.
maxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.
maxPubPoolTime	Type: `positive integer` (Optional) Specifies the maximum period of time, in milliseconds, for which records are pooled before being published. If not set, pooling time uses default value and the pooling strategy is governed by **maxPubPoolSize**. Default value is 5000.
useTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.
espRSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **espAuthType** is set to server_rsa, or the encrypted attribute for **espPassword** is set to true, or both.

Parameter Name	Description
espRSAKeyStorePassword	Type: `string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **espAuthType** is set to server_rsa, or the encrypted attribute for **espPassword** is set to true, or both.
espKerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **espAuthType** is set to kerberos.
espKerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **espAuthType** is set to kerberos.
espKerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **espAuthType** is set to kerberos.
espKerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **espAuthType** is set to kerberos.
espEncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **espPassword** is set to true. If left blank, RSA is used as default.

FIX Input Adapter

The FIX Input adapter reads FIX messages from any number of file, socket, and session connectors.

Property Label	Description
Adapter Directory Path	Property ID: **baseDir** Type: `directory` (Required) Specifies the path to the adapter base directory. No default value. This property is ignored if the Connector Remote Directory Path property is supplied. Use a forward slash for both UNIX and Windows paths.

Property Label	Description
Configuration File Path	Property ID: **configFilePath** Type: `configFilename` (Required) Specifies the absolute path to the adapter configuration file. No default value. This property is ignored if the Remote Configuration File Path property is supplied. Use a forward slash for both UNIX and Windows paths.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.
Adapter Remote Directory	Property ID: **remoteBaseDir** Type: `string` (Advanced) Specifies the path to the adapter remote base directory (for remote execution only). No default value. If this property is supplied, the Connector Directory Path property is ignored. Use a forward slash for both UNIX and Windows paths.
Adapter Configuration File Path	Property ID: **remoteConfigFilePath** Type: `string` (Advanced) Specifies the absolute path to the adapter configuration file (for remote execution only). No default value. If this property is supplied, the configuration file path property is ignored. Use a forward slash for both UNIX and Windows paths.

Event Stream Proccessor Server Properties

The Server connection properties are provided as attributes and subelements of the `<sdk>` node in the configuration file.

Parameter Name	Description
server	(Required) Specifies the attribute of the **sdk** node.
serverHost	(Required) Specifies name of the machine the Server is running on.
serverPort	(Required) Specifies port number the Server is listening on.
serverWorkspace	(Required) Workspace.
serverProject	(Required) Program name.
serverUser	(Optional) User name.
serverPassword	(Optional) Password.

FIX Dictionary

The FIX adapter dictionary contains the definitions of FIX message types, components, and fields.

Parameter Name	Description
fixDictionary	Type: `string` (Required) Name of the FIX dictionary file. Dictionary files for all supported FIX versions (4.0 through 5.0) are provided in the `dictionary` folder. You can edit the definitions of FIX message types, components, and fields.

Stream Configuration

Use the `streams` section in the configuration file to map FIX message types to stream clusters.

Parameter Name	Description
name	Type: `string` (Required) Name of the main stream in a stream cluster. No default value.

Parameter Name	Description
messageName	Type: `string` (Required) Name of a message type hosted on the stream cluster. No default value.

To host FIX messages of type `Quote` in a stream cluster descending from the `MyQuotes` stream, add this fragment to the `<streams>` group:

```
<stream>
    <name>MyQuotes</name>
    <messageName>Quote</messageName>
</stream>
```

Connectors

The `connector` section in the FIX configuration file defines the file and socket connectors.

Parameter Name	Description
streamNames	Type: `streamNameType` (Required) Lists the names of the main streams in stream clusters where messages coming through this connector are hosted.

Inbound and Outbound Connectors

The **inbound** and **outbound** parameter in the FIX configuration file lists inbound and outbound file and socket connectors.

Parameter Name	Description
doValidation	Type: `boolean` (Optional) If set to true, inbound messages coming through this connector are validated for correct message length and checksum. If set to false, the message length and checksum fields are ignored. Invalid messages are discarded and the errors are logged. Message data is validated against the data dictionary during the parsing of the message. Default value is true.

See also

- *Example: Using All-In-One* on page 206

Sample Configuration File for All-In-One Connectors

Sample configuration file (`adapter.xml`) for the all-in-one connectors in the FIX adapter.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
```

```
 Sybase ESP FIX adapter configuration file
-->
- <adapter>
- <!--
 Adapter Controller
-->
- <controller>
<controllerPort>13579</controllerPort>
</controller>
- <!--
 Event Stream Processor settings
-->
- <esp>
- <espConnection>
<espProjectUri>esp://localhost:19011/w1/p1</espProjectUri>
</espConnection>
- <espSecurity>
<espUser>espuser</espUser>
<espPassword encrypted="false">espuser</espPassword>
<espAuthType>none</espAuthType>
- <!--
        <espRSAKeyStore>/keystore/keystore.jks</espRSAKeyStore>
        <espRSAKeyStorePassword>Sybase123</espRSAKeyStorePassword>
-->
<espEncryptionAlgorithm>RSA</espEncryptionAlgorithm>
</espSecurity>
</esp>
- <!--
 FIX dictionary
-->
<fixDictionary>FIX44.xml</fixDictionary>
- <!--
 Stream cluster to FIX message mapping
-->
- <streams>
- <stream>
<name>MyQuotes</name>
<messageName>Quote</messageName>
</stream>
- <stream>
<name>MyOrders</name>
<messageName>NewOrderSingle</messageName>
</stream>
</streams>
- <!--
 Connectors
-->
- <connectors>
- <outbound>
- <fileConnector>
<fileName>orders.fix</fileName>
- <streamNames>
<streamName>MyOrders</streamName>
</streamNames>
</fileConnector>
</outbound>
```

```
</connectors>
- <!--
 FIX Session Settings
-->
- <sessionSettings>
- <default>
<ConnectionType>acceptor</ConnectionType>
<SocketAcceptPort>23456</SocketAcceptPort>
<FileLogPath>logs</FileLogPath>
<FileStorePath>store</FileStorePath>
<DataDictionary>FIX44.xml</DataDictionary>
<HeartBtInt>600</HeartBtInt>
<BeginString>FIX.4.4</BeginString>
<StartTime>00:00:00</StartTime>
<EndTime>23:59:59</EndTime>
<SenderCompID>SYBASE</SenderCompID>
</default>
- <sessionSetting>
<TargetCompID>COUNTERPARTYA</TargetCompID>
</sessionSetting>
- <sessionSetting>
<TargetCompID>COUNTERPARTYB</TargetCompID>
</sessionSetting>
</sessionSettings>
- <!--
 Session logins
-->
- <sessionLogins>
- <senderLogin>
<username>MyUsername</username>
<password>MyPassword</password>
<NextExpectedMsgSeqNum>1</NextExpectedMsgSeqNum>
</senderLogin>
- <targetLogin>
<TargetCompID>COUNTERPARTYA</TargetCompID>
<username>UsernameA</username>
<password>PasswordA</password>
</targetLogin>
- <targetLogin>
<TargetCompID>COUNTERPARTYB</TargetCompID>
<username>UsernameB</username>
<password>PasswordB</password>
</targetLogin>
</sessionLogins>
- <!--
 Sessions
-->
- <sessions>
- <inbound>
- <session>
<TargetCompID>COUNTERPARTYA</TargetCompID>
- <streamNames>
<streamName>MyQuotes</streamName>
</streamNames>
</session>
</inbound>
```

```
- <outbound>
- <session>
<TargetCompID>COUNTERPARTYA</TargetCompID>
- <streamNames>
<streamName>MyOrders</streamName>
</streamNames>
</session>
- <session>
<TargetCompID>COUNTERPARTYB</TargetCompID>
- <streamNames>
<streamName>MyQuotes</streamName>
</streamNames>
</session>
</outbound>
</sessions>
</adapter>
```

File Connectors

The **fileConnector** parameter in the FIX configuration file lists property values for file connectors.

Parameter Name	Description
fileName	Type: `string` (Required) The relative or absolute path to the file containing FIX data. Use a forward slash for both UNIX and Windows paths.
streamNames	Type: `streamNamesType` (Required) Lists the names of the main streams in stream clusters where messages coming through this connector are hosted.
doValidation	Type: `boolean` (Optional) If set to true, inbound messages coming through this connector are validated for correct message length and checksum. If set to false, the message length and checksum fields are ignored. Invalid messages are discarded and the errors are logged. Message data is validated against the data dictionary during the parsing of the message. Default value is true.
inputBuffer	Type: `integer` (Optional) Enables the adapter to process a specified number of kilobytes from the file. Default value is 1024.

See also

- *Example: Using File Connectors* on page 197

Sample Configuration File for File Connectors

Sample configuration file (`adapter.xml`) for the file connectors in the FIX adapter.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
 Sybase ESP FIX adapter configuration file
-->
- <adapter>
- <!--
 Adapter Controller
-->
- <controller>
<controllerPort>13579</controllerPort>
</controller>
- <!--
 Event Stream Processor settings
-->
- <esp>
- <espConnection>
<espProjectUri>esp://localhost:19011/w1/p1</espProjectUri>
</espConnection>
- <espSecurity>
<espUser>espuser</espUser>
<espPassword encrypted="false">espuser</espPassword>
<espAuthType>none</espAuthType>
- <!--
        <espRSAKeyStore>/keystore/keystore.jks</espRSAKeyStore>
        <espRSAKeyStorePassword>Sybase123</espRSAKeyStorePassword>
-->
<espEncryptionAlgorithm>RSA</espEncryptionAlgorithm>
</espSecurity>
</esp>
- <!--
 FIX dictionary
-->
<fixDictionary>FIX44.xml</fixDictionary>
- <!--
 Stream cluster to FIX message mapping
-->
- <streams>
- <stream>
<name>MyQuotes</name>
<messageName>Quote</messageName>
</stream>
- <stream>
<name>MyOrders</name>
<messageName>NewOrderSingle</messageName>
</stream>
</streams>
- <!--
 Connectors
-->
```

```
- <connectors>
- <inbound>
- <fileConnector>
<fileName>quotes.fix</fileName>
<doValidation>false</doValidation>
- <streamNames>
<streamName>MyQuotes</streamName>
</streamNames>
</fileConnector>
</inbound>
- <outbound>
- <fileConnector>
<fileName>orders.fix</fileName>
- <streamNames>
<streamName>MyOrders</streamName>
</streamNames>
</fileConnector>
</outbound>
</connectors>
- <!--
 Sessions
-->
<sessions />
</adapter>
```

Client Socket Connectors

The **clientSocketConnector** parameters in the FIX configuration file define the name, IP address, validation scheme, stream names, and port of the Server to send FIX messages to.

Parameter Name	Description
dataHost	Type: `string` (Required) Specifies the name or IP address of the server to send FIX messages to.
dataPort	Type: `nonNegativeInteger` (Required) Specifies the port to connect to.
streamNames	Type: `streamNameType` (Required) Lists the names of the main streams in stream clusters where messages coming through this connector are hosted.

Parameter Name	Description
doValidation	Type: `boolean` (Optional) If set to true, inbound messages coming through this connector are validated for correct message length and checksum. If set to false, the message length and checksum fields are ignored. Invalid messages are discarded and the errors are logged. Message data is validated against the data dictionary during the parsing of the message. Default value is true.

See also

- *Example: Using Client Socket Connectors* on page 200

Sample Configuration File for Client Socket Connectors

Sample configuration file (`adapter.xml`) for the client socket connectors in the FIX adapter.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
 Sybase ESP FIX adapter configuration file
-->
- <adapter>
- <!--
 Adapter Controller
-->
- <controller>
<controllerPort>13579</controllerPort>
</controller>
- <!--
 Event Stream Processor settings
-->
- <esp>
- <espConnection>
<espProjectUri>esp://localhost:19011/w1/p1</espProjectUri>
</espConnection>
- <espSecurity>
<espUser>espuser</espUser>
<espPassword encrypted="false">espuser</espPassword>
<espAuthType>none</espAuthType>
- <!--
        <espRSAKeyStore>/keystore/keystore.jks</espRSAKeyStore>
        <espRSAKeyStorePassword>Sybase123</espRSAKeyStorePassword>
-->
<espEncryptionAlgorithm>RSA</espEncryptionAlgorithm>
</espSecurity>
</esp>
- <!--
 FIX dictionary
-->
```

```
<fixDictionary>FIX44.xml</fixDictionary>
- <!--
 Stream cluster to FIX message mapping
-->
- <streams>
- <stream>
<name>MyQuotes</name>
<messageName>Quote</messageName>
</stream>
- <stream>
<name>MyOrders</name>
<messageName>NewOrderSingle</messageName>
</stream>
</streams>
- <!--
 Connectors
-->
- <connectors>
- <inbound>
- <clientSocketConnector>
<dataHost>localhost</dataHost>
<dataPort>43210</dataPort>
<doValidation>true</doValidation>
- <streamNames>
<streamName>MyQuotes</streamName>
</streamNames>
</clientSocketConnector>
</inbound>
- <outbound>
- <clientSocketConnector>
<dataHost>localhost</dataHost>
<dataPort>32109</dataPort>
- <streamNames>
<streamName>MyOrders</streamName>
</streamNames>
</clientSocketConnector>
</outbound>
</connectors>
- <!--
 Sessions
-->
<sessions />
</adapter>
```

Server Socket Connectors

The **serverSocketConnector** parameter in the FIX configuration file defines the port on which the Server is listening for client connections.

Parameter Name	Description
dataPort	Type: `nonNegativeInteger` (Required) Specifies the port the server is listening on for client connections.

Parameter Name	Description
streamNames	Type: `streamNameType` (Required) Lists the names of the main streams in stream clusters where messages coming through this connector are hosted.
doValidation	Type: `boolean` (Optional) If set to true, inbound messages coming through this connector are validated for correct message length and checksum. If set to false, the message length and checksum fields are ignored. Invalid messages are discarded and the errors are logged. Message data is validated against the data dictionary during the parsing of the message. Default value is true.

See also

- *Example: Using Server Socket Connectors* on page 203

Sample Configuration File for Server Socket Connectors

Sample configuration file (`adapter.xml`) for the server socket connectors in the FIX adapter.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
 Sybase ESP FIX adapter configuration file
-->
- <adapter>
- <!--
 Adapter Controller
-->
- <controller>
<controllerPort>13579</controllerPort>
</controller>
- <!--
 Event Stream Processor settings
-->
- <esp>
- <espConnection>
<espProjectUri>esp://localhost:19011/w1/p1</espProjectUri>
</espConnection>
- <espSecurity>
<espUser>espuser</espUser>
<espPassword encrypted="false">espuser</espPassword>
<espAuthType>none</espAuthType>
- <!--
        <espRSAKeyStore>/keystore/keystore.jks</espRSAKeyStore>
        <espRSAKeyStorePassword>Sybase123</espRSAKeyStorePassword>
-->
```

```
<espEncryptionAlgorithm>RSA</espEncryptionAlgorithm>
</espSecurity>
</esp>
- <!--
 FIX dictionary
-->
<fixDictionary>FIX44.xml</fixDictionary>
- <!--
 Stream cluster to FIX message mapping
-->
- <streams>
- <stream>
<name>MyQuotes</name>
<messageName>Quote</messageName>
</stream>
- <stream>
<name>MyOrders</name>
<messageName>NewOrderSingle</messageName>
</stream>
</streams>
- <!--
 Connectors
-->
- <connectors>
- <inbound>
- <serverSocketConnector>
<dataPort>54321</dataPort>
<doValidation>true</doValidation>
- <streamNames>
<streamName>MyQuotes</streamName>
</streamNames>
</serverSocketConnector>
</inbound>
- <outbound>
- <serverSocketConnector>
<dataPort>43210</dataPort>
- <streamNames>
<streamName>MyOrders</streamName>
</streamNames>
</serverSocketConnector>
</outbound>
</connectors>
- <!--
 Sessions
-->
<sessions />
</adapter>
```

Session Settings

Properties in the `sessionSettings` section configure default and specific settings for session connections with the FIX engine.

Default Settings

Properties that configure default settings for all session connections.

Property Name	Description
ConnectionType	Type: `string` (Required) Specifies whether the adapter acts as a server or a client. Each adapter instance operates in either acceptor mode or initiator mode, but cannot simultaneously operate in both modes. Valid values are: • **Acceptor –** the adapter acts as a server accepting connection requests from FIX session initiators. • **Initiator –** the adapter acts as a client connecting to FIX session acceptors. No default value.
SocketAcceptPort	Type: `nonNegativeInteger` (Required) Specifies the port on which the adapter listens for connections from FIX session initiators. No default value. Operates only in initiator mode.
FileLogPath	Type: `string` (Required) Specifies the directory path for message logs. Both absolute and relative paths are accepted. No default value. Use a forward slash for both UNIX and Windows paths.
FileStorePath	Type: `string` (Required) Specifies the directory path for message stores. Both absolute and relative paths are accepted. No default value. Use a forward slash for both UNIX and Windows paths.
StartTime	Type: `string` (Required) Specifies the time the FIX session is activated. No default value.
EndTime	Type: `string` (Required) Specifies the time the FIX session is deactivated. No default value.

Property Name	Description
DataDictionary	Type: `string` (Required) Specifies the absolute or relative paths to the FIX dictionary file path. No default value. Use a forward slash for both UNIX and Windows paths.
BeginString	Type: `string` (Required) Specifies the value of the **BeginString** field (tag 8) in outbound FIX messages. No default value.
SenderCompID	Type: `string` (Required) Specifies the value of the **SenderCompID** field (tag 49) in outbound FIX messages. An adapter instance uses the same **SenderCompID** value for all session connections.
HeartBtInt	Type: `nonNegativeInteger` (Optional) Specifies the heartbeat interval, in seconds. Default value is 10. Operates only in initiator mode.
ReconnectInterval	Type: `positiveInteger` (Optional) Specifies the interval, in seconds, between reconnection attempts. The adapter keeps trying to reconnect if it fails to connect to the acceptor engine at start-up or if the connection is lost afterward. Default value is 30. Operates only in initiator mode.
LogonTimeout	Type: `nonNegativeInteger` (Optional) Specifies the number of seconds to wait for a logon response before disconnecting from the acceptor engine. Default value is 10. Operates only in initiator mode.
LogoutTimeout	Type: `nonNegativeInteger` (Optional) Specifies the number of seconds to wait for a logout response before disconnecting from the acceptor engine. Default value is 2. Operates only in initiator mode.

Specific Settings

Properties that configure specific settings for specific session connections.

Property Name	Description
TargetCompID	Type: `string` (Required) Specifies the value of the **TargetCompID** field (tag 56) in outbound FIX messages.
SocketConnectHost	Type: `string` (Required) Specifies the acceptor engine's host name or IP address. Operates only in initiator mode.
SocketConnectPort	Type:`non-negative integer` (Required) Specifies the port on which the acceptor engine is listening for connections. Operates only in initiator mode .

Session Logins

Properties in the `sessionLogins` section configure sender and target login properties for session connections with the FIX engine.

Sender Login Properties

If specified, the adapter attaches **senderLogin** properties to outbound login messages at the beginning of FIX sessions.

Parameter Name	Description
username	Type: `string` (Required) Specifies the sender's user name.
password	Type: `string` (Required) Specifies the sender's password.
NextExpectedMsgSeqNum	Type: `string` (Required) Specifies the value of the **NextExpectedMsg-SeqNum** field (tag 789) in outbound login messages.

Target Login Properties

For each inbound login message, the FIX adapter tries to match the values of the **username** and **password** fields with the ones specified for the corresponding **TargetCompID** field.

Note: An error is logged if the user names or the passwords do not match.

Parameter Name	Description
username	Type: `string` (Required) Specifies the target's user name.
password	Type: `string` (Required) Specifies the target's password.
TargetCompID	Type: `string` (Required) Specifies the value of the **TargetCompID** field (tag 56) in inbound login messages.

Session Properties

Properties in **sessionProperties** sections identify inbound and outbound FIX sessions and indicate the main streams of the stream clusters that host data exchanged during FIX sessions.

Property Name	Description
TargetCompID	Type: `string` (Required) Specifies the session identifier. An inbound and an outbound session with the same identifier are implemented as a single two-way session. No default value.
streamNames	Type: `streamNamesType` (Required) Lists the names of the main streams in stream clusters, where messages exchanged over this FIX session are hosted. • For inbound sessions, unmapped messages are ignored; mapped messages are written to all stream clusters that they are mapped to. • For outbound sessions, the header and trailer fields are added or updated, if necessary, to ensure the validity of the outgoing FIX messages. Two stream clusters hosting messages of the same type are not required to share the same structure.

Example: Receiving and Hosting Inbound Messages

Receive inbound messages from a specified target, and host it in a specified stream cluster with a specified main stream.

Inbound messages received over a FIX session from a target identified as `COUNTERPARTYA` are hosted in a stream cluster for which `MyQuotes` is the main stream. Messages of types other than Quote are ignored.

```
<inbound>
    <session>
        <TargetCompID>COUNTERPARTYA</TargetCompID>
        <streamNames>
            <streamName>MyQuotes</streamName>
        </streamNames>
    </session>
</inbound>
```

Logging

The FIX Input adapter uses Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing the logging configuration is part of the FIX Input adapter distribution.

You can modify the logging levels of the `log4j.properties` configuration file which is located in the `%ESP_HOME%\adapters\<adapter-name>\config` directory. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Duplicate Messages

Configure the <enable> and <statusDirectory> properties to avoid having the adapter input and output duplicate messages.

By default, after the FIX adapter restarts, the FixMsgId value is reset to 0, which results in duplicate messages. To change this behavior and have the adapter log the last FixMsgId values

for the input and output streams instead, add the following to your adapter configuration file (`adapter.xml`):

```
<resume>
  <enable>true</enable>
  <statusDirectory></statusDirectory>
 </resume>
```

Set the <enable> tag to true to enable this behavior, and specify the FixMsgId log location in the <statusDirectory>. If you do not specify a location, the value defaults to the current directory. To reset the FixMsgIds to 0, delete the FixMsgId log files before restarting the FIX adapter. Note that the FixMsgId gets reset to 0 whenever the FIX adapter is started with this feature disabled.

As a result of enabling this behavior, when you restart the FIX adapter, the adapter compares the last FixMsgIds in the logs with the values in the rows it receives from the input streams, and only sends the FIX messages which have a newer FixMsgId to the FIX server. This prevents duplicate orders and other transactions from being sent to the FIX server. On the output side, the adapter uses the last logged FixMsgId +1 rather than starting at 0, which prevents rows with duplicate key values from being published to Event Stream Processor.

Operation

Operate the FIX adapter from the command line.

To allow the `adapter.xml` configuration to be placed in any desired location, ensure that the full file path appears along with the **start**, **stop**, and **status** commands.

Note: You can define long file path names as environment variables.

Starting the FIX Adapter

To start the FIX adapter from the command line, start Event Stream Processor, verify parameters, and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

   ```
   cd %ESP_HOME%\cluster\nodes\node1
     %ESP_HOME%\bin\esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   %ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

```
%ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

UNIX:

a. Start the example cluster.

```
cd $ESP_HOME/cluster/nodes/node1
  $ESP_HOME/bin/esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
$ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

2. Start the adapter.

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/fix/bin`
	`./adapter.sh <configuration file path> start`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/fix/bin`
	`adapter.bat <configuration file path> start`

You can use the **esp_subscribe** utility to ensure that FIX messages are successfully published to Event Stream Processor.

See also

- *Start Command* on page 164

Checking the FIX Adapter Status

Run the **status** command in a terminal or command window to check adapter status. Check the FIX adapter status:

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/fix/bin`
	`./adapter.sh <configuration file path> status`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/fix/bin`
	`adapter.bat <configuration file path> status`

You see the adapter status: running or stopped.

See also

- *Status Command* on page 165

Stopping the FIX Adapter

Run the **stop** command in a terminal or command window to stop the adapter.

Prerequisites

When you are running the adapter from the command line, stop the adapter first before stopping the project.

Task

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/fix/bin`
	`./adapter.sh $ADAPTER stop &`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/fix/bin`
	`adapter.bat %ADAPTER% stop`

See also

- *Stop Command* on page 165

Examples

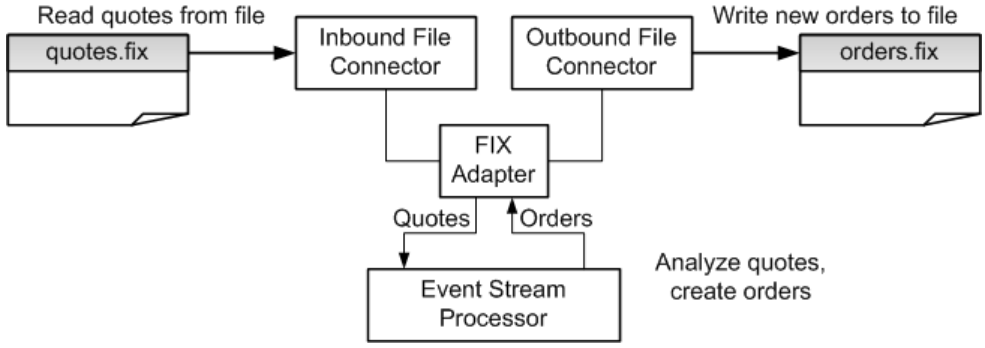
Use the working examples provided with the adapter to learn how to subscribe to real-time market data and publish data to Event Stream Processor.

The scripts provided with the examples do not require a network connection.

Example: Using File Connectors

Use file connectors to read Quote messages from a file, and publish them to the Event Stream Processor. If the value of the OfferPx field is less than 30.0, write a NewOrderSingle message to another file.

Figure 4: File Connectors



1. Set the username and password in the example environment:

Operating System	Step
UNIX	1. Edit the `set_example_env.sh` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	1. Edit the `set_example_env.bat` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

2. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

3. Start the respective subscriber utility for Event Stream Processor.

Operating System	Step
UNIX	Open a terminal window and enter: `./esp-subscribe.sh`
Windows	Open a command window and enter: `esp-subscribe.bat`

4. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: `start_adapter.sh`
Windows	Open a command window and enter: `start_adapter.bat`

5. Wait five to ten seconds for the adapter to initialize.
   - The MyQuotes stream now contains two records. The MyOrders stream contains one record. Use the Event Stream Processor subscriber utility to view the content of the streams.
   - The `orders.fix` file now contains a NewOrderSingle message.

See also

- *File Connectors* on page 181

Example: Using File Connectors With Batch Publishing

Use file connectors to read Quote messages from a file, and publish them to Event Stream Processor. The adapter uses the FIX `ABFIX44.xml` dictionary file, which is used for this example only.

Prerequisites

Set the value of the **maxPubPoolSize** parameter to 2000 to enable batch publishing (also referred to as record pooling or block publishing).

Task

1. Set the username and password in the example environment:

Operating System	Step
UNIX	1. Edit the `set_example_env.sh` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	1. Edit the `set_example_env.bat` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

2. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

3. Start the respective subscriber utility for Event Stream Processor.

Operating System	Step
UNIX	Open a terminal window and enter: `./esp-subscribe.sh`
Windows	Open a command window and enter: `esp-subscribe.bat`

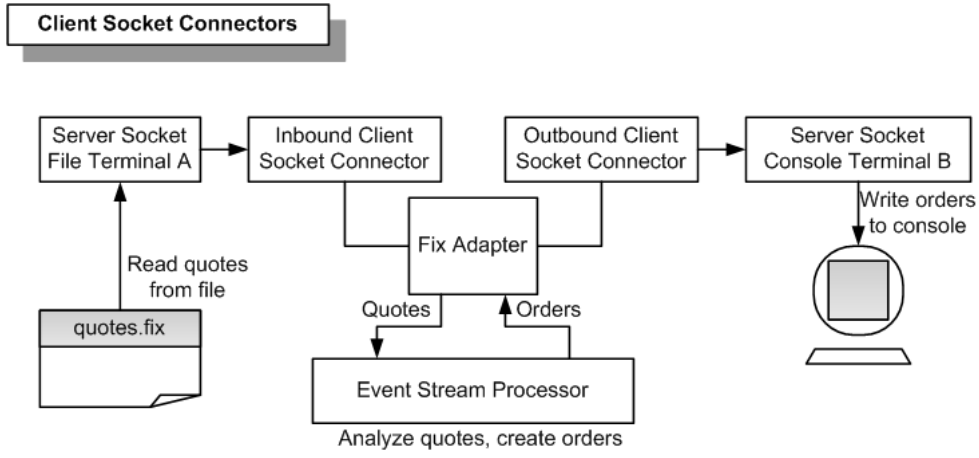
4. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: `start_adapter.sh`
Windows	Open a command window and enter: `start_adapter.bat`

5. Wait five to ten seconds for the adapter to initialize.
   - There are a total of 10096 records in the `8kmsg.fix` file. The FIX adapter reads these records and publishes them to the ESP server in batches of 2000.
   - After 10000 records are published, the adapter waits five seconds, then publishes the remaining 96 records.

Example: Using Client Socket Connectors

Use client socket connectors to read Quote messages from a server socket, and publish them to the Event Stream Processor. If the value of the OfferPx field is less than 30.0, the adapter writes a NewOrderSingle message to another server socket.



1. Set the username and password in the example environment:

Operating System	Step
UNIX	1. Edit the `set_example_env.sh` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	1. Edit the `set_example_env.bat` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

2. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

3. Start the respective subscriber utility for Event Stream Processor.

Operating System	Step
UNIX	Open a terminal window and enter: `./esp-subscribe.sh`
Windows	Open a command window and enter: `esp-subscribe.bat`

4. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: `start_adapter.sh`

Operating System	Step
Windows	Open a command window and enter: `start_adapter.bat`

5. Wait five to ten seconds for the adapter to initialize.
6. Start server socket terminal B.

Operating System	Step
UNIX	Open a terminal window and enter: `./terminalB.sh`
Windows	Open a command window and enter: `terminalB.bat`

7. Start server socket terminal A.

Operating System	Step
UNIX	Open a terminal window and enter: `./terminalA.sh`
Windows	Open a command window and enter: `terminalA.bat`

- The MyQuotes stream now contains two records, and the MyOrders stream contains one record. Use the Event Stream Processor subscriber utility to view the content of the streams.
- The terminal B console window now contains a NewOrderSingle message.

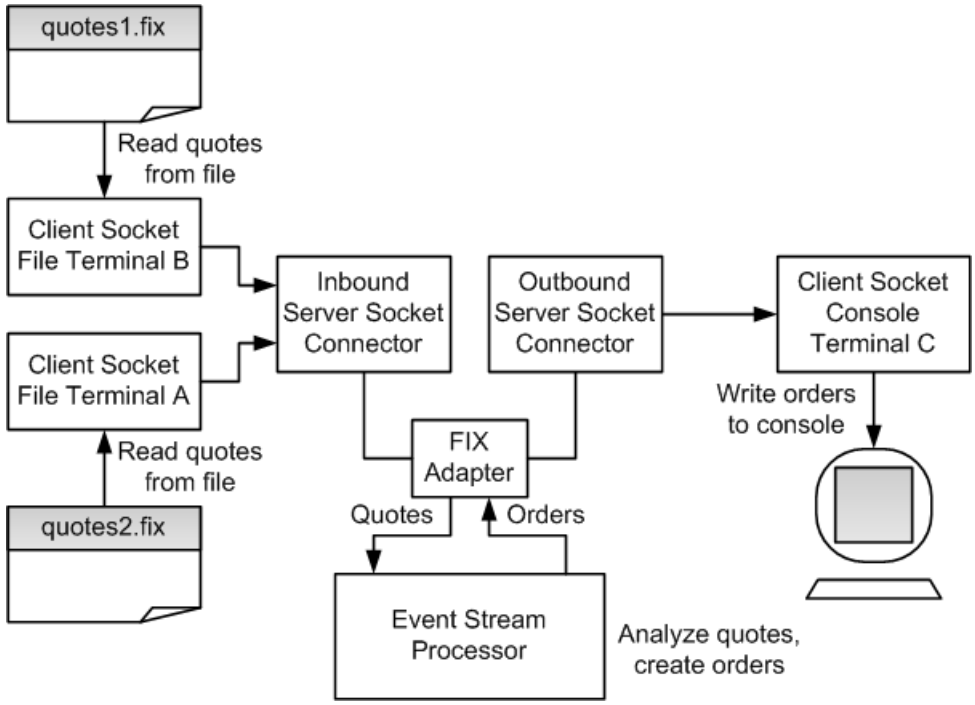
See also

- *Client Socket Connectors* on page 183

Example: Using Server Socket Connectors

Use server socket connectors to read Quote messages from two client sockets, and publish them to the Event Stream Processor. If the value of the OfferPx field is less than 30.0, the FIX adapter writes a NewOrderSingle message to a third client socket.

Figure 5: Server Socket Connectors



1. Set the username and password in the example environment:

Operating System	Step
UNIX	1. Edit the `set_example_env.sh` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	1. Edit the `set_example_env.bat` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

2. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

3. Start the respective subscriber utility for Event Stream Processor.

Operating System	Step
UNIX	Open a terminal window and enter: `./esp-subscribe.sh`
Windows	Open a command window and enter: `esp-subscribe.bat`

4. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: `start_adapter.sh`
Windows	Open a command window and enter: `start_adapter.bat`

5. Wait five to ten seconds for the adapter to initialize.
6. Start output terminal C.

Operating System	Step
UNIX	Open a terminal window and enter: `./terminalC.sh`
Windows	Open a command window and enter: `terminalC.bat`

7. Start output terminal B.

Operating System	Step
UNIX	Open a terminal window and enter: `./terminalB.sh`
Windows	Open a command window and enter: `terminalB.bat`

8. Start output terminal A.

Operating System	Step
UNIX	Open a terminal window and enter: `./terminalA.sh`
Windows	Open a command window and enter: `terminalA.bat`

- The MyQuotes stream now contains three records. The MyOrders stream contains one record. Use the Event Stream Processor subscriber utility to view the content of the streams.
- The terminal C console window now contains a NewOrderSingle message.

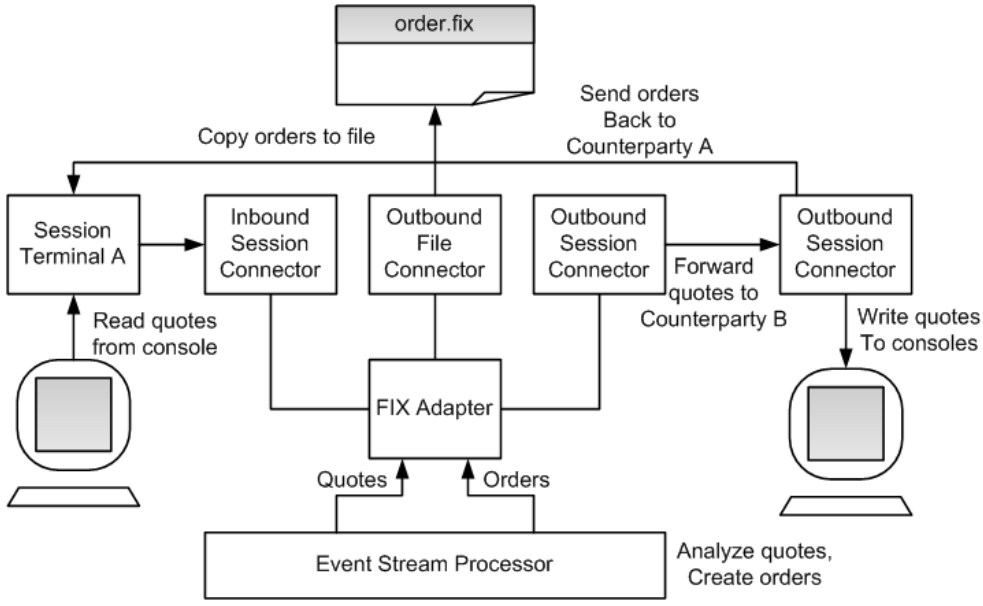
See also

- *Server Socket Connectors* on page 185

Example: Using All-In-One

Engage in a two-way FIX session with counterparty A, and in a one-way FIX session with counterparty B. If the value of the OfferPx field is less than 30.0, the FIX adapter sends a NewOrderSingle message back to counterparty A and copies it to a file.

Figure 6: All-In-One



1. Set the username and password in the example environment:

Operating System	Step
UNIX	1. Edit the `set_example_env.sh` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	1. Edit the `set_example_env.bat` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

2. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

3. Start the respective subscriber utility for Event Stream Processor.

Operating System	Step
UNIX	Open a terminal window and enter: `./esp-subscribe.sh`
Windows	Open a command window and enter: `esp-subscribe.bat`

4. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: `start_adapter.sh`
Windows	Open a command window and enter: `start_adapter.bat`

5. Wait five to ten seconds for the adapter to initialize.
6. Start output terminal B.

Operating System	Step
UNIX	Open a terminal window and enter: `./terminalB.sh`
Windows	Open a command window and enter: `terminalB.bat`

7. Wait five to ten seconds for the adapter to initialize.
8. Start output terminal A.

Operating System	Step
UNIX	Open a terminal window and enter: `./terminalA.sh`
Windows	Open a command window and enter: `terminalA.bat`

9. Wait five to ten seconds for the adapter to initialize.
10. Copy the contents of the `quotes.fix` file and paste it into the terminal A console window.

- The MyQuotes stream now contains two records. The MyOrders stream contains one record. Use the Event Stream Processor subscriber utility to view the content of the streams.
- The terminal B console window now contains a NewOrderSingle message.
- The `orders.fix` file now contains a copy of the NewOrderSingle message.

See also

- *Inbound and Outbound Connectors* on page 178

Flex Output Adapter

The SAP Sybase Event Stream Processor Flex Output adapter is a software interface that allows you to obtain data from streams in an Event Stream Processor project and provide it to a full range of Adobe Flex applications.

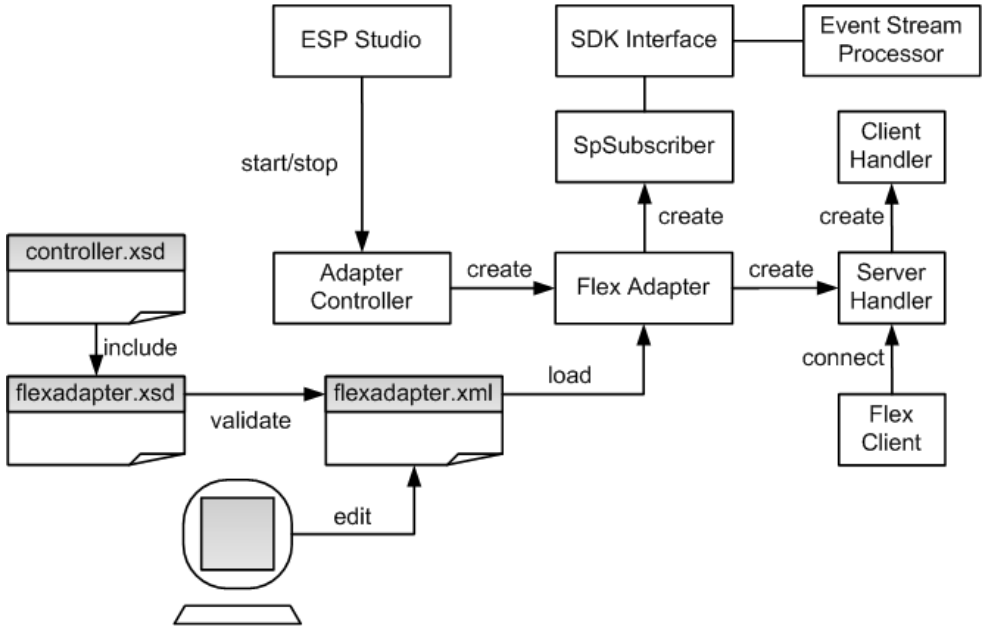
The Flex adapter:

- Runs an internal Flex server, listens, and accepts client connections
- Parses client requests and subscribes to streams
- Filters stream records, converts them into XML, and sends them to clients, one record at a time

Control Flow

The Flex adapter loads its configuration from a file (for example, `adapter-pubsub.xml`), and validates it against the adapter schema (`flexadapter.xsd`), which includes the API-wide controller schema (`controller.xsd`).

Figure 7: Flex Adapter Control Flow



The Adapter Controller creates an instance of the adapter, and receives and executes **start**, **stop**, and **status** commands.

Start Command

The **start** command configures and starts the control interface, gets the server handler to start listening for client connections, creates a separate client handler to serve each client connection, and connects the SpSubscriber component to Event Stream Processor via the SDK interface.

The adapter ignores the **start** command if it is executed when there is a running instance of the adapter, and sends a warning.

See also

- *Starting the Flex Adapter* on page 220

Stop Command

The **stop** command disconnects the SpSubscriber from Event Stream Processor, stops the Server Handler from accepting new client connections, causes the Client Handlers to finalize the responses to existing clients and disconnects them, and terminates the adapter process.

If the **stop** command is executed when there is no instance of a running adapter, the command is ignored and a warning is sent.

See also

- *Stopping the Flex Adapter* on page 221

Status Command

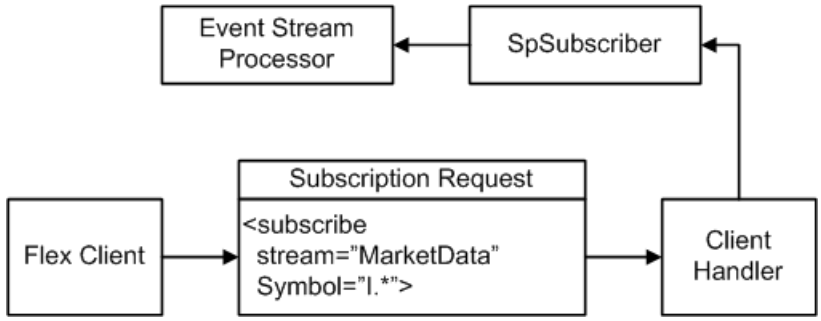
The **status** command reports the adapter status, and the Adapter Controller prints its status: running or stopped.

See also

- *Checking the Flex Adapter Status* on page 221

Message Flow

The message flow between the adapter and any Flex client is initiated when the client sends a subscribe request indicating the stream name and a column filter (optional).



Ensure the request uses this format:

```
<subscribe stream="MyStream" myFilterColumn1="MyRegex1" .../>
```

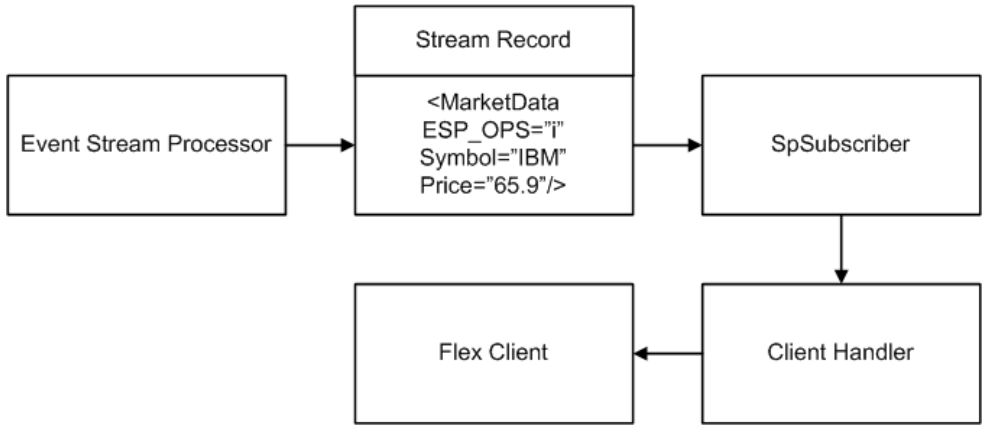
Ensure the filtered columns are `string` datatypes. Regular expressions are accepted as column values. For example, as a result of the following request, the client receives all records on the MarketData stream in which the Symbol begins with the capital letter “I”:

```
<subscribe stream="MarketData" Symbol="I.*"/>
```

The records returned to the clients have this format:

```
<MyStream ESP_OPS="i" myColumn1="MyValue1" myColumn2="MyValue2" .../
>
```

where ESP_OPS is the record opcode. The valid opcode values are "i" (INSERT), "u" (UPDATE), "d" (DELETE), and "p" (UPSERT). All columns with non-null values are included, regardless of the opcode. Null column values are ignored.



To unsubscribe from a previously subscribed stream, the client sends a request using this format:

```
<unsubscribe stream="MyStream"/>
```

The client can subscribe concurrently to any number of different streams. To change the column filter values, unsubscribe from the stream first, then subscribe with the new filter.

In the event of an Event Stream Processor failover, the SDK API switches to the spare Event Stream Processor instance without message loss.

Stream Handler

Use a stream handler for client-server communication.

Although Flex clients can use raw XML to subscribe to and receive stream data from Event Stream Processor, SAP recommends that you delegate the client-server communication to a Stream Handler. To use the Stream Handler functionality, include the `streamhandler.swc` library, located in the `lib` directory, in your Flex client build.

Here is an example of using the Stream Handler in the ActionScript code:

```
    import com.sybase.esp.adapter.flex.StreamHandler;
private var streamHandler:StreamHandler = new StreamHandler(
"localhost", 23456, onConnect, onDisconnect, onRecord);
private function onConnect(event:Event):void {
// Invoked after the client has successfully connected to
// the adapter
}
private function onDisconnect(event:Event):void {
// Invoked after the client has disconnected from
// the adapter
}
```

```
private function onRecord(streamName:String, opcode:String,
record:Object):void {
// Invoked when the client receives a record from
// the adapter
}
```

To subscribe to a stream, invoke the `subscribe()` method of the Stream Handler with the stream name and filter parameters. The filter is an ActionScript object with several properties, one for each filtered column. You can use regular expressions as property values. For example:

```
var filter:Object = new Object();
filter.Symbol = "I.*";
streamHandler.subscribe("Stream1", filter);
```

Ensure that filtered columns are string type. To receive all stream records, without filtering, do not add any properties to the filter object. To unsubscribe from a stream, invoke the `unsubscribe()` method of the Stream Handler, passing in the stream name as a parameter.

For example:

```
streamHandler.subscribe("Stream1");
```

Implement the `onRecord()` callback method to process the records coming on a subscription. The callback has three parameters:

- **streamName**
- **Opcode**
- An object which contains all non-null column values as properties

For example:

```
private function onRecord(streamName:String, opcode:String,
record:Object):void {
trace("Record received on stream " + streamName);
trace("Opcode=" + opcode);
trace("Column values:");
for (var column:String in record)
trace(column + "=" + record[column];
```

Setting the JAVA_HOME Environment Variable

Set the JAVA_HOME environment variable to point to the Java directory.

Prerequisites

- Install Java Runtime Environment version 1.7.0_1 or higher. To see if you have a suitable version of Java installed, go to *http://www.java.com/en/download/installed.jsp*.
- To download and install Java, go to *http://jdl.sun.com/webapps/getjava/BrowserRedirect?locale=en"&"host=www.java.com:80*.

Task
Set the JAVA_HOME environment variable to the directory path where Java Runtime Environment 1.7.0_1 or higher is installed.

Next
Verify that the ESP_HOME environment variable is set correctly.

Configuration
Configuration information for the Flex adapter.

Flex Adapter Directory
The adapter directory contains all files, such as configuration files, templates, examples, and JAR files, relating to the adapter.

```
README.txt  Quick Guide
ReleaseNotes.txt  Release Notes

bin/
    start_adapter.bat  Standalone adapter startup script
    start_adapter.sh      Standalone adapter startup script
    adapter-plugin.bat Plug-in connector startup script
    adapter-plugin.sh  Plug-in connector startup script

config/
    controller.xsd     Controller schema
    log4j.properties   Sample logging configuration
    flexadapter.xsd    Flex Adapter schema
    login.config  Authentication configuration


examples/        Working example

lib/

libj/

    commons-codec-1.3.jar Required by SDK API
    commons-collections-3.2.1.jar
    commons-configuration-1.6.jar
    commons-lang-2.6.jar
    commons-logging-1.1.jar Logging library
    esp_adapter_api.jar Adapter API code
    esp_adapter_flex.jar  Flex adapter library
    esp_i18n.jar
    esp_license.jar
    esp_sdk.jar ESP SDK library
    log4j-1.2.16.jar Logging library
    postgresql.jar
    sylapi.jar
    ws-commons-util-1.0.2.jar Required by ESP SDK
    xerces-impl-2.9.1.jar XML parser library
    xmlrpc-client-3.1.3.jar Required by ESP SDK
    xmlrpc-common-3.1.3.jar Required by ESP SDK
```

Schema and Configuration File

The adapter configuration loads from a file and validates against the adapter schema.

The example folder contains the `adapter.xml` sample adapter configuration file.

You must provide a valid configuration file and ensure that the adapter configuration validates against the adapter schema.

Adapter Controller Parameter

The Adapter Controller parameter specifies the adapter command and control port.

This parameter is defined in the `controller.xsd` file in the `config` directory.

Parameter Name	Description
controllerPort	Type: `positive intger` (Optional) Specifies the adapter command and control port. User commands are sent to this port on localhost.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the Flex adapter.

These parameters are defined in the `controller.xsd` file in the `config` directory.

Parameter Name	Description
espAuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **espAuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
espUser	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **espAuthType**). No default value.

Parameter Name	Description
espPassword	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **espPassword** value is encrypted. Default value is false. If set to true, the password value is decrypted using **espRSAKeyStore** and **espRSAKeyStorePassword**.
espProjectUri	Type: `string` (Required) Specifies the total project URI to connect to Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
pulseInterval	Type: `non-negative integer` (Optional) Specifies the time interval, in seconds, during which outbound record changes are coalesced by Event Stream Processor, then received by the adapter as a single event. If not set or set to 0, record changes are received individually as they occur.
espHeartbeatPeriod	Type: `positive integer` (Optional) Specifies the length of time, in seconds, that adapter waits before sending the next heartbeat to Event Stream Processor. If Event Stream Processor fails to receive two consecutive heartbeats, all records published by the adapter are marked stale. The default value is 10.
recordQueueCapacity	Type: `positive integer` (Optional) Specifies capacity of the record queues. Default value is 4096.

Parameter Name	Description
maxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.
maxPubPoolTime	Type: `positive integer` (Optional) Specifies the maximum period of time, in milliseconds, for which records are pooled before being published. If not set, pooling time is unlimited and the pooling strategy is governed by **maxPubPoolSize**. No default value.
useTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.
espRSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **espAuthType** is set to server_rsa, or the encrypted attribute for **espPassword** is set to true, or both.
espRSAKeyStorePassword	Type: `string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **espAuthType** is set to server_rsa, or the encrypted attribute for **espPassword** is set to true, or both.
espKerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **espAuthType** is set to kerberos.
espKerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **espAuthType** is set to kerberos.

Parameter Name	Description
espKerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **espAuthType** is set to kerberos.
espKerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **espAuthType** is set to kerberos.
espEncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **espPassword** is set to true. If left blank, RSA is used as default.

Flex Server Settings

The **serverPort** parameter, which specifies the port on which the adapter runs its Flex server, is defined in the `flexadapter.xsd` file in the `config` directory.

Parameter Name	Description
serverPort	Type: `int` (Required) Specifies the port on which the adapter runs its Flex server.

Sample Flex Configuration File

Sample configuration file (`adapter.xml`) for the Flex adapter.

This file is in the `example` folder.

```
<adapter>

<!-- Adapter Controller -->
 <controller>
  <controllerPort>13579</controllerPort>
 </controller>

<!-- Sybase ESP Server settings -->
 <esp>
   <espConnection>
        <espProjectUri>esp://localhost:19011/w1/p1</espProjectUri>
   </espConnection>

   <espSecurity>
        <espUser>espuser</espUser>
        <espPassword encrypted="false">espuser</espPassword>
```

```
        <espAuthType>none</espAuthType>
<!--  <espRSAKeyStore>/keystore/keystore.jks</espRSAKeyStore>
       <espRSAKeyStorePassword>Sybase123</espRSAKeyStorePassword> --
>
        <espEncryptionAlgorithm>RSA</espEncryptionAlgorithm>
   </espSecurity>
  <maxPubPoolSize>1</maxPubPoolSize>
 </esp>

<!-- Flex specific -->
 <flex>
  <serverPort>23456</serverPort>
 </flex>

</adapter>
```

Logging

The Flex Output adapter uses the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing the logging configuration is part of the Flex Output adapter distribution.

You can modify the logging levels of the `log4j.properties` configuration file which is located in the `%ESP_HOME%\adapters\<adapter-name>\config` directory. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Operation

Start, stop, or check adapter status from the command line.

Starting the Flex Adapter

To start the Flex adapter from the command line, start Event Stream Processor and execute the **start** command.

Prerequisites

Ensure that the port on which the adapter is listening for client connections is open for TCP connections from the machines where the Flex clients are to be run.

Task

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

   ```
   cd %ESP_HOME%\cluster\nodes\node1
     %ESP_HOME%\bin\esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   %ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

   UNIX:

   a. Start the example cluster.

   ```
   cd $ESP_HOME/cluster/nodes/node1
     $ESP_HOME/bin/esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   $ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   $ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   $ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

2. Start the adapter.

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/flex/bin`
	`./adapter.sh <configuration file path> start`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/flex/bin`
	`adapter.bat <configuration file path> start`

See also

- *Start Command* on page 209

Checking the Flex Adapter Status

To check the Flex adapter status from the command line, execute the **status** command.

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/flex/bin`
	`./adapter.sh <configuration file path> status`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/flex/bin`
	`adapter.bat <configuration file path> status`

You see the adapter status: running or stopped.

See also

- *Status Command* on page 210

Stopping the Flex Adapter

To stop the Flex adapter from the command line, execute the **stop** command.

Prerequisites

When you are running the adapter from the command line, stop the adapter first before stopping the project.

Task

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/flex/bin`
	`./adapter.sh <configuration file path> stop`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/flex/bin`
	`adapter.bat <configuration file path> stop`

See also

- *Stop Command* on page 210

Example: Sending a Subscription Request

Send a subscription request to the adapter, and see a stream record from Event Stream Processor in a Web browser.

Prerequisites

- Install a Web server (port default is 80), Flash policy server (default is 843), and a Web browser with the Flash plug-in.
- SAP recommends that the Web server and the policy server run on the same machine on which the adapter is installed.
- If the Web server and policy server are running on a different machine than the one on which the adapter is installed, ensure the ports listed above are open for TCP connections from the machine where the Web browser is running.
- Copy the `example.swf` file from the adapter `example` directory to the content area of the Web server.
- The Flex Server port default is 23456.
- The Web browser can be used on the same machine or on a different machine.

Task

1. Set the username and password for the adapter:
   a) Edit `adapter.xml`.
   b) In the <User> and <Password> elements, enter `sybase`.
2. Set the username and password in the example environment:

Operating System	Step
UNIX	1. Edit the `set_example_env.sh` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	1. Edit the `set_example_env.bat` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

3. Edit the `start_adapter.sh` script.
4. Set the JAVA_HOME environment variable to the directory where the Java Runtime Environment (JRE) is installed.
5. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

6. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: `start_adapter.sh`
Windows	Open a command window and enter: `start_adapter.bat`

7. Wait five to ten seconds for the adapter to initialize.
8. Upload a stream record.

Operating System	Step
UNIX	Open a terminal window and enter: `./upload.sh`
Windows	Open a command window and enter: `upload.bat`

9. Point the Web browser to the `example.swf` file. For example:

```
http://localhost:80/sybase/example.swf
```

10. You see this stream record in the browser window:

```
Stream = Stream1
Opcode = i
Symbol = IBM
Price = 12.50
```

FTP CSV Input and Output Adapter

The FTP CSV Input adapter obtains CSV data from an FTP server and publishes it to Event Stream Processor. The FTP CSV Output adapter takes data from Event Stream Processor, formats it to CSV format, and saves it to a file on an FTP server.

FTP CSV Input Adapter Configuration

Configure the FTP CSV Input adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

Transporter Module: FTP Input Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **FTPInputTransporterParameters** element.
FTPInputTransporterParameters	(Required) Section containing parameters for the FTP Input transporter.
Host	Type: `string` (Required) Specify the server name or IP address of the FTP server to which you are connecting.
Port	Type: `integer` (Required) Specify the port address for the FTP server to which you are connecting. The default value is 21.
LoginType	Type: `enum` (Required) Specify the login type for the FTP server. There are two valid types: normal and anonymous.

Parameter	Description
User	Type: `string` (Required if **LoginType** is set to normal) Specify the login account for the FTP server.
Password	Type: `string` (Required if **LoginType** is set to normal) Specify the login password for the FTP server.
FtpFilePath	Type: `string` (Required) Specify the absolute path to the data files in the FTP server.
FtpFileName	Type: `string` (Required) Specify the filename of the data files in the FTP server.
MaxBlockSize	Type: `int` (Required) Specify the max data block size to transfer from the FTP server. The default value is 2048.
TransferMode	Type: `string` (Required) Specify the transfer mode for the FTP connection. There are two valid values: active or passive. The default value is active.
RetryPeriod	Type: `second` (Required) Specify the period of time, in seconds, to try and reconnect to the FTP server if you disconnect unexpectedly. The default value is 30.
RetryCount	Type: `integer` (Required) Specify the retry counts to try to reconnect to the FTP server if you disconnect unexpectedly. The default value is 0.

Formatter Module: Stream to String Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **StreamToStringFormatterParameters** parameter.
StreamToStringFormatterParameters	(Required) Section containing the Stream to String formatter parameters.
Delimiter	Type: `string` (Required) Specify the symbol used to separate columns. The default value is "\n".
IncludeDelimiter	Type: `boolean` (Required) If set to true, the delimiter is part of current row. If set to false, the delimiter is not part of the current row. The default value is false.
AppendString	Type: `string` (Required if **IncludeDelimiter** is set to true) If set to true, specify the string to append to the end of each result row. No default value.

Parameter	Description
AppendPosition	Type: `string` (Required if **IncludeDelimiter** is set to true) Specify the position to which the **AppendString** parameter takes effect. There are two valid values: front and end. The default value is front.
IgnoreSpace	Type: `boolean` (Required) Specify whether to trim the space char. The default value is true.

Formatter Module: CSV String to ESP Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **CsvStringToEspFormatterParameters** parameter.
CsvStringToEspFormatterParameters	(Required) Section containing the CSV String to ESP formatter parameters.

Parameter	Description
ExpectStreamNameOpcode	Type: `boolean` (Required) If set to true, the adapter interprets the first two fields of the incoming CSV line as stream name and opcode respectively. The adapter discards messages with unmatched stream names. The accepted opcodes are: • i or I: INSERT • d or D: DELETE • u or U: UPDATE • p or P: UPSERT • s or S: SAFEDELETE The default value is false.
Delimiter	Type: `string` (Advanced) The symbols used to separate the column. The default value is a comma (,).
HasHeader	Type: `boolean` (Advanced) Determines whether the first line of the file contains the description of the fields. Default value is false.
DateFormat	Type: `string` (Advanced) The format string for parsing date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.
TimestampFormat	Type: `string` (Advanced) Format string for parsing timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

ESP Connector Module: ESP Publisher

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **EspPublisherParameters** parameter.
EspPublisherParameters	(Required) Section containing parameters for the ESP publisher.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.

Parameter	Description
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream to which the adapter publishes data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
MaxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.
UseTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.
SafeOps	Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT, and converts DELETE to SAFEDELETE. The default value is false.
SkipDels	Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. The default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the FTP CSV Input adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.

Parameter	Description
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using key-store • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

FTP CSV Input Adapter Studio Properties

Adapter type: toolkit_ftp_csv_input. Set these properties for the FTP CSV Input adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Column Delimiter	Property ID: **csvDelimiter** Type: `string` (Advanced) Specify the symbol used to separate the columns.
Date Format	Propterty ID: **csvDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **csvTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.

Property Label	Description
Has Header	Property ID: **csvHasHeader** Type: `boolean` (Advanced) Indicate whether the first line of the file contains the description of the CSV fields.
Steam name, opcode expected	Property ID: **csvExpectStreamNameOpcode** Type: `boolean` (Optional) If set to true, the adapter interprets the first two fields of the incoming CSV line as stream name and opcode. The adapter discards messages with unmatched values.
Charset Name	Property ID: **charsetName** Type: `string` (Advanced) Specify the name of a supported charest.
Host	Property ID: **host** Type: `string` (Required) Specify the server name or IP address of the FTP server to which you are connecting.
Port	Property ID: **port** Type: `int` (Required) Specify the port for the FTP server to which you are connecting.
Login Type	Property ID: **loginType** Type: `string` (Required) Specify the login type for the FTP server. Valid values: normal, anonymous.
User	Property ID: **user** Type: `string` (Optional) Specify the login account for the FTP server (required if **loginType** is set to normal).

Property Label	Description
Password	Property ID: **password** Type: `string` (Optional) Specify the login password for the FTP server (required if **loginType** is set to normal).
File Name	Property ID: **ftpFileName** Type: `string` (Required) Specify the file name of the data file on the FTP server.
File Path	Property ID: **ftpFilePath** Type: `string` (Required) Specify the absolute path to the data file on the FTP server.
Transfer Mode	Property ID: **transferMode** Type: `string` (Optional) Specify the transfer mode for the FTP connection. Valid values: active, passive.
Buffer Size	Property ID: **inputBufferSize** Type: `uint` (Advanced) Specify the buffer size of the socket connection in bytes.
Retry Period	Property ID: **retryPeriod** Type: `uint` (Optional) Specify the period of time, in seconds, to try and reconnect to the FTP server if you disconnect unexpectedly.
Retry Number	Property ID: **retryNumber** Type: `uint` (Optional) Specify the number of times to try and reconnect to the FTP server if you disconnect unexpectedly.

Property Label	Description
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: FTP CSV Input Adapter

Sample adapter configuration file for the FTP CSV Input adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
  <Name>ftp_csv_input</Name>
  <Description>An adapter which gets csv data from ftp server,
transforms to ESP data format, and publishes to ESP stream.</
Description>
  <Log4jProperty>./log4j.properties</Log4jProperty>
  <Modules>
    <Module type="transporter">
      <InstanceName>MyFtpInTransporter</InstanceName>
      <Name>FtpInputTransporter</Name>
      <Next>MyStreamingInputFormatter</Next>
      <Parameters>
        <FtpInputTransporterParameters>
          <Host>10.128.108.103</Host>
          <Port>21</Port>
          <User>anonymous</User>
          <Password>anonymous</Password>
          <LoginType>normal</LoginType>
          <FtpFilePath>/aaa</FtpFilePath>
          <FtpFileName>input.csv</FtpFileName>
          <MaxBlockSize>10240</MaxBlockSize>
          <TransferMode>active</TransferMode>
          <RetryPeriod>3000</RetryPeriod>
          <RetryNumber>0</RetryNumber>
       </FtpInputTransporterParameters>
      </Parameters>
    </Module>

     <Module type="formatter">
      <InstanceName>MyStreamingInputFormatter</InstanceName>
      <Name>StreamToStringFormatter</Name>
```

```
      <Next>CsvStringToEspFormatter</Next>
      <Parameters>
        <StreamToStringFormatterParameters>
          <Delimiter><![CDATA[\n]]></Delimiter>
          <IncludeDelimiter>true</IncludeDelimiter>
          <AppendString><![CDATA[\n]]></AppendString>
          <AppendPosition>front</AppendPosition>
          <IgnoreSpace>true</IgnoreSpace>
       </StreamToStringFormatterParameters>
      </Parameters>
    </Module>

    <Module type="formatter">
      <InstanceName>CsvStringToEspFormatter</InstanceName>
      <Name>CsvStringToEspFormatter</Name>
      <Next>MyInStream_Publisher</Next>
      <Parallel>true</Parallel>
      <Parameters>
           <CsvStringToEspFormatterParameters>
               <ExpectStreamNameOpcode>true</ExpectStreamNameOpcode>
           </CsvStringToEspFormatterParameters>
      </Parameters>
    </Module>

    <Module type="espconnector">
      <InstanceName>MyInStream_Publisher</InstanceName>
      <Name>EspPublisher</Name>
      <Parameters>
        <EspPublisherParameters>
          <ProjectName>EspProject1</ProjectName>
          <StreamName>BaseInput</StreamName>
          <MaxPubPoolSize>1</MaxPubPoolSize>
          <UseTransactions>false</UseTransactions>
          <SafeOps>true</SafeOps>
          <SkipDels>true</SkipDels>
        </EspPublisherParameters>
      </Parameters>
      <BufferMaxSize>10240</BufferMaxSize>
    </Module>

  </Modules>

  <EspProjects>
    <EspProject>
      <Name>EspProject1</Name>
      <Uri>esp://localhost:19011/sample_workspace/ftp_csv_input</
Uri>
      <Security>
        <User></User>
        <Password encrypted="false"></Password>
        <AuthType>user_password</AuthType>
      </Security>
    </EspProject>
  </EspProjects>
  <GlobalParameters></GlobalParameters>
</Adapter>
```

FTP CSV Output Adapter Configuration

Configure the FTP CSV Output adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

ESPConnector Module: ESP Subscriber

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, espconnector.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **EspSubscriberParameters** parameter.
EspSubscriberParameters	(Required) Section containing parameters for the ESP subscriber.

Parameter	Description
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Specifies the unique project tag of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream from which the adapter subscribes to data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.

Formatter Module: ESP to CSV String Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **EspToCsvStringFormatterParameters** parameter.
EspToCsvStringFormatterParameters	(Required) Section containing parameters for the ESP to CSV String formatter parameters.
PrependStreamNameOpcode	Type: `boolean` (Optional) If set to true, the adapter prepends the stream name and the opcode in each row of data that is generated. The default value is false.
Delimiter	Type: `string` (Advanced) The symbols used to separate the column. The default value is a comma (,).
HasHeader	Type: `boolean` (Advanced) Determines whether the first line of the file contains the description of the fields. The default value is false.
DateFormat	Type: `string` (Advanced) The format string for date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.
TimestampFormat	Type: `string` (Advanced) Format string for timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

Formatter Module: String to Stream Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **StringToStreamFormatterParameters** parameter.
StringToStreamFormatterParameters	(Required) Section containing parameters for the String to Stream formatter parameters.
Delimiter	Type: `string` (Required) Specify the symbol used to separate columns. The default value is "\n".
IncludeDelimiter	Type: `boolean` (Required) If set to true, the delimiter is part of current row. If set to false, the delimiter is not part of the current row. The default value is false.
AppendString	Type: `string` (Required if **IncludeDelimiter** is set to true) If set to true, specify the string to append to the end of each result row. No default value.

Parameter	Description
AppendPosition	Type: `string` (Required if **IncludeDelimiter** is set to true) Specify the position to which the **AppendString** parameter takes effect. There are two valid values: front and end. The default value is front.
IgnoreSpace	Type: `boolean` (Required) Specify whether to ignore the space char. The default value is false.
CharsetName	Type: `string` (Advanced) Specify the name of a supported charset. The default value is US-ASCII.

Transporter Module: FTP Output Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **FTPOutputTraansporterParameters** parameter.
FTPOutputTraansporterParameters	(Required) Section containing the parameters for the FTP Output transporter.
Host	Type: `string` (Required) Specify the server name or IP address of the FTP server to which you are connecting.

Parameter	Description
Port	Type: `integer` (Required) Specify the port address for the FTP server to which you are connecting. The default value is 21.
LoginType	Type: `enum` (Required) Specify the login type for the FTP server. There are two valid types: normal and anonymous.
User	Type: `string` (Required if **LoginType** is set to normal) Specify the login account for the FTP server.
Password	Type: `string` (Required if **LoginType** is set to normal) Specify the login password for the FTP server.
FtpFilePath	Type: `string` (Required) Specify the absolute path to the data files in the FTP server.
FtpFileName	Type: `string` (Required) Specify the filename of the data files in the FTP server.
MaxBlockSize	Type: `int` (Required) Specify the max data block size to transfer to the FTP server. The default value is 2048.
Overwrite	Type: `boolean` (Required) If set to true, the transporter overwrites the file on the FTP server, if it exists. If this parameter is set to false, the transporter appends the output to the end of the existing file. The default value is false.

Parameter	Description
TransferMode	Type: `string` (Required) Specify the transfer mode for the FTP connection. There are two valid values: active or passive. The default value is active.
RetryPeriod	Type: `second` (Required) Specify the period of time, in seconds, to try and reconnect to the FTP server if you disconnect unexpectedly. The default value is 30.
RetryCount	Type: `integer` (Required) Specify the retry counts to try to reconnect to the FTP server if you disconnect unexpectedly. The default value is 0.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the FTP CSV Output adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.

Parameter	Description
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.

Parameter	Description
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

FTP CSV Output Adapter Studio Properties

Adapter type: toolkit_ftp_csv_output. Set these properties for the FTP CSV Output adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Column Delimiter	Property ID: **csvDelimiter** Type: `string` (Advanced) Specify the symbol used to separate the columns.

Property Label	Description
Date Format	Propterty ID: **csvDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **csvTimestampFormat** Type: `string` (Advanced) Specify the format for parsing time-stamp values.
Has Header	Property ID: **csvHasHeader** Type: `boolean` (Advanced) Indicate whether the first line of the file contains the description of the CSV fields.
Prepend stream name, opcode	Property ID: **csvPrependStreamNameOpcode** Type: `boolean` (Optional) If set to true, the adapter prepends the stream name and the opcode in each row of data that is generated.
Charset Name	Property ID: **charsetName** Type: `string` (Advanced) Specify the name of a supported charest.
Host	Property ID: **host** Type: `string` (Required) Specify the server name or IP address of the FTP server to which you are connecting.
Port	Property ID: **port** Type: `int` (Required) Specify the port for the FTP server to which you are connecting.

Property Label	Description
Login Type	Property ID: **loginType** Type: `string` (Required) Specify the login type for the FTP server. Valid values: normal, anonymous.
User	Property ID: **user** Type: `string` (Optional) Specify the login account for the FTP server (required if **loginType** is set to normal).
Password	Property ID: **password** Type: `string` (Optional) Specify the login password for the FTP server (required if **loginType** is set to normal).
File Name	Property ID: **ftpFileName** Type: `string` (Required) Specify the file name of the data file on the FTP server.
File Path	Property ID: **ftpFilePath** Type: `string` (Required) Specify the absolute path to the data file on the FTP server.
Transfer Mode	Property ID: **transferMode** Type: `string` (Optional) Specify the transfer mode for the FTP connection. Valid values: active, passive.
Buffer Size	Property ID: **inputBufferSize** Type: `uint` (Advanced) Specify the buffer size of the socket connection in bytes.

Property Label	Description
Over Write	Property ID: **overwrite** Type: `uint` (Optional) If set to true, the transporter overwrites the file on the FTP server, if it exists. If set to false, the transporter appends the output to the end of the existing file.
Retry Number	Property ID: **retryNumber** Type: `uint` (Optional) Specify the number of times to try and reconnect to the FTP server if you disconnect unexpectedly.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: FTP CSV Output Adapter

Sample adapter configuration file for the FTP CSV Output adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>ftp_csv_output</Name>
   <Description>An adapter which transforms ESP data to csv format,
and save to file to ftp server</Description>
    <Log4jProperty>./log4j.properties</Log4jProperty>
    <Modules>
        <Module type="espconnector">
            <InstanceName>MyOutStream_Subscriber</InstanceName>
            <Name>EspSubscriber</Name>
            <Next>EspToCsvStringFormatter</Next>
            <Parameters>
                <EspSubscriberParameters>
                    <ProjectName>EspProject2</ProjectName>
```

```
                    <StreamName>BaseOutput</StreamName>
                </EspSubscriberParameters>
            </Parameters>
        </Module>

        <Module type="formatter">
            <InstanceName>EspToCsvStringFormatter</InstanceName>
            <Name>EspToCsvStringFormatter</Name>
            <Next>StreamingOutputFormatter</Next>
            <Parallel>true</Parallel>
            <Parameters>
                <EspToCsvStringFormatterParameters>
                    <PrependStreamNameOpcode>true</
PrependStreamNameOpcode>
                </EspToCsvStringFormatterParameters>
            </Parameters>
        </Module>

        <Module type="formatter">
          <InstanceName>StreamingOutputFormatter</InstanceName>
          <Name>StringToStreamFormatter</Name>
          <Next>FtpOutTransporter</Next>
          <Parameters>
            <StringToStreamFormatterParameters>
              <Delimiter>\n</Delimiter>
              <IncludeDelimiter>true</IncludeDelimiter>
              <AppendString>\n</AppendString>
              <AppendPosition>end</AppendPosition>
             <IgnoreSpace>true</IgnoreSpace>
           </StringToStreamFormatterParameters>
          </Parameters>
        </Module>
        <Module type="transporter">
          <InstanceName>FtpOutTransporter</InstanceName>
          <Name>FtpOutputTransporter</Name>
          <Parameters>
            <FtpOutputTransporterParameters>
              <Host>10.128.108.103</Host>
              <Port>21</Port>
              <User>anonymous</User>
              <Password>anonymous</Password>
              <LoginType>normal</LoginType>
              <FtpFilePath>/ccc</FtpFilePath>
              <FtpFileName>output.csv</FtpFileName>
              <MaxBlockSize>2048</MaxBlockSize>
              <TransferMode>active</TransferMode>
              <Overwrite>true</Overwrite>
              <RetryPeriod>3000</RetryPeriod>
              <RetryNumber>0</RetryNumber>
           </FtpOutputTransporterParameters>
          </Parameters>
        </Module>
    </Modules>

    <EspProjects>
        <EspProject>
```

```
            <Name>EspProject2</Name>
            <Uri>esp://localhost:19011/sample_workspace/
ftp_csv_output</Uri>
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Adapter Controller Parameters

The Adapter Controller port listens for commands. The `controller.xml` file is located in `%ESP_HOME%/adapters/framework/config` directory. This file is shared among all the adapter within the `%ESP_HOME%/adapters/framework/instances` directory.

Parameter	Description
ControlPort	(Required) Section containing the **MinPort** and **MaxPort** parameters.
MinPort	Type: `int` (Required) The minimum port number that can act as a control port number. The adapter framework allocates this number automatically. A recommended value is 19082.
MaxPort	Type: `int` (Optional) The maximum port number that can act as a control port number. The adapter framework allocates this number automatically. The default value is 65535.
Statistics_Update_Interval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework publishes statistics data to the ESP project to which the adapter is connected. A recommended value is 5000.

Parameter	Description
MonitorInterval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework checks the status of internal resource allocation. The framework logs a message in the log file if a resource is about to be exhausted. A recommended value is 3000.

Logging

The FTP CSV Input and Output adapter use the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing the logging configuration is located in the `%ESP_HOME%\adapters\framework\config` directory.

Specify the location of the logging file you wish to use in the **Log4jProperty** parameter within the adapter configuration file. You can modify the logging levels within this file or the `$ESP_HOME/adapters/framework/config/log4j.properties`, which is used by default. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Starting the FTP CSV Adapter

To start the FTP CSV adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

a. Start the example cluster.

```
cd %ESP_HOME%\cluster\nodes\node1
  %ESP_HOME%\bin\esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
%ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
%ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
%ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

UNIX:

a. Start the example cluster.

```
cd $ESP_HOME/cluster/nodes/node1
  $ESP_HOME/bin/esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
$ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

2. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: For the FTP CSV Input adapter: `cd $ESP_HOME/adapters/framework/instances/ftp_csv_input` For the FTP CSV Output adapter: `cd $ESP_HOME/adapters/framework/instances/ftp_csv_output` `./start_adapter.sh <adapter configuration file name>`

Operating System	Step
Windows	Open a command window and enter: For the FTP CSV Input adapter: `cd %ESP_HOME%/adapters/framework/instances/ftp_csv_input` For the FTP CSV Output adapter: `cd %ESP_HOME%/adapters/framework/instances/ftp_csv_output` `start_adapter.bat <adapter configuration file name>`

Stopping the FTP CSV Adapter

To stop the FTP CSV adapter from the command line, execute the **stop** command.

Operating System	Step
UNIX	Open a terminal window and enter: For the FTP CSV Input adapter: `cd $ESP_HOME/adapters/framework/instances/ftp_csv_input` For the FTP CSV Output adapter: `cd $ESP_HOME/adapters/framework/instances/ftp_csv_output` `./stop_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the FTP CSV Input adapter: `cd %ESP_HOME%/adapters/framework/instances/ftp_csv_input` For the FTP CSV Output adapter: `cd %ESP_HOME%/adapters/framework/instances/ftp_csv_output` `stop_adapter.bat <adapter configuration file name>`

FTP XML Input and Output Adapter

The FTP XML Input adapter reads data from an XML document on an FTP server into Event Stream Processor. The FTP XML Output adapter reads XML data from an ESP project, writes it to an XML document, and uploads this file to the FTP server.

FTP XML Input Adapter Configuration

Configure the FTP XML Input adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

Transporter Module: FTP Input Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.

Parameter	Description
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **FTPInputTransporterParameters** element.
FTPInputTransporterParameters	(Required) Section containing parameters for the FTP Input transporter.
Host	Type: `string` (Required) Specify the server name or IP address of the FTP server to which you are connecting.
Port	Type: `integer` (Required) Specify the port address for the FTP server to which you are connecting. The default value is 21.
LoginType	Type: `enum` (Required) Specify the login type for the FTP server. There are two valid types: normal and anonymous.
User	Type: `string` (Required if **LoginType** is set to normal) Specify the login account for the FTP server.
Password	Type: `string` (Required if **LoginType** is set to normal) Specify the login password for the FTP server.
FtpFilePath	Type: `string` (Required) Specify the absolute path to the data files in the FTP server.
FtpFileName	Type: `string` (Required) Specify the filename of the data files in the FTP server.

Parameter	Description
MaxBlockSize	Type: `int` (Required) Specify the max data block size to transfer from the FTP server. The default value is 2048.
TransferMode	Type: `string` (Required) Specify the transfer mode for the FTP connection. There are two valid values: active or passive. The default value is active.
RetryPeriod	Type: `second` (Required) Specify the period of time, in seconds, to try and reconnect to the FTP server if you disconnect unexpectedly. The default value is 30.
RetryCount	Type: `integer` (Required) Specify the retry counts to try to reconnect to the FTP server if you disconnect unexpectedly. The default value is 0.

Formatter Module: Stream to String Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.

Parameter	Description
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **StreamToStringFormatterParameters** parameter.
StreamToStringFormatterParameters	(Required) Section containing the Stream to String formatter parameters.
Delimiter	Type: `string` (Required) Specify the symbol used to separate columns. The default value is "\n".
IncludeDelimiter	Type: `boolean` (Required) If set to true, the delimiter is part of current row. If set to false, the delimiter is not part of the current row. The default value is false.
AppendString	Type: `string` (Required if **IncludeDelimiter** is set to true) If set to true, specify the string to append to the end of each result row. No default value.
AppendPosition	Type: `string` (Required if **IncludeDelimiter** is set to true) Specify the position to which the **AppendString** parameter takes effect. There are two valid values: front and end. The default value is front.
IgnoreSpace	Type: `boolean` (Required) Specify whether to trim the space char. The default value is true.

Formatter Module: XML String to ESP Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.

Parameter	Description
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **XmlStringToEspFormatterParameters** parameter.
XmlStringToEspFormatterParameters	(Required) Section containing the XML String to ESP formatter parameters.
DateFormat	Type: `string` (Optional) The format string for parsing date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.
TimestampFormat	Type: `string` (Optional) Format string for parsing timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

ESP Connector Module: ESP Publisher

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **EspPublisherParameters** parameter.
EspPublisherParameters	(Required) Section containing parameters for the ESP publisher.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.

Parameter	Description
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream to which the adapter publishes data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
MaxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.
UseTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.
SafeOps	Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT, and converts DELETE to SAFEDELETE. The default value is false.
SkipDels	Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. The default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the FTP XML Input adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.

Parameter	Description
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using key-store • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

FTP XML Input Adapter Studio Properties

Adapter type: toolkit_ftp_xmllist_input. Set these properties for the FTP XML Input adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Date Format	Propterty ID: **xmllistDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **xmllistTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.
Charset Name	Property ID: **charsetName** Type: `string` (Advanced) Specify the name of a supported charest.

Property Label	Description
Host	Property ID: **host** Type: `string` (Required) Specify the server name or IP address of the FTP server to which you are connecting.
Port	Property ID: **port** Type: `int` (Required) Specify the port for the FTP server to which you are connecting.
Login Type	Property ID: **loginType** Type: `string` (Required) Specify the login type for the FTP server. Valid values: normal, anonymous.
User	Property ID: **user** Type: `string` (Optional) Specify the login account for the FTP server (required if **loginType** is set to normal).
Password	Property ID: **password** Type: `string` (Optional) Specify the login password for the FTP server (required if **loginType** is set to normal).
File Name	Property ID: **ftpFileName** Type: `string` (Required) Specify the file name of the data file on the FTP server.
File Path	Property ID: **ftpFilePath** Type: `string` (Required) Specify the absolute path to the data file on the FTP server.

Property Label	Description
Transfer Mode	Property ID: **transferMode** Type: `string` (Optional) Specify the transfer mode for the FTP connection. Valid values: active, passive.
Buffer Size	Property ID: **inputBufferSize** Type: `uint` (Advanced) Specify the buffer size of the socket connection in bytes.
Retry Period	Property ID: **retryPeriod** Type: `uint` (Optional) Specify the period of time, in seconds, to try and reconnect to the FTP server if you disconnect unexpectedly.
Retry Number	Property ID: **retryNumber** Type: `uint` (Optional) Specify the number of times to try and reconnect to the FTP server if you disconnect unexpectedly.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: FTP XML Input Adapter

Sample adapter configuration file for the FTP XML Input adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
  <Name>ftp_xmllist_input</Name>
  <Description>An adapter which gets xml list data from ftp server,
transforms to ESP data format, and publishes to ESP stream.</
Description>
  <Log4jProperty>./log4j.properties</Log4jProperty>
  <Modules>
    <Module type="transporter">
      <InstanceName>MyFtpInputTransporter</InstanceName>
      <Name>FtpInputTransporter</Name>
      <Next>MyStreamToStringFormatter</Next>
      <Parameters>
        <FtpInputTransporterParameters>
          <Host>10.128.108.103</Host>
          <Port>21</Port>
          <User>anonymous</User>
          <Password>anonymous</Password>
          <LoginType>normal</LoginType>
          <FtpFilePath>/aaa</FtpFilePath>
          <FtpFileName>input.xml</FtpFileName>
          <MaxBlockSize>1024</MaxBlockSize>
          <TransferMode>active</TransferMode>
          <RetryPeriod>3000</RetryPeriod>
          <RetryNumber>0</RetryNumber>
       </FtpInputTransporterParameters>
      </Parameters>
    </Module>

     <Module type="formatter">
      <InstanceName>MyStreamToStringFormatter</InstanceName>
      <Name>StreamToStringFormatter</Name>
      <Next>MyXmlStringToEspFormatter</Next>
      <Parameters>
        <StreamToStringFormatterParameters>
          <Delimiter><![CDATA[<BaseInput]]></Delimiter>
          <IncludeDelimiter>true</IncludeDelimiter>
          <AppendString><![CDATA[<BaseInput]]></AppendString>
          <AppendPosition>front</AppendPosition>
          <IgnoreSpace>true</IgnoreSpace>
       </StreamToStringFormatterParameters>
      </Parameters>
    </Module>

    <Module type="formatter">
      <InstanceName>MyXmlStringToEspFormatter</InstanceName>
      <Name>XmlStringToEspFormatter</Name>
      <Next>MyInStream_Publisher</Next>
      <Parallel>true</Parallel>
      <Parameters>
      </Parameters>
```

```
    </Module>


    <Module type="espconnector">
      <InstanceName>MyInStream_Publisher</InstanceName>
      <Name>EspPublisher</Name>
      <Parameters>
        <EspPublisherParameters>
          <ProjectName>EspProject1</ProjectName>
          <StreamName>BaseInput</StreamName>
          <MaxPubPoolSize>1</MaxPubPoolSize>
          <UseTransactions>false</UseTransactions>
          <SafeOps>true</SafeOps>
          <SkipDels>true</SkipDels>
        </EspPublisherParameters>
      </Parameters>
      <BufferMaxSize>10240</BufferMaxSize>
    </Module>

 </Modules>

  <EspProjects>
    <EspProject>
      <Name>EspProject1</Name>
      <Uri>esp://localhost:19011/sample_workspace/
ftp_xmllist_input</Uri>
      <Security>
        <User></User>
        <Password encrypted="false"></Password>
        <AuthType>user_password</AuthType>
        <!--
          <RSAKeyStore>/keystore/keystore.jks</RSAKeyStore>
          <RSAKeyStorePassword>Sybase123</RSAKeyStorePassword>
        -->
        <!--
         <KerberosKDC>KDC</KerberosKDC>
         <KerberosRealm>REALM</KerberosRealm>
          <KerberosService>service/instance</KerberosService>
        <KerberosTicketCache>/tmp/krb5cc_user</KerberosTicketCache>
        -->
        <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
      </Security>
    </EspProject>
  </EspProjects>
  <GlobalParameters></GlobalParameters>
</Adapter>
```

FTP XML Output Adapter Configuration

Configure the FTP XML Output adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

ESPConnector Module: ESP Subscriber

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, espconnector.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **EspSubscriberParameters** parameter.
EspSubscriberParameters	(Required) Section containing parameters for the ESP subscriber.

Parameter	Description
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Specifies the unique project tag of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream from which the adapter subscribes to data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.

Formatter Module: ESP to XML String Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **EspToXmlStringFormatterParameters** parameter.
EspToXmlStringFormatterParameters	(Required) Section containing parameters for the ESP to XML String formatter parameters.
DateFormat	Type: `string` (Optional) The format string for date values from an ESP project. For example, `yyyy-MM-dd'T'HH:mm:ss.`
TimestampFormat	Type: `string` (Optional) Format string for timestamp values from an ESP project. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

Formatter Module: String to Stream Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.

Parameter	Description
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **StringToStreamFormatterParameters** parameter.
StringToStreamFormatterParameters	(Required) Section containing parameters for the String to Stream formatter parameters.
Delimiter	Type: `string` (Required) Specify the symbol used to separate columns. The default value is "\n".
IncludeDelimiter	Type: `boolean` (Required) If set to true, the delimiter is part of current row. If set to false, the delimiter is not part of the current row. The default value is false.
AppendString	Type: `string` (Required if **IncludeDelimiter** is set to true) If set to true, specify the string to append to the end of each result row. No default value.
AppendPosition	Type: `string` (Required if **IncludeDelimiter** is set to true) Specify the position to which the **AppendString** parameter takes effect. There are two valid values: front and end. The default value is front.

Parameter	Description
IgnoreSpace	Type: `boolean` (Required) Specify whether to ignore the space char. The default value is false.
CharsetName	Type: `string` (Advanced) Specify the name of a supported charset. The default value is US-ASCII.

Transporter Module: FTP Output Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **FTPOutputTraansporterParameters** parameter.
FTPOutputTraansporterParameters	(Required) Section containing the parameters for the FTP Output transporter.
Host	Type: `string` (Required) Specify the server name or IP address of the FTP server to which you are connecting.
Port	Type: `integer` (Required) Specify the port address for the FTP server to which you are connecting. The default value is 21.

Parameter	Description
LoginType	Type: `enum` (Required) Specify the login type for the FTP server. There are two valid types: normal and anonymous.
User	Type: `string` (Required if **LoginType** is set to normal) Specify the login account for the FTP server.
Password	Type: `string` (Required if **LoginType** is set to normal) Specify the login password for the FTP server.
FtpFilePath	Type: `string` (Required) Specify the absolute path to the data files in the FTP server.
FtpFileName	Type: `string` (Required) Specify the filename of the data files in the FTP server.
MaxBlockSize	Type: `int` (Required) Specify the max data block size to transfer to the FTP server. The default value is 2048.
Overwrite	Type: `boolean` (Required) If set to true, the transporter overwrites the file on the FTP server, if it exists. If this parameter is set to false, the transporter appends the output to the end of the existing file. The default value is false.
TransferMode	Type: `string` (Required) Specify the transfer mode for the FTP connection. There are two valid values: active or passive. The default value is active.

Parameter	Description
RetryPeriod	Type: `second` (Required) Specify the period of time, in seconds, to try and reconnect to the FTP server if you disconnect unexpectedly. The default value is 30.
RetryCount	Type: `integer` (Required) Specify the retry counts to try to reconnect to the FTP server if you disconnect unexpectedly. The default value is 0.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the FTP XML Output adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.

Parameter	Description
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

FTP XML Output Adapter Studio Properties

Adapter type: toolkit_ftp_xmllist_output. Set these properties for the FTP XML Output adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Date Format	Propterty ID: **xmllistDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **xmllistTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.

Property Label	Description
Charset Name	Property ID: **charsetName** Type: `string` (Advanced) Specify the name of a supported charest.
Host	Property ID: **host** Type: `string` (Required) Specify the server name or IP address of the FTP server to which you are connecting.
Port	Property ID: **port** Type: `int` (Required) Specify the port for the FTP server to which you are connecting.
Login Type	Property ID: **loginType** Type: `string` (Required) Specify the login type for the FTP server. Valid values: normal, anonymous.
User	Property ID: **user** Type: `string` (Optional) Specify the login account for the FTP server (required if **loginType** is set to normal).
Password	Property ID: **password** Type: `string` (Optional) Specify the login password for the FTP server (required if **loginType** is set to normal).
File Name	Property ID: **ftpFileName** Type: `string` (Required) Specify the file name of the data file on the FTP server.

Property Label	Description
File Path	Property ID: **ftpFilePath** Type: `string` (Required) Specify the absolute path to the data file on the FTP server.
Transfer Mode	Property ID: **transferMode** Type: `string` (Optional) Specify the transfer mode for the FTP connection. Valid values: active, passive.
Over Write	Property ID: **overwrite** Type: `uint` (Optional) If set to true, the transporter overwrites the file on the FTP server, if it exists. If set to false, the transporter appends the output to the end of the existing file.
Retry Period	Property ID: **retryPeriod** Type: `uint` (Optional) Specify the period of time, in seconds, to try and reconnect to the FTP server if you disconnect unexpectedly.
Retry Number	Property ID: **retryNumber** Type: `uint` (Optional) Specify the number of times to try and reconnect to the FTP server if you disconnect unexpectedly.

Property Label	Description
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: FTP XML Output Adapter

Sample adapter configuration file for the FTP XML Output adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
  <Name>ftp_xmllist_output</Name>
  <Description>An adapter which transforms ESP data to xml list
format, and save to file on ftp server.</Description>
  <Log4jProperty>./log4j.properties</Log4jProperty>
  <Modules>
      <Module type="espconnector">
        <InstanceName>MyOutStream_Subscriber</InstanceName>
        <Name>EspSubscriber</Name>
        <Next>MyXmlListOutputFormatter</Next>
        <Parameters>
            <EspSubscriberParameters>
                <ProjectName>EspProject2</ProjectName>
                <StreamName>BaseOutput</StreamName>
            </EspSubscriberParameters>
        </Parameters>
    </Module>

      <Module type="formatter">
        <InstanceName>MyXmlListOutputFormatter</InstanceName>
        <Name>EspToXmlStringFormatter</Name>
        <Next>MyStringToStreamFormatter</Next>
        <Parallel>true</Parallel>
        <Parameters>
        </Parameters>
    </Module>

    <Module type="formatter">
      <InstanceName>MyStringToStreamFormatter</InstanceName>
      <Name>StringToStreamFormatter</Name>
      <Next>MyFtpOutputTransporter</Next>
```

```
      <Parameters>
        <StringToStreamFormatterParameters>
          <Delimiter>\n</Delimiter>
          <IncludeDelimiter>true</IncludeDelimiter>
          <AppendString>\n</AppendString>
          <AppendPosition>end</AppendPosition>
          <IgnoreSpace>true</IgnoreSpace>
          <CharsetName>US-ASCII</CharsetName>
        </StringToStreamFormatterParameters>
      </Parameters>
    </Module>

    <Module type="transporter">
      <InstanceName>MyFtpOutputTransporter</InstanceName>
      <Name>FtpOutputTransporter</Name>
      <Parameters>
        <FtpOutputTransporterParameters>
          <Host>10.128.108.103</Host>
          <Port>21</Port>
          <User>anonymous</User>
          <Password>anonymous</Password>
          <LoginType>normal</LoginType>
          <FtpFilePath>/ccc</FtpFilePath>
          <FtpFileName>ccc.txt</FtpFileName>
          <MaxBlockSize>2048</MaxBlockSize>
          <TransferMode>active</TransferMode>
          <Overwrite>true</Overwrite>
          <RetryPeriod>3000</RetryPeriod>
          <RetryNumber>0</RetryNumber>
        </FtpOutputTransporterParameters>
      </Parameters>
    </Module>
  </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject2</Name>
            <Uri>esp://localhost:19011/sample_workspace/
ftp_xmllist_output</Uri>
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Adapter Controller Parameters

The Adapter Controller port listens for commands. The `controller.xml` file is located in `%ESP_HOME%/adapters/framework/config` directory. This file is shared among all the adapter within the `%ESP_HOME%/adapters/framework/instances` directory.

Parameter	Description
ControlPort	(Required) Section containing the **MinPort** and **MaxPort** parameters.
MinPort	Type: `int` (Required) The minimum port number that can act as a control port number. The adapter framework allocates this number automatically. A recommended value is 19082.
MaxPort	Type: `int` (Optional) The maximum port number that can act as a control port number. The adapter framework allocates this number automatically. The default value is 65535.
Statistics_Update_Interval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework publishes statistics data to the ESP project to which the adapter is connected. A recommended value is 5000.
MonitorInterval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework checks the status of internal resource allocation. The framework logs a message in the log file if a resource is about to be exhausted. A recommended value is 3000.

Logging

The FTP XML Input and Output adapter use the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing

the logging configuration is located in the `%ESP_HOME%\adapters\framework\config` directory.

Specify the location of the logging file you wish to use in the **Log4jProperty** parameter within the adapter configuration file. You can modify the logging levels within this file or the `$ESP_HOME/adapters/framework/config/log4j.properties`, which is used by default. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
```

```
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Starting the FTP XML Adapter

To start the FTP XML adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

   ```
   cd %ESP_HOME%\cluster\nodes\node1
     %ESP_HOME%\bin\esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   %ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

   UNIX:

a. Start the example cluster.

```
cd $ESP_HOME/cluster/nodes/node1
  $ESP_HOME/bin/esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
$ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

2. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: For the FTP XML Input adapter: `cd $ESP_HOME/adapters/framework/instances/ftp_xmllist_input` For the FTP XML Output adapter: `cd $ESP_HOME/adapters/framework/instances/ftp_xmllist_output` `./start_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the FTP XML Input adapter: `cd %ESP_HOME%/adapters/framework/instances/ftp_xmllist_input` For the FTP XML Output adapter: `cd %ESP_HOME%/adapters/framework/instances/ftp_xmllist_output` `start_adapter.bat <adapter configuration file name>`

Stopping the FTP XML Adapter

To stop the FTP XML adapter from the command line, execute the **stop** command.

Operating System	Step
UNIX	Open a terminal window and enter: For the FTP XML Input adapter: `cd $ESP_HOME/adapters/framework/instances/ftp_xmllist_input` For the FTP XML Output adapter: `cd $ESP_HOME/adapters/framework/instances/ftp_xmllist_output` `./stop_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the FTP XML Input adapter: `cd %ESP_HOME%/adapters/framework/instances/ftp_xmllist_input` For the FTP XML Output adapter: `cd %ESP_HOME%/adapters/framework/instances/ftp_xmllist_output` `stop_adapter.bat <adapter configuration file name>`

HTTP Output Adapter

Adapter type: httpplugin. The SAP Sybase Event Stream Processor HTTP adapter publishes data from Event Stream Processor to external clients.

The HTTP adapter:

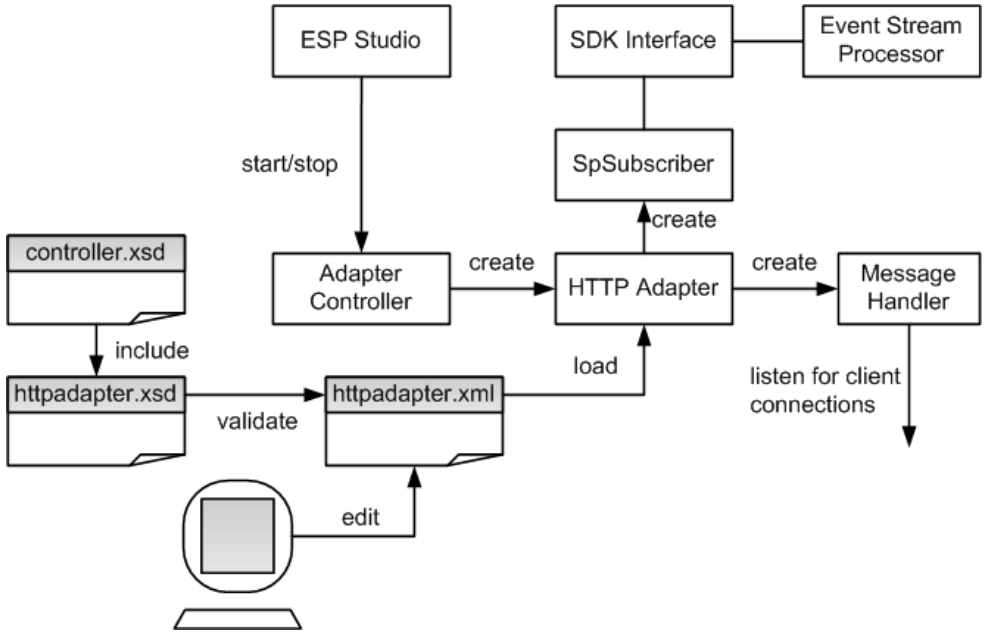
- Runs an internal HTTP server, listens to and accepts client connections
- Extracts SQL queries from client requests and subscribes to streams
- Converts stream records into XML, and sends XML to clients in chunk-coded HTTP responses

Control Flow

The adapter loads its configuration from a file (for example, `adapter.xml`) and validates it against the adapter schema (`httpadapter.xsd`), which includes the API-wide controller schema (`controller.xsd`).

You cannot edit schemas.

Figure 8: HTTP Adapter Control Flow



The Adapter Controller creates an instance of the adapter, receives and executes user commands. The Adapter Controller executes **start**, **stop**, and **status** commands.

Start Command

The **start** command configures and starts the adapter command and control interface, gets the Message Handler to start listening for client connections, and connects the SpSubscriber component to Event Stream Processor via the SDK interface.

The adapter ignores the **start** command if it is executed when there is a running instance of the adapter, and sends a warning.

See also

- *Starting the HTTP Adapter* on page 300

Stop Command

The **stop** command disconects the SpSubscriber from Event Stream Processor,causes the Message Handler to finalize the HTTP responses to the existing clients, disconnect them and stop listening for connections from new clients, and terminates he adapter process.

If the **stop** command is executed when there is no instance of a running adapter, the command is ignored and a warning is sent.

See also

- *Stopping the HTTP Adapter* on page 301

Status Command

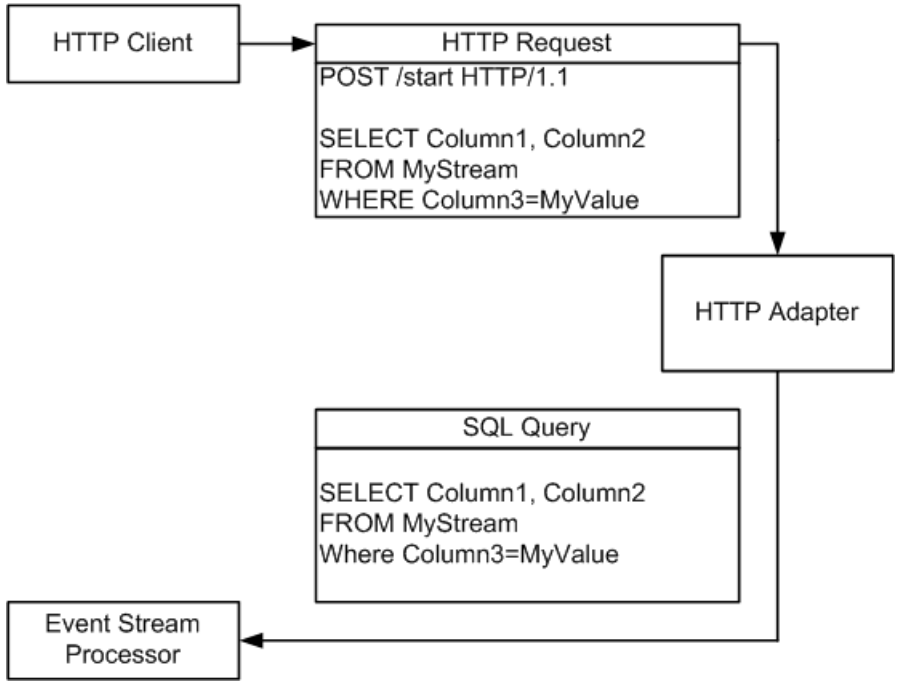
The **status** command reports the adapter status, and the Adapter Controller prints out its status: either running or stopped.

See also

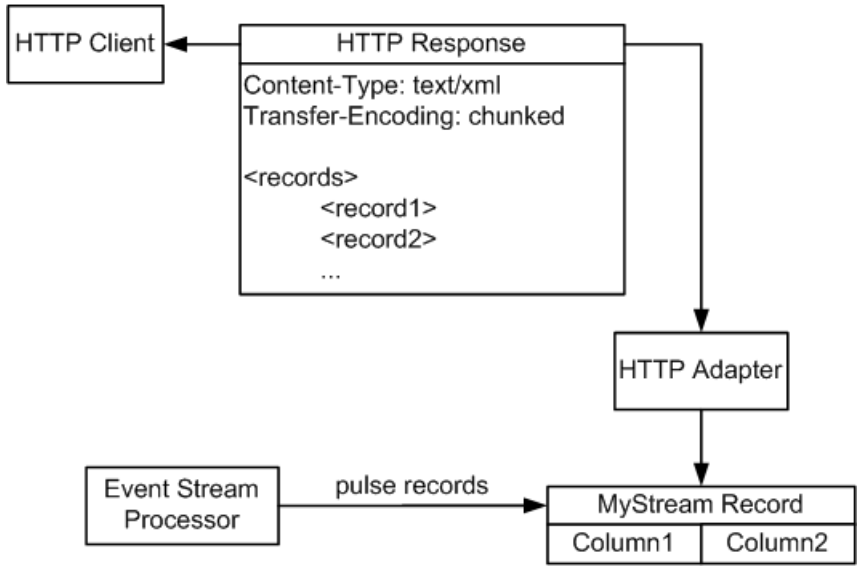
- *Checking the HTTP Adapter Status* on page 301

Message Flow

The message flow between the adapter and an HTTP client is initiated when the client sends a POST request with the **start** command in it, and a body consisting of a SQL query, to Event Stream Processor.



Changes in the corresponding stream are pulsed back to the HTTP client as XML-formatted chunk-coded HTTP responses.



Specify the pulse interval in the adapter configuration. In the event of a failover, the SDK API switches, as configured, to the spare Event Stream Processor instance without message loss.

See also

- *Event Stream Processor Parameters* on page 293

Setting the JAVA_HOME Environment Variable

Set the JAVA_HOME environment variable to point to the Java directory.

Prerequisites

Install Java Runtime Environment version 1.7.0_1 or higher. To see if you have a suitable version of Java installed, go to *http://www.java.com/en/download/installed.jsp*.

Task

Set the JAVA_HOME environment variable to the directory path where Java Runtime Environment 1.7.0_1 or higher is installed.

Configuration

Configuration information for the HTTP adapter.

HTTP Adapter Directory

The adapter directory contains all files, such as configuration files, templates, examples, and JAR files, relating to the adapter.

```
README.txt Quick Guide
ReleaseNotes.txt Release Notes
```

```
bin/
start_adapter.bat Standalone adapter startup script
start_adapter.sh Standalone adapter startup script
adapter-plugin.bat Plug-in connector startup script
adapter-plugin.sh Plug-in connector startup script

config/
controller.xsd Controller schema
log4j.properties Sample logging configuration
httpadapter.xsd Adapter schema
login.config Authentication configuration

examples/ Working example

libj/

commons-codec-1.3.jar Required by SDK API
commons-collections-3.2.1.jar
commons-configuration-1.6.jar
commons-lang-2.6.jar
commons-logging-1.1.jar Logging library
esp_adapter_api.jar Adapter API code
esp_adapter_http.jar http adapter library
esp_i18n.jar
esp_license.jar
esp_sdk.jar ESP SDK library
log4j-1.2.16.jar Logging library
postgresql.jar
sylapi.jar
ws-commons-util-1.0.2.jar Required by ESP SDK
xerces-impl-2.9.1.jar XML parser library
xmlrpc-client-3.1.3.jar Required by ESP SDK
xmlrpc-common-3.1.3.jar Required by ESP SDK
```

Schema and Configuration File

The adapter configuration is loaded from a file and validated against the adapter schema.

The `example` folder contains a sample adapter configuration file. Provide a valid configuration file, and ensure the adapter configuration validates against the adapter schema.

Adapter Controller Parameter

The **controllerPort** parameter specifies the adapter command and control port.

Parameter Name	Description
controllerPort	Type: `positive integer` (Required) Specifies the adapter command and control port. User commands are sent to this port on localhost.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the HTTP adapter.

These parameters are defined in the `controller.xsd` file in the `config` directory.

Parameter Name	Description
espAuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **espAuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
espUser	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **espAuthType**). No default value.
espPassword	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **espPassword** value is encrypted. Default value is false. If set to true, the password value is decrypted using **espRSAKeyStore** and **espRSAKeyStorePassword**.
espProjectUri	Type: `string` (Required) Specifies the total project URI to connect to Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.

Parameter Name	Description
pulseInterval	Type: `non-negative integer` (Optional) Specifies the time interval, in seconds, during which outbound record changes are coalesced by Event Stream Processor, then received by the adapter as a single event. If not set or set to 0, record changes are received individually as they occur.
espHeartbeatPeriod	Type: `positive integer` (Optional) Specifies the length of time, in seconds, that adapter waits before sending the next heartbeat to Event Stream Processor. If Event Stream Processor fails to receive two consecutive heartbeats, all records published by the adapter are marked stale. The default value is 10.
recordQueueCapacity	Type: `positive integer` (Optional) Specifies capacity of the record queues. Default value is 4096.
maxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.
maxPubPoolTime	Type: `positive integer` (Optional) Specifies the maximum period of time, in milliseconds, for which records are pooled before being published. If not set, pooling time is unlimited and the pooling strategy is governed by **maxPubPoolSize**. No default value.
useTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.

Parameter Name	Description
espRSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **espAuthType** is set to server_rsa, or the encrypted attribute for **espPassword** is set to true, or both.
espRSAKeyStorePassword	Type: `string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **espAuthType** is set to server_rsa, or the encrypted attribute for **espPassword** is set to true, or both.
espKerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **espAuthType** is set to kerberos.
espKerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **espAuthType** is set to kerberos.
espKerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **espAuthType** is set to kerberos.
espKerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **espAuthType** is set to kerberos.
espEncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **espPassword** is set to true. If left blank, RSA is used as default.

See also

- *Message Flow* on page 290

HTTP Server Settings

The **httpPort** and **contentType** parameters specify HTTP Server settings.

Parameter	Description
httpPort	Type: `integer` (Required) Specifies the port on which the adapter runs its HTTP server.
contentType	Type: `string` (Required) Specifies the content type of HTTP responses. The adapter supports the text/plain and text/html content types.

Sample HTTP Configuration File

Sample configuration file (`adapter.xml`) for the HTTP adapter.

This file is in the `example` folder.

```
<adapter>
- <!--  Adapter Controller
  -->
- <controller>
  <controllerPort>13579</controllerPort>
  </controller>
- <!--  Event Stream Processor Settings
  -->
- <esp>
- <espConnection>
  <espHost>localhost</espHost>
  <espPort>22000</espPort>
- <!--     <espProjectUri>esp://localhost:19011/ws1/p1</
espProjectUri>
  -->
  </espConnection>
- <espSecurity>
  <espUser>espuser</espUser>
  <espPassword encrypted="false">espuser</espPassword>
  <espAuthType>none</espAuthType>
- <!--
    <espRSAKeyFile>/keyfilepath/espuser.private.der</espRSAKeyFile>
    <espRSAKeyStore>/keystore/keystore.jks</espRSAKeyStore>
    <espRSAKeyStorePassword>Sybase123</espRSAKeyStorePassword>

  -->
  <espEncryptionAlgorithm>RSA</espEncryptionAlgorithm>
  </espSecurity>
  <maxPubPoolSize>1</maxPubPoolSize>
  </esp>
- <!--  HTTP specific
  -->
```

```
- <http>
  <httpPort>23456</httpPort>
  <contentType>text/html</contentType>
  </http>
  </adapter>
```

HTTP Output Adapter

The HTTP Output adapter receives SQL queries wrapped in HTTP requests from a client application, such as a Web browser, and sends chunk-coded stream content back to the client.

You can configure the adapter on any source stream as an outbound data location. The authentication method is set to Event Stream Processing standards: rsa, kerberos, or Native OS (user name/password). To use this adapter, ensure SAP HTTP adapter version 1.0 or later is installed.

Property Label	Description
Connector Directory Path	Property ID: **baseDir** Type: `directory` (Required) Specifies the absolute path to the adapter installation directory. This property is ignored if the **Connector Remote Directory Path** property is supplied. No default value. Use a forward slash for both UNIX and Windows paths.
Configuration File Path	Property ID: **configFilePath** Type: `configFilename` (Required) Specifies the absolute path to the adapter configuration file. This property is ignored if the **Remote Configuration Path** property is supplied. Ne default value. Use a forward slash for both UNIX and Windows paths.
Connector Remote Directory Path	Property ID: **remoteBaseDir** Type: `string` (Advanced) Specifies the path to the adapter remote base directory, for remote execution only. If this property is supplied, the **Connector Directory Path** is ignored. No default value. Use a forward slash for both UNIX and Windows paths.

Property Label	Description
Remote Configuration File Path	Property ID: **remoteConfigFilePath** Type: `string` (Advanced) Specifies the path to the adapter remote configuration file, for remote execution only. If this property is supplied, the **Configuration File Path** property is ignored. No default value. Use a forward slash for both UNIX and Windows paths.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Logging

The HTTP Output adapter uses the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing the logging configuration is part of the HTTP Output adapter distribution.

You can modify the logging levels of the `log4j.properties` configuration file which is located in the `%ESP_HOME%\adapters\<adapter-name>\config` directory. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.

Level	Description
INFO	Logs events for informational purposes.
DEBUG	Logs general debugging events.
TRACE	Logs fine-grained debug messages that capture the flow of the application.
ALL	Logs all events.

Note: Setting the log level to DEBUG or ALL may result in large log files. The default value is INFO.

Here is a sample log4j.properties file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The log4j.rootLogger option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Operation

You can operate the HTTP Output adapter from the command line.

Starting the HTTP Adapter

To start the HTTP adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

   ```
   cd %ESP_HOME%\cluster\nodes\node1
     %ESP_HOME%\bin\esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   %ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

   UNIX:

   a. Start the example cluster.

   ```
   cd $ESP_HOME/cluster/nodes/node1
     $ESP_HOME/bin/esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   $ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   $ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   $ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

2. Start the adapter.

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/http/bin`
	`./adapter.sh <configuration file path> start`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/http/bin`
	`adapter.bat <configuration file path> start`

See also

- *Start Command* on page 289

Checking the HTTP Adapter Status

To check the HTTP adapter status from the command line, execute the **status** command.

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/http/bin`
	`./adapter.sh <configuration file path> status`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/http/bin`
	`adapter.bat <configuration file path> status`

You see the adapter status: running or stopped.

See also

- *Status Command* on page 290

Stopping the HTTP Adapter

To stop the HTTP adapter from the command line, execute the **stop** command.

Prerequisites

When you are running the adapter from the command line, stop the adapter first before stopping the project.

Task

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/http/bin`
	`./adapter.sh <configuration file path> stop`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/http/bin`
	`adapter.bat <configuration file path> stop`

See also

- *Stop Command* on page 289

Example: Sending, Receiving, and Viewing Data

Use the working example provided in the adapter distribution to learn how to send a SQL query to the adapter, and receive XML-formatted stream data from Event Stream Processor and view it in a Web browser.

Prerequisites

A network connection.

Task

1. Set the username and password for the adapter:
   a) Edit `adapter.xml`.
   b) In the <User> and <Password> elements, enter `sybase`.
2. Set the username and password in the example environment:

Operating System	Step
UNIX	1. Edit the `set_example_env.sh` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	1. Edit the `set_example_env.bat` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

3. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

4. Edit the `start_adapter.sh` script.
5. Set the JAVA_HOME environment variable to the directory where the Java Runtime Environment (JRE) is installed.
6. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: `start_adapter.sh`
Windows	Open a command window and enter: `start_adapter.bat`

7. Wait five to ten seconds for the adapter to initialize.
8. Start uploading stream records.

Operating System	Step
UNIX	Open a terminal window and enter: `./upload.sh`
Windows	Open a command window and enter: `upload.bat`

9. Load the HTTPAdapterClient.html page in a Web browser.
10. Enter a valid SQL query, for example:

```
SELECT * FROM Stream1
```

Note: Queries that return a smaller number of records may not appear on some browsers because those browsers appear to expect a certain amount of data to be present in the buffer cache before they display the data. For example:

```
SELECT * FROM Stream1
where intcol = 10
```

- Mozilla FireFox browser -- displays record.
- Google Chrome browser -- does not display record.
- Internet Explorer browser -- does not display record.

```
SELECT * FROM Stream1
where intCol > 10
```

(executed in debug mode)

- Mozilla FireFox browser -- records appear after they are sent to the browser.
- Google Chrome browser -- records appear after the fifth record is sent to the browser.
- Internet Explorer browser -- records appear after the 20th record is sent to the browser.

11. Click **Submit**.
    Note the records being streamed into the Web browser window.

JDBC Input and Output Adapter

The JDBC Input adapter receives data from tables in a database and inputs it into Event Stream Processor. The JDBC Output adapter sends data from Event Stream Processor to a database table.

JDBC Input Adapter Configuration

Configure the JDBC Input adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

Transporter Module: JDBC Input Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **JdbcInputTransporterParameters** element.
JdbcInputTransporterParameters	(Required) Section containing parameters for the JDBC Input transporter.
Host	Type: `string` (Required) Specify the server name of the database to which you are connecting the adapter.
Port	Type: `integer` (Required) Specify the port number for connecting to the database server.
Username	Type: `string` (Required) Specify the username you are using to connect to the database server.

Parameter	Description
Password	Type: `string` (Required) Specify the password for connecting to the database server. Includes an "encrypted" attribute indicating whether the password value is encrypted. If set to true, the password value is decrypted using the **RSAKeyStore**, **RSAKeyStorePassword**, and **RSAKeyStoreAlias** parameters.
DbName	Type: `string` (Required) Specify the database name to which you want to connect.
DBType	Type: `string` (Required) Specify the database type to which you want to connect.
DbDriver	Type: `string` (Required) Specify the JDBC driver class for your JDBC driver.
Table	Type: `string` (Optional) Specify the name of the table in the target database from which you want the adapter to read.
Query	Type: `string` (Optional) Specify which SQL query you want the adapter to execute. No default value. Set either the **Table** or **Query** parameter. If you define both parameters, the adapter only uses the **Query** parameter.
RSAKeyStore	Type: `string` (Dependent required) Specify the location of an RSA keystore file which contains the key used to encrypt or decrypt the password set in the **Password** parameter. This parameter is required if the password value is encrypted.

Parameter	Description
RSAKeyStorePassword	Type: `string` (Dependent required) Stores the password to the RSA keystore file specified in the **RSAKeyStore** parameter. This parameter is required if the password value is encrypted.
RSAKeyStoreAlias	Type: `string` (Dependent required) Specifies the keystore alias. This parameter is required if the password value is encrypted.

ESP Connector Module: ESP Publisher

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **EspPublisherParameters** parameter.
EspPublisherParameters	(Required) Section containing parameters for the ESP publisher.

Parameter	Description
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream to which the adapter publishes data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
MaxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.

Parameter	Description
MaxPubPoolTime	Type: `positive integer` (Optional) Specifies the maximum period of time, in milliseconds, for which records are pooled before being published. If not set, pooling time is unlimited and the pooling strategy is governed by **maxPubPoolSize**. No default value.
UseTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.
SafeOps	Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT, and converts DELETE to SAFEDELETE. The default value is false.
SkipDels	Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. The default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the JDBC Input adapter. You can specify multiple projects.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.

Parameter	Description
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.

Parameter	Description
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

See also

- *Adapter Support for Schema Discovery* on page 1005

JDBC Input Adapter Studio Properties

Adapter type: toolkit_jdbc_objlist_input. Set these properties for the JDBC Input adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Date Format	Propterty ID: **objectlistDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **objectlistTimestampFormat** Type: `string` (Advanced) Specify the format for parsing time-stamp values.
DB Type	Property ID: **dbType** Type: `string` (Required) Specify the database type to which you want to connect. Valid values: ORA-CLE:THIN, ORACLE:OCI8, SQLSERVER, ASE, DB2:APP, DB2:NET, DB2:UN2, HANA, KDB.
Host	Property ID: **host** Type: `string` (Required) Specify the server name or IP address of the database to which you are connecting.
Port	Property ID: **port** Type: `int` (Required) Specify the port for the database to which you are connecting.

Property Label	Description
DB Name	Property ID: **dbName** Type: `string` (Optional) Specify the database name to which you want to connect.
Username	Property ID: **user** Type: `string` (Optional) Specify the database username.
Password	Property ID: **password** Type: `string` (Optional) Specify the database password.
SQL Query	Property ID: **query** Type: `string` (Advanced) Specify the SQL query you want the adapter to execute. Set either the DB Table or SQL Query parameter. If you define both parameters, the adapter only uses the SQL Query parameter.
Table Name	Property ID: **table** Type: `string` (Optional) Specify the name of the table in the target database from which you want the adapter to read.
Driver Class	Property ID: **dbDriver** Type: `string` (Advanced) Specify the JDBC driver class to use.

Property Label	Description
DB Password Encrypted	Property ID: **encrypted** Type: `boolean` (Advanced) Specify whether the password value is encrypted. If set to true, the password value is decrypted using the RSA Key Store, RSA Key Store Password, and RSA Key Store Alias parameters.
RSA Key Store	Property ID: **rsaKeyStore** Type: `filename` (Advanced) File path for RSA key store.
RSA Key Store Password	Property ID: **rsaKeyStorePassword** Type: `string` (Advanced) RSA key store password.
RSA Key Store Alias	Property ID: **rsaKeyStoreAlias** Type: `string` (Advanced) An alias stored in the RSA key store.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: JDBC Input Adapter

Sample adapter configuration file for the JDBC Input adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>jdbc_input</Name>
    <Description>An adapter which gets data from database through
```

```
JDBC interface then publish to ESP stream.</Description>
    <Log4jProperty>./log4j.properties</Log4jProperty>
    <Modules>
        <Module type="transporter">
            <InstanceName>MyJdbcInputTransporter</InstanceName>
            <Name>JdbcInputTransporter</Name>
            <Next>ObjectListToEspFormatterInstance</Next>
            <Parameters>
                <JdbcInputTransporterParameters>
                    <Host>10.128.99.92</Host>
                    <Port>5000</Port>
                    <Username></Username>
                    <Password encrypted="false"></Password>
                    <!-- <RSAKeyStore>/keystore/keystore.jks</
RSAKeyStore> <RSAKeyStorePassword>sybase</RSAKeyStorePassword>
                     <RSAKeyStoreAlias>serverkey</RSAKeyStoreAlias>
-->
                    <DbName>master</DbName>
                    <DbType>ASE</DbType>
                    <DbDriver>com.sybase.jdbc4.jdbc.SybDriver</
DbDriver>
                    <Table>MyInStream</Table>
                </JdbcInputTransporterParameters>
            </Parameters>
        </Module>

        <Module type="formatter">
            <InstanceName>ObjectListToEspFormatterInstance</
InstanceName>
            <Name>ObjectListToEspFormatter</Name>
            <Next>MyInStream_Publisher</Next>
            <Parallel>true</Parallel>
            <Parameters>
                <ObjectListToEspFormatterParameters>
                    <DateFormat>yyyy-MM-dd HH:mm:ss.SSS</DateFormat>
                    <TimestampFormat>yyyy-MM-dd HH:mm:ss.SSS</
TimestampFormat>
                </ObjectListToEspFormatterParameters>
            </Parameters>
        </Module>


        <Module type="espconnector">
            <InstanceName>MyInStream_Publisher</InstanceName>
            <Name>EspPublisher</Name>
            <Parameters>
                <EspPublisherParameters>
                    <ProjectName>EspProject1</ProjectName>
                    <StreamName>MyInStream</StreamName>
                    <MaxPubPoolSize>1</MaxPubPoolSize>
                    <UseTransactions>false</UseTransactions>
                    <SafeOps>true</SafeOps>
                    <SkipDels>true</SkipDels>
                </EspPublisherParameters>
            </Parameters>
        </Module>
```

```
    </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject1</Name>
            <Uri>esp://localhost:19011/sample_workspace/
jdbc_objlist_input</Uri>
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
             <!-- <RSAKeyStore>/keystore/keystore.jks</RSAKeyStore>
<RSAKeyStorePassword>Sybase123</RSAKeyStorePassword> -->
                <!-- <KerberosKDC>KDC</KerberosKDC>
<KerberosRealm>REALM</KerberosRealm>
                 <KerberosService>service/instance</KerberosService>
<KerberosTicketCache>/tmp/krb5cc_user</KerberosTicketCache> -->
                <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
            </Security>
        </EspProject>


    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

JDBC Output Adapter Configuration

Configure the JDBC Output adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

ESP Connector Module: ESP Subscriber

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, espconnector.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **EspSubscriberParameters** parameter.
EspSubscriberParameters	(Required) Section containing parameters for the ESP subscriber.

Parameter	Description
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Specifies the unique project tag of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream from which the adapter subscribes to data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.

Transporter Module: JDBC Output Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **JdbcOutputTransporterParameters** parameter.
JdbcOutputTransporterParameters	(Required) Section containing the JDBC Output transporter parameters.
Host	Type: `string` (Required) Specify the server name of the database to which you are connecting the adapter.
Port	Type: `integer` (Required) Specify the port number for connecting to the database server.
Username	Type: `string` (Required) Specify the username you are using to connect to the database server.
Password	Type: `string` (Required) Specify the password for connecting to the database server. Includes an "encrypted" attribute indicating whether the password value is encrypted. If set to true, the password value is decrypted using the **RSAKeyStore**, **RSAKeyStorePassword**, and **RSAKeyStoreAlias** parameters.
DbName	Type: `string` (Required) Specify the database name to which you want to connect.
DBType	Type: `string` (Required) Specify the database type to which you want to connect.

Parameter	Description
DbDriver	Type: `string` (Required) Specify the JDBC driver class for your JDBC driver.
Table	Type: `string` (Optional) Specify the name of the table in the target database to which you want the adapter to write.
SqlInsert	Type: `string` (Optional) Specify which SQL clause you want the adapter to execute. No default value. Set either the **Table** or **SqlInsert** parameter. If you define both parameters, the adapter only uses the **SqlInsert** parameter.
RSAKeyStore	Type: `string` (Dependent required) Specify the location of an RSA keystore file which contains the key used to encrypt or decrypt the password set in the **Password** parameter. This parameter is required if the password value is encrypted.
RSAKeyStorePassword	Type: `string` (Dependent required) Stores the password to the RSA keystore file specified in the **RSAKeyStore** parameter. This parameter is required if the password value is encrypted.
RSAKeyStoreAlias	Type: `string` (Dependent required) Specifies the keystore alias. This parameter is required if the password value is encrypted.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the JDBC Output adapter. You can specify multiple ESP projects.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.

Parameter	Description
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using key-store • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

See also

- *Adapter Support for Schema Discovery* on page 1005

JDBC Output Adapter Studio Properties

Adapter type: toolkit_jdbc_objlist_output. Set these properties for the JDBC Output adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
DB Type	Property ID: **dbType** Type: `string` (Required) Specify the database type to which you want to connect. Valid values: ORACLE:THIN, ORACLE:OCI8, SQLSERVER, ASE, DB2:APP, DB2:NET, DB2:UN2, HANA, KDB.
Host	Property ID: **host** Type: `string` (Required) Specify the server name or IP address of the database to which you are connecting.

Property Label	Description
Port	Property ID: **port** Type: `int` (Required) Specify the port for the database to which you are connecting.
DB Name	Property ID: **dbName** Type: `string` (Optional) Specify the database name to which you want to connect.
Username	Property ID: **user** Type: `string` (Optional) Specify the database username.
Password	Property ID: **password** Type: `string` (Optional) Specify the database password.
SQL Insert	Property ID: **SqlInsert** Type: `string` (Advanced) Specify the SQL insert clause you want the adapter to execute. Set either the DB Table or SQL Insert parameter. If you define both parameters, the adapter only uses the SQL Insert parameter.
Table Name	Property ID: **table** Type: `string` (Optional) Specify the name of the table in the target database from which you want the adapter to write.
Driver Class	Property ID: **dbDriver** Type: `string` (Advanced) Specify the JDBC driver class to use.

Property Label	Description
DB Password Encrypted	Property ID: **encrypted** Type: `boolean` (Advanced) Specify whether the password value is encrypted. If set to true, the password value is decrypted using the RSA Key Store, RSA Key Store Password, and RSA Key Store Alias parameters.
RSA Key Store	Property ID: **rsaKeyStore** Type: `filename` (Advanced) File path for RSA key store.
RSA Key Store Password	Property ID: **rsaKeyStorePassword** Type: `string` (Advanced) RSA key store password.
RSA Key Store Alias	Property ID: **rsaKeyStoreAlias** Type: `string` (Advanced) An alias stored in the RSA key store.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: JDBC Output Adapter

Sample adapter configuration file for the JDBC Output adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>jdbc_objlist_output</Name>
   <Description>An adapter which ouputs ESP data to datbase through
```

```
JDBC interface.</Description>
    <Log4jProperty>./log4j.properties</Log4jProperty>
    <Modules>
        <Module type="espconnector">
            <InstanceName>MyOutStream_Subscriber</InstanceName>
            <Name>EspSubscriber</Name>
            <Next>EspToObjectListFormatterInstance</Next>
            <Parameters>
                <EspSubscriberParameters>
                    <ProjectName>EspProject1</ProjectName>
                    <StreamName>MyInStream</StreamName>
                </EspSubscriberParameters>
            </Parameters>
        </Module>

        <Module type="formatter">
            <InstanceName>EspToObjectListFormatterInstance</
InstanceName>
            <Name>EspToObjectListFormatter</Name>
            <Next>MyJdbcOutputTransporter</Next>
            <Parallel>true</Parallel>
            <Parameters>
                <EspToObjectListFormatterParameters>
                  <!--For output adapter with JDBC transporter, the
value must be "true".
                   Then, the java.sql.Date will be used as output
type for ESP DATE and TIMESTAMP
                     value instead of java.util.Date. The
java.sql.Timestamp will be used as output
                     type for ESP BIGDATETIME value instead of
java.util.Date. -->
                    <OutputAsSQLDatetimeFormat>true</
OutputAsSQLDatetimeFormat>
                </EspToObjectListFormatterParameters>
            </Parameters>
        </Module>

        <Module type="transporter">
            <InstanceName>MyJdbcOutputTransporter</InstanceName>
            <Name>JdbcOutputTransporter</Name>
            <Parameters>
                <JdbcOutputTransporterParameters>
                    <Host>10.128.99.92</Host>
                    <Port>5000</Port>
                    <Username></Username>
                    <Password encrypted="false"></Password>
                    <!-- <RSAKeyStore>/keystore/keystore.jks</
RSAKeyStore> <RSAKeyStorePassword>sybase</RSAKeyStorePassword>
                     <RSAKeyStoreAlias>serverkey</RSAKeyStoreAlias>
-->
                    <DbName>master</DbName>
                    <DbType>ASE</DbType>
                    <DbDriver>com.sybase.jdbc4.jdbc.SybDriver</
DbDriver>
                    <Table>MyInStream</Table>
                </JdbcOutputTransporterParameters>
```

```
            </Parameters>
        </Module>

    </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject1</Name>
            <Uri>esp://localhost:19011/sample_workspace/
jdbc_objlist_output</Uri>
            <!--can specify multiple uri's -->
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
             <!-- <RSAKeyStore>/keystore/keystore.jks</RSAKeyStore>
<RSAKeyStorePassword>Sybase123</RSAKeyStorePassword> -->
                <!-- <KerberosKDC>KDC</KerberosKDC>
<KerberosRealm>REALM</KerberosRealm>
                 <KerberosService>service/instance</KerberosService>
<KerberosTicketCache>/tmp/krb5cc_user</KerberosTicketCache> -->
                <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
            </Security>
        </EspProject>

    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Adapter Controller Parameters

The Adapter Controller port listens for commands. The `controller.xml` file is located in `%ESP_HOME%/adapters/framework/config` directory. This file is shared among all the adapter within the `%ESP_HOME%/adapters/framework/instances` directory.

Parameter	Description
ControlPort	(Required) Section containing the **MinPort** and **MaxPort** parameters.
MinPort	Type: `int` (Required) The minimum port number that can act as a control port number. The adapter framework allocates this number automatically. A recommended value is 19082.

Parameter	Description
MaxPort	Type: `int` (Optional) The maximum port number that can act as a control port number. The adapter framework allocates this number automatically. The default value is 65535.
Statistics_Update_Interval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework publishes statistics data to the ESP project to which the adapter is connected. A recommended value is 5000.
MonitorInterval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework checks the status of internal resource allocation. The framework logs a message in the log file if a resource is about to be exhausted. A recommended value is 3000.

Logging

The JDBC Input and Output adapter use the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing the logging configuration is located in the `%ESP_HOME%\adapters\framework\config` directory.

Specify the location of the logging file you wish to use in the **Log4jProperty** parameter within the adapter configuration file. You can modify the logging levels within this file or the `$ESP_HOME/adapters/framework/config/log4j.properties`, which is used by default. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.

Level	Description
WARN	Logs events that possibly lead to errors.
INFO	Logs events for informational purposes.
DEBUG	Logs general debugging events.
TRACE	Logs fine-grained debug messages that capture the flow of the application.
ALL	Logs all events.

Note: Setting the log level to DEBUG or ALL may result in large log files. The default value is INFO.

Here is a sample log4j.properties file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Starting the JDBC Adapter

To start the JDBC adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

   ```
   cd %ESP_HOME%\cluster\nodes\node1
     %ESP_HOME%\bin\esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   %ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

   UNIX:

   a. Start the example cluster.

   ```
   cd $ESP_HOME/cluster/nodes/node1
     $ESP_HOME/bin/esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   $ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   $ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   $ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

2. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: For the JDBC Input adapter: `cd $ESP_HOME/adapters/framework/instances/jdbc_input` For the JDBC Output adapter: `cd $ESP_HOME/adapters/framework/instances/jdbc_output` `./start_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the JDBC Input adapter: `cd %ESP_HOME%/adapters/framework/instances/jdbc_input` For the JDBC Output adapter: `cd %ESP_HOME%/adapters/framework/instances/jdbc_output` `start_adapter.bat <adapter configuration file name>`

Stopping the JDBC Adapter

To stop the JDBC adapter from the command line, execute the **stop** command.

Operating System	Step
UNIX	Open a terminal window and enter: For the JDBC Input adapter: `cd $ESP_HOME/adapters/framework/instances/jdbc_input` For the JDBC Output adapter: `cd $ESP_HOME/adapters/framework/instances/jdbc_output` `./stop_adapter.sh <adapter configuration file name>`

Operating System	Step
Windows	Open a command window and enter: For the JDBC Input adapter: `cd %ESP_HOME%/adapters/framework/instances/jdbc_input` For the JDBC Output adapter: `cd %ESP_HOME%/adapters/framework/instances/jdbc_output` `stop_adapter.bat <adapter configuration file name>`

JMS Adapter

Event Stream Processor supports these types of JMS Input and Output adapters: CSV, FIX, Object Array, and XML.

Configuring a Queuing System for JMS Adapter

To use the JMS adapters, configure the queuing system with the naming server.

Prerequisites

Queuing systems that support JNDI naming servers.

Task

1. Set up a naming server.

   Some queuing systems contain internal naming servers that could connect to JMS adapters.
2. Set up the queuing system to use the naming server to administer JMS objects.

   For more information, consult the documentation provided with your third-party queuing system.
3. Obtain the JNDI context library and the URL to get to the naming server.

   Note: To use the SAP Sybase Event Stream Processor Adapter for JMS to integrate or communicate with TIBCO Enterprise Message Services, you must have a valid license for TIBCO Enterprise Message Services from TIBCO or from an authorized TIBCO channel.

   For example, for Apache Active MQ, these are `org.apache.activemq.jndi.ActiveMQInitialContextFactory` and `tcp://localhost:61616`.
4. Set the **jndiContextFactory** and **jndiURL** properties for the JMS adapter.

5. Obtain the name that the JMS connection factory is bound by.
6. Set the **connectionFactory** property to this name for the adapter.
7. Ensure that appropriate `JNDI` and `JMS factory` classes are in the Java class path.

Note: To run the adapter, obtain and place vendor specific JMS jar files in the `$ESP_HOME/java` folder and restart the ESP Server.

JMS CSV Input and Output Adapter

The JMS CSV Input adapter reads CSV data from a JMS server ands outputs this data into Event Stream Processor. The JMS CSV Output adapter sends CSV data from Event Stream Processor to a JMS server.

JMS CSV Input Adapter Configuration

Configure the JMS CSV Input adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

Transporter Module: JMS Input Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
BufferMaxSize	Type: `integer` (Advanced) Specify the capacity of the buffer queue between the transporter module and the next module. The default value is 10240.
Parameters	(Required) Section containing the **JMSInputTransporterParameters** element.
JMSInputTransporterParameters	(Required) Section containing parameters for the JMS Input transporter.
ConnectionFactory	Type: `string` (Required) Specify the connection factory class name. No default value.
JndiContextFactory	Type: `string` (Required) Specify the context factory for JNDI context initialization. No default value.
JndiUrl	Type: `string` (Required) Specify the JNDI URL. No default value.
Destination Type	Type: `string` (Required) Specify the destination type. Valid values are: QUEUE and TOPIC. The default value is QUEUE.
DestinationName	Type: `string` (Required) Specify the destination name. No default value.

Parameter	Description
MessageType	Type: `string` (Required) Specify the message type you want the JMS transporter to process. These types are supported: • TEXT - for receiving and sending messages in text string • OBJARRAY - for receiving and sending messages in custom format No default value.
SubscriptionMode	Type: `string` (Optional) Specify the subscription mode for TOPIC (see the **Destination Type** parameter). Valid values are DURABLE and NONDURABLE. Default value is NONDURABLE.
ScanDepth	Type: `integer` (Optional) Specify the depth of the schema discovery. The adapter reads the number of rows specified by this parameter value when discovering the input data schema. The default value is three.
ClientID	Type: `string` (Required for DURABLE subscription mode only) Specifies the client identifier for a JMS client. Required for creating a durable subscription in JMS. Can be any string, but must be unique for each topic. No default value. Example: `client1` .

Parameter	Description
SubscriptionName	Type: `string` (Required for DURABLE subscription mode only) Specifies a unique name identifying a durable subscription. Required for creating a durable subscription in JMS. Can be any string, but must be unique within a given client ID. No default value. Example: `subscription1`.

Formatter Module: CSV String to ESP Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **CSVString-ToEspFormatterParameters** parameter.
CSVStringToEspFormatterParameters	(Required) Section containing the CSV String to ESP formatter parameters.

Parameter	Description
ExpectStreamNameOpcode	Type: `boolean` (Required) If set to true, the adapter interprets the first two fields of the incoming CSV line as stream name and opcode respectively. The adapter discards messages with unmatched stream names. The accepted opcodes are: • i or I: INSERT • d or D: DELETE • u or U: UPDATE • p or P: UPSERT • s or S: SAFEDELETE The default value is false.
Delimiter	Type: `string` (Advanced) The symbols used to separate the column. The default value is a comma (,).
HasHeader	Type: `boolean` (Advanced) Determines whether the first line of the file contains the description of the fields. Default value is false.
DateFormat	Type: `string` (Advanced) The format string for parsing date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.
TimestampFormat	Type: `string` (Advanced) Format string for parsing timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

ESP Connector Module: ESP Publisher

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **EspPublisherParameters** parameter.
EspPublisherParameters	(Required) Section containing parameters for the ESP publisher.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.

Parameter	Description
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream to which the adapter publishes data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
MaxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.
MaxPubPoolTime	Type: `positive integer` (Optional) Specifies the maximum period of time, in milliseconds, for which records are pooled before being published. If not set, pooling time is unlimited and the pooling strategy is governed by **maxPubPoolSize**. No default value.
UseTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.
SafeOps	Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT, and converts DELETE to SAFEDELETE. The default value is false.

Parameter	Description
SkipDels	Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. The default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the JMS CSV Input adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.

Parameter	Description
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

See also

- *Adapter Support for Schema Discovery* on page 1005

JMS CSV Input Adapter Studio Properties

Adapter type: toolkit_jms_csv_input. Set these properties for the JMS CSV Input adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Column Delimiter	Property ID: **csvDelimiter** Type: `string` (Advanced) Specify the symbol used to separate the columns.

Property Label	Description
Date Format	Propterty ID: **csvDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **csvTimestampFormat** Type: `string` (Advanced) Specify the format for parsing time-stamp values.
Has Header	Property ID: **csvHasHeader** Type: `boolean` (Advanced) Indicate whether the first line of the file contains the description of the CSV fields.
Steam name, opcode expected	Property ID: **csvExpectStreamNameOpcode** Type: `boolean` (Optional) If set to true, the adapter interprets the first two fields of the incoming CSV line as stream name and opcode. The adapter discards messages with unmatched values.
Connection Factory	Property ID: **connectionFactory** Type: `string` (Required) Connection factory name.
JNDI Context Factory	Property ID: **jndiContextFactory** Type: `string` (Required) Context factory for JDNI context initialization.
JNDI URL	Property ID: **jdniURL** Type: `string` (Required) JDNI URL.

Property Label	Description
JMS Destination Type	Property ID: **destinationType** Type: `string` (Required) Specify the destination type. Valid values: QUEUE, TOPIC.
Destination Name	Property ID: **destinationName** Type: `string` (Required) Specify the destination name.
Subscription Mode	Property ID: **subscriptionMode** Type: `string` (Optional) Specify the subscription mode. Valid values: DURABLE, NONDURABLE.
Scan Depth	Property ID: **scanDepth** Type: `int` (Advanced) Specify the depth of the schema discovery. The adapter reads the number of rows specified by this parameter value when discovering the input data scheme.
Client ID	Property ID: **clientID** Type: `string` (Advanced) Specify the client identifier (required when durable subscription is enabled).
Subscription Name	Property ID: **subscriptionName** Type: `string` (Advanced) Specify a unique name identifying a durable subscription.

Property Label	Description
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: JMS CSV Input Adapter

Sample adapter configuration file for the JMS CSV Input adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>jms_csv_input</Name>
   <Description>An adapter which receives CSV string mesage from JMS
server, transforms to ESP data format, and publishes to ESP stream.</
Description>
    <Log4jProperty>./log4j.properties</Log4jProperty>
    <Modules>
        <Module type="transporter">
            <InstanceName>JmsInputTransporter</InstanceName>
            <Name>JmsInputTransporter</Name>
            <Next>CSVInputFormatter</Next>
            <BufferMaxSize>10240</BufferMaxSize>
            <Parameters>
                <JMSInputTransporterParameters>
                    <ConnectionFactory>ConnectionFactory</
ConnectionFactory>

<JndiContextFactory>org.apache.activemq.jndi.ActiveMQInitialContext
Factory</JndiContextFactory>
                    <JndiURL>tcp://localhost:61616</JndiURL>
                    <DestinationName>queue.jmscsv.test</
DestinationName>
                    <DestinationType>QUEUE</DestinationType>
                    <MessageType>TEXT</MessageType>
                    <ScanDepth>3</ScanDepth>
                </JMSInputTransporterParameters>
            </Parameters>
        </Module>

        <Module type="formatter">
            <InstanceName>CSVInputFormatter</InstanceName>
```

```
            <Name>CsvStringToEspFormatter</Name>
            <Next>MyInStream_Publisher</Next>
            <Parallel>true</Parallel>
            <Parameters>
                <CsvStringToEspFormatterParameters>
                    <HasHeader>true</HasHeader>
                    <ExpectStreamNameOpcode>true</
ExpectStreamNameOpcode>
                </CsvStringToEspFormatterParameters>
            </Parameters>
        </Module>

        <Module type="espconnector">
            <InstanceName>MyInStream_Publisher</InstanceName>
            <Name>EspPublisher</Name>
            <Parameters>
                <EspPublisherParameters>
                    <ProjectName>EspProject1</ProjectName>
                    <StreamName>BaseInput</StreamName>
                    <MaxPubPoolSize>1</MaxPubPoolSize>
                    <UseTransactions>false</UseTransactions>
                </EspPublisherParameters>
            </Parameters>
            <BufferMaxSize>10240</BufferMaxSize>
        </Module>
    </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject1</Name>
            <Uri>esp://localhost:19011/sample_workspace/
jms_csv_input</Uri>
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

JMS CSV Output Adapter Configuration

Configure the JMS CSV Output adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

ESPConnector Module: ESP Subscriber

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, espconnector.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	Section containing the **EspSubscriberParameters** parameter.
EspSubscriberParameters	(Required) Section containing parameters for the ESP subscriber.

Parameter	Description
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Specifies the unique project tag of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream from which the adapter subscribes to data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.

Formatter Module: ESP to CSV String Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **EspToCsvStringFormatterParameters** parameter.
EspToCsvStringFormatterParameters	(Required) Section containing the ESP to CSV String formatter parameters.
PrependStreamNameOpcode	Type: `boolean` (Optional) If set to true, the adapter prepends the stream name and the opcode in each row of data that is generated. The default value is false.
Delimiter	Type: `string` (Advanced) The symbols used to separate the column. The default value is a comma (,).
HasHeader	Type: `boolean` (Advanced) Determines whether the first line of the file contains the description of the fields. The default value is false.
DateFormat	Type: `string` (Advanced) The format string for date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.
TimestampFormat	Type: `string` (Advanced) Format string for timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

Transporter Module: JMS Output Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **JMSOutputTransporterParameters** parameter.
JMSOutputTransporterParameters	(Required) Section containing the JMS Output transporter parameters.
ConnectionFactory	Type: `string` (Required) Specify the connection factory class name. No default value.
JndiContextFactory	Type: `string` (Required) Specify the context factory for JNDI context initialization. No default value.
JndiUrl	Type: `string` (Required) Specify the JNDI URL. No default value.
Destination Type	Type: `string` (Required) Specify the destination type. Valid values are: QUEUE and TOPIC. The default value is QUEUE.

Parameter	Description
DestinationName	Type: `string` (Required) Specify the destination name. No default value.
MessageType	Type: `string` (Required) Specify the message type you want the JMS transporter to process. These types are supported: • TEXT - for receiving and sending messages in text string • OBJARRAY - for receiving and sending messages in custom format No default value.
DeliveryMode	Type: `string` (Optional) Specify the delivery mode type. Valid values are: • PERSISTENT • NON_PERSISTENT Default value is PERSISTENT.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the JMS CSV Output adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.

Parameter	Description
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.

Parameter	Description
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

JMS CSV Output Adapter Studio Properties

Adapter type: toolkit_jms_csv_output. Set these properties for the JMS CSV Output adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Column Delimiter	Property ID: **csvDelimiter** Type: `string` (Advanced) Specify the symbol used to separate the columns.
Date Format	Propterty ID: **csvDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **csvTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.
Has Header	Property ID: **csvHasHeader** Type: `boolean` (Advanced) Indicate whether the first line of the file contains the description of the CSV fields.
Prepend stream name, opcode	Property ID: **csvPrependStreamNameOpcode** Type: `boolean` (Optional) If set to true, the adapter prepends the stream name and the opcode in each row of data that is generated.
Connection Factory	Property ID: **connectionFactory** Type: `string` (Required) Connection factory name.

Property Label	Description
JNDI Context Factory	Property ID: **jndiContextFactory** Type: `string` (Required) Context factory for JDNI context initialization.
JNDI URL	Property ID: **jdniURL** Type: `string` (Required) JDNI URL.
JMS Destination Type	Property ID: **destinationType** Type: `string` (Required) Specify the destination type. Valid values: QUEUE, TOPIC.
Destination Name	Property ID: **destinationName** Type: `string` (Required) Specify the destination name.
Delivery Mode	Property ID: **deliveryMode** Type: `string` (Optional) Specify the delivery mode. Valid values: PERSISTENT, NON_PERSISTENT.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: JMS CSV Output Adapter

Sample adapter configuration file for the JMS CSV Output adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>jms_csv_output</Name>
    <Description>An adapter which transforms ESP data to CSV format
and sends out the data to JMS server.</Description>
    <Log4jProperty>./log4j.properties</Log4jProperty>
    <Modules>
        <Module type="espconnector">
            <InstanceName>MyOutStream_Subscriber</InstanceName>
            <Name>EspSubscriber</Name>
            <Next>CSVOutputFormatter</Next>
            <Parameters>
                <EspSubscriberParameters>
                    <ProjectName>EspProject2</ProjectName>
                    <StreamName>BaseOutput</StreamName>
                </EspSubscriberParameters>
            </Parameters>
        </Module>

        <Module type="formatter">
            <InstanceName>CSVOutputFormatter</InstanceName>
            <Name>EspToCsvStringFormatter</Name>
            <Next>JmsOutputTransporter</Next>
            <Parallel>true</Parallel>
            <Parameters>
                <EspToCsvStringFormatterParameters>
                    <HasHeader>true</HasHeader>
                    <PrependStreamNameOpcode>true</
PrependStreamNameOpcode>
                </EspToCsvStringFormatterParameters>
            </Parameters>
        </Module>

        <Module type="transporter">
            <InstanceName>JmsOutputTransporter</InstanceName>
            <Name>JmsOutputTransporter</Name>
            <Parameters>
                <JMSOutputTransporterParameters>
                    <ConnectionFactory>ConnectionFactory</
ConnectionFactory>

<JndiContextFactory>org.apache.activemq.jndi.ActiveMQInitialContext
Factory</JndiContextFactory>
                    <JndiURL>tcp://localhost:61616</JndiURL>
                    <DestinationName>queue.jmscsv.test</
DestinationName>
                    <DestinationType>QUEUE</DestinationType>
                    <MessageType>TEXT</MessageType>
                </JMSOutputTransporterParameters>
            </Parameters>
        </Module>
```

```
    </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject2</Name>
            <Uri>esp://localhost:19011/sample_workspace/
jms_csv_output</Uri>
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Adapter Controller Parameters

The Adapter Controller port listens for commands. The `controller.xml` file is located in `%ESP_HOME%/adapters/framework/config` directory. This file is shared among all the adapter within the `%ESP_HOME%/adapters/framework/instances` directory.

Parameter	Description
ControlPort	(Required) Section containing the **MinPort** and **MaxPort** parameters.
MinPort	Type: `int` (Required) The minimum port number that can act as a control port number. The adapter framework allocates this number automatically. A recommended value is 19082.
MaxPort	Type: `int` (Optional) The maximum port number that can act as a control port number. The adapter framework allocates this number automatically. The default value is 65535.
Statistics_Update_Interval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework publishes statistics data to the ESP project to which the adapter is connected. A recommended value is 5000.

Parameter	Description
MonitorInterval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework checks the status of internal resource allocation. The framework logs a message in the log file if a resource is about to be exhausted. A recommended value is 3000.

Logging

The JMS CSV Input and Output adapter use the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing the logging configuration is located in the `%ESP_HOME%\adapters\framework\config` directory.

Specify the location of the logging file you wish to use in the **Log4jProperty** parameter within the adapter configuration file. You can modify the logging levels within this file or the `$ESP_HOME/adapters/framework/config/log4j.properties`, which is used by default. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Starting the JMS CSV Adapter

To start the JMS CSV adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

```
cd %ESP_HOME%\cluster\nodes\node1
  %ESP_HOME%\bin\esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
%ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
%ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
%ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

UNIX:

a. Start the example cluster.

```
cd $ESP_HOME/cluster/nodes/node1
  $ESP_HOME/bin/esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
$ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

2. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: For the JMS CSV Input adapter: `cd $ESP_HOME/adapters/framework/instances/jms_csv_input` For the JMS CSV Output adapter: `cd $ESP_HOME/adapters/framework/instances/jms_csv_output` `./start_adapter.sh <adapter configuration file name>`

Operating System	Step
Windows	Open a command window and enter: For the JMS CSV Input adapter: `cd %ESP_HOME%/adapters/framework/instances/jms_csv_input` For the JMS CSV Output adapter: `cd %ESP_HOME%/adapters/framework/instances/jms_csv_output` `start_adapter.bat <adapter configuration file name>`

Stopping the JMS CSV Adapter

To stop the JMS CSV adapter from the command line, execute the **stop** command.

When you are running the adapter from the command line, stop the adapter first before stopping the project.

Operating System	Step
UNIX	Open a terminal window and enter: For the JMS CSV Input adapter: `cd $ESP_HOME/adapters/framework/instances/jms_csv_input` For the JMS CSV Output adapter: `cd $ESP_HOME/adapters/framework/instances/jms_csv_output` `./stop_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the JMS CSV Input adapter: `cd %ESP_HOME%/adapters/framework/instances/jms_csv_input` For the JMS CSV Output adapter: `cd %ESP_HOME%/adapters/framework/instances/jms_csv_output` `stop_adapter.bat <adapter configuration file name>`

JMS FIX Input Adapter

Adapter type: jms_fix_in. The JMS FIX Input adapter subscribes to messages from a JMS queue or topic, and writes these messages as stream records.

Each stream hosts FIX messages of a certain type. The adapter discards messages of any other FIX type. Most FIX fields are stored in the same order in stream columns however these fields can be stored in a different order:

- BeginString
- BodyLength
- MsgType
- CheckSum

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
FIX Version	Property ID: **fixVersion** Type: `choice` (Required) FIX Version. Valid values are: • 4.2 • 4.3 • 4.4 • 5.0 Default value is 4.2.
FIX Message Type	Property ID: **fixMessageType** Type: `string` (Required) FIX Message type. No default value.

Property Label	Description
Connection Factory	Property ID: **connectionFactory** Type: `string` (Required) Connection factory class name. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** ConnectionFactory • **TIBCO –** QueueConnectionFactory • **WebSphere MQ –** MyMQConnFactory No default value.
JNDI Context Factory	Property ID: **jndiContextFactory** Type: `string` (Required) Context factory for JNDI context initialization. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** org.apache.activemq.jndi.ActiveMQInitialContextFactory • **TIBCO –** com.tibco.tibjms.naming.TibjmsInitialContextFactory • **WebSphere MQ –** com.sun.jndi.fscontext.RefFSContextFactory No default value.

Property Label	Description
JNDI URL	Property ID: **jndiURL** Type: `string` (Required) JNDI URL. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** tcp://server:61616 • **TIBCO –** tibjmsnaming://server:7222 • **WebSphere MQ –** file:/var/mqm/jndi/ WebSphere MQ is different as it requires a separate naming server to be configured with it. By default, WebSphere MQ only provides a file-based naming server. No default value. .
Destination Type	Property ID: **destinationType** Type: `choice` (Required) Destination type. Valid values are: • QUEUE • TOPIC Default value is QUEUE.
Destination Name	Property ID: **destinationName** Type: `string` (Required) Destination name. No default value.
Subscription Mode	Property ID: **subscriptionMode** Type: `choice` (Optional) Specifies the subscription mode for TOPIC. Default value is NONDURABLE. Valid values are DURABLE and NONDURABLE.

Property Label	Description
Client ID	Property ID: **clientID** Type: `string` (Required if subscription mode is set to DURABLE) Specifies the client identifier for a JMS client. Required for creating a durable subscription in JMS. Can be any string, but must be unique for each topic. No default value.
Subscription Name	Property ID: **subscriptionName** Type: `string` (Required if subscription mode is set to DURABLE) Specifies a unique name identifying a durable subscription. Required for creating a durable subscription in JMS. Can be any string, but must be unique within a given client ID. No default value.
Batch Size	Property ID: **batchsize** Type: `uint` (Optional) Specifies number of records in a batch to commit in durable subscription mode. Default value is 1.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Date format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.

Property Label	Description
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Timestamp format. Default value is YYYY-MM-DDTHH:MM:SS.SSS .

Known limitations:

- This adapter is not a full FIX engine.
- Supports only FIX versions 4.2, 4.3, 4.4, and 5.0.
- Repeating groups and components are not supported.
- If the connection to the message broker is lost, the adapter does not attempt to reconnect.
- Supports only INSERT opcode.

See also

- *Chapter 4, Guaranteed Delivery* on page 1013

JMS FIX Output Adapter

Adapter type: jms_fix_out. The JMS FIX Output adapter publishes FIX messages to a JMS queue or topic.

Each stream hosts FIX messages of a certain type. Messages of any other FIX type are discarded. All FIX fields except the following are stored in the same order in stream columns.

- BeginString
- BodyLength
- MsgType
- CheckSum

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
FIX Version	Property ID: **fixVersion** Type: `choice` (Required) FIX Version. Valid values are: • 4.2 • 4.3 • 4.4 • 5.0 Default value is 4.2.
FIX Message Type	Property ID: **fixMessageType** Type: `string` (Required) FIX message type .
Connection Factory	Property ID: **connectionFactory** Type: `string` (Required) Connection factory class name. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** ConnectionFactory • **TIBCO –** QueueConnectionFactory • **WebSphere MQ –** MyMQConnFactory No default value.

Property Label	Description
JNDI Context Factory	Property ID: **jndiContextFactory** Type: `string` (Required) Context factory for JNDI context initialization. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** org.apache.activemq.jndi.ActiveMQInitialContextFactory • **TIBCO –** com.tibco.tibjms.naming.TibjmsInitialContextFactory • **WebSphere MQ –** com.sun.jndi.fscontext.RefFSContextFactory No default value.
JNDI URL	Property ID: **jndiURL** Type: `string` (Required) JNDI URL. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** tcp://server:61616 • **TIBCO –** tibjmsnaming://server:7222 • **WebSphere MQ –** file:/var/mqm/jndi/ WebSphere MQ is different as it requires a separate naming server to be configured with it. By default, WebSphere MQ only provides a file-based naming server. No default value.
Destination Type	Property ID: **destinationType** Type: `choice` (Required) Destination type. Valid values are: • QUEUE • TOPIC Default value is QUEUE.

Property Label	Description
Destination Name	Property ID: **destinationName** Type: `string` (Required) Destination name.
Delivery Mode	Property ID: **deliveryMode** Type: `choice` (Optional) Type of delivery mode. Valid values are: • PERSISTENT • NON_PERSISTENT Default value is PERSISTENT.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.
Column To Message Property Map	Property ID: **columnPropertyMap** Type: `string` (Advanced) Comma-delimited MyColumn=MyMessageProperty correspondence list.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Date format. Default value is YYYY-MM-DDTHH:MM:SS.SSS .

Property Label	Description
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Timestamp format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.
Runs Adapter in GD Mode	Property ID: **enableGDMode** Type: `boolean` (Advanced) If set to true, the adapter runs in guaranteed delivery (GD) mode and all GD-related parameters become required. Default value is false.
Name of Column Holding GD Key	Property ID: **gdKeyColumn** Type: `string` (Advanced) Specifies column name in the Flex operator holding the GD key. The GD key is a constantly increasing value that uniquely identifies every event regardless of the opcode in the stream of interest. No default value.
Name of Column Holding opcode	Property ID: **gdOpcodeColumn** Type: `string` (Advanced) Specifies name of column in Flex operator holding opcode. The opcode is the operation code (for example, inserts, update, or delete) of the event occurring in the stream of interest. No default value.
Name of Truncate Stream	Property ID: **gdControlStream** Type: `string` (Advanced) Specifies name of the control window in the GD setup. The control window is a source stream that informs the Flex operator of which data has been processed by the adapter and can be safely deleted. No default value.
Purge After Number of Records	Property ID: **gdPurgeInternal** Type: `int` (Advanced) Specifies number of records after which to purge the Flex operator. Default value is 1000.

Property Label	Description
Batch Size to Update Truncate Stream	Property ID: **gdBatchSize** Type: `int` (Advanced) Specifies number of records after which the control window must be updated with the latest GD key. Default value is 1000.

Known limitations:

- This adapter is not a full FIX engine.
- Supports only FIX versions 4.2, 4.3, 4.4, and 5.0.
- Repeating groups and components are not supported.
- Schema discovery is not supported.
- If the connection to the message broker is lost, the adapter does not attempt to reconnect.
- Supports only INSERT opcode.

See also

- *Chapter 4, Guaranteed Delivery* on page 1013

JMS Object Array Input and Output Adapter

The JMS Object Array Input adapter receives object array data from a JMS server and publishes it to Event Stream Processor. The JMS Object Array Output adapter takes data from Event Stream Processor, formats it into object array format, and sends it to a JMS server.

JMS Object Array Input Adapter Configuration

Configure the JMS Object Array Input adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

Transporter Module: JMS Input Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
BufferMaxSize	Type: `integer` (Advanced) Specify the capacity of the buffer queue between the transporter module and the next module. The default value is 10240.
Parameters	(Required) Section containing the **JMSInputTransporterParameters** element.
JMSInputTransporterParameters	(Required) Section containing parameters for the JMS Input transporter.
ConnectionFactory	Type: `string` (Required) Specify the connection factory class name. No default value.
JndiContextFactory	Type: `string` (Required) Specify the context factory for JNDI context initialization. No default value.

Parameter	Description
JndiUrl	Type: `string` (Required) Specify the JNDI URL. No default value.
Destination Type	Type: `string` (Required) Specify the destination type. Valid values are: QUEUE and TOPIC. The default value is QUEUE.
DestinationName	Type: `string` (Required) Specify the destination name. No default value.
MessageType	Type: `string` (Required) Specify the message type you want the JMS transporter to process. These types are supported: • TEXT - for receiving and sending messages in text string • OBJARRAY - for receiving and sending messages in custom format No default value.
SubscriptionMode	Type: `string` (Optional) Specify the subscription mode for TOPIC (see the **Destination Type** parameter). Valid values are DURABLE and NONDURABLE. Default value is NONDURABLE.
ScanDepth	Type: `integer` (Optional) Specify the depth of the schema discovery. The adapter reads the number of rows specified by this parameter value when discovering the input data schema. The default value is three.

Parameter	Description
ClientID	Type: `string` (Required for DURABLE subscription mode only) Specifies the client identifier for a JMS client. Required for creating a durable subscription in JMS. Can be any string, but must be unique for each topic. No default value. Example: `client1`.
SubscriptionName	Type: `string` (Required for DURABLE subscription mode only) Specifies a unique name identifying a durable subscription. Required for creating a durable subscription in JMS. Can be any string, but must be unique within a given client ID. No default value. Example: `subscription1`.

Formatter Module: Object List to ESP Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.

Parameter	Description
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **ObjectListToEspFormatterParameters** parameter.
ObjectListToEspFormatterParameters	(Required) Section containing the Object List to ESP formatter parameters.
DateFormat	Type: `string` (Optional) The format string for parsing date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.
TimestampFormat	Type: `string` (Optional) Format string for parsing timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

ESP Connector Module: ESP Publisher

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **EspPublisherParameters** parameter.

Parameter	Description
EspPublisherParameters	(Required) Section containing parameters for the ESP publisher.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream to which the adapter publishes data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
MaxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.

Parameter	Description
MaxPubPoolTime	Type: `positive integer` (Optional) Specifies the maximum period of time, in milliseconds, for which records are pooled before being published. If not set, pooling time is unlimited and the pooling strategy is governed by **maxPubPoolSize**. No default value.
UseTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.
SafeOps	Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT, and converts DELETE to SAFEDELETE. The default value is false.
SkipDels	Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. The default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the JMS Object List Input adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.

Parameter	Description
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.

Parameter	Description
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

JMS Object Array Input Adapter Studio Properties

Adapter type: toolkit_jms_objlist_input. Set these properties for the JMS Object Array Input adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Date Format	Propterty ID: **objectlistDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **objectlistTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.
Connection Factory	Property ID: **connectionFactory** Type: `string` (Required) Connection factory name.
JNDI Context Factory	Property ID: **jndiContextFactory** Type: `string` (Required) Context factory for JDNI context initialization.
JNDI URL	Property ID: **jdniURL** Type: `string` (Required) JDNI URL.
JMS Destination Type	Property ID: **destinationType** Type: `string` (Required) Specify the destination type. Valid values: QUEUE, TOPIC.

Property Label	Description
Destination Name	Property ID: **destinationName** Type: `string` (Required) Specify the destination name.
Subscription Mode	Property ID: **subscriptionMode** Type: `string` (Optional) Specify the subscription mode. Valid values: DURABLE, NONDURABLE.
Scan Depth	Property ID: **scanDepth** Type: `int` (Advanced) Specify the depth of the schema discovery. The adapter reads the number of rows specified by this parameter value when discovering the input data scheme.
Client ID	Property ID: **clientID** Type: `string` (Advanced) Specify the client identifier (required when durable subscription is enabled).
Subscription Name	Property ID: **subscriptionName** Type: `string` (Advanced) Specify a unique name identifying a durable subscription.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: JMS Object Array Input Adapter

Sample adapter configuration file for the JMS Object Array Input adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>jms_objlist_input</Name>
    <Description>An adapter which receives object array from JMS
server, transforms to ESP data format, and publishes to ESP stream.</
Description>
    <Log4jProperty>./log4j.properties</Log4jProperty>
    <Modules>
        <Module type="transporter">
            <InstanceName>JmsInputTransporter</InstanceName>
            <Name>JmsInputTransporter</Name>
            <Next>ObjectListToEspFormatter</Next>
            <BufferMaxSize>10240</BufferMaxSize>
            <Parameters>
                <JMSInputTransporterParameters>
                    <ConnectionFactory>ConnectionFactory</
ConnectionFactory>

<JndiContextFactory>org.apache.activemq.jndi.ActiveMQInitialContext
Factory</JndiContextFactory>
                    <JndiURL>tcp://localhost:61616</JndiURL>
                 <DestinationName>queue.array.test</DestinationName>
                    <DestinationType>QUEUE</DestinationType>
                    <MessageType>OBJARRAY</MessageType>
                </JMSInputTransporterParameters>
            </Parameters>
        </Module>

        <Module type="formatter">
            <InstanceName>ObjectListToEspFormatter</InstanceName>
            <Name>ObjectListToEspFormatter</Name>
            <Next>MyInStream_Publisher</Next>
            <Parallel>true</Parallel>
            <Parameters>
            </Parameters>
        </Module>

        <Module type="espconnector">
            <InstanceName>MyInStream_Publisher</InstanceName>
            <Name>EspPublisher</Name>
            <Parameters>
                <EspPublisherParameters>
                    <ProjectName>EspProject1</ProjectName>
                    <StreamName>BaseInput</StreamName>
                    <MaxPubPoolSize>1</MaxPubPoolSize>
                    <UseTransactions>false</UseTransactions>
                </EspPublisherParameters>
            </Parameters>
            <BufferMaxSize>10240</BufferMaxSize>
        </Module>
    </Modules>
```

```
    <EspProjects>
        <EspProject>
            <Name>EspProject1</Name>
            <Uri>esp://localhost:19011/sample_workspace/
jms_objlist_input</Uri>
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

JMS Object Array Output Adapter Configuration

Configure the JMS Object Array Output adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

ESPConnector Module: ESP Subscriber

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, espconnector.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	Section containing the **EspSubscriberParameters** parameter.
EspSubscriberParameters	(Required) Section containing parameters for the ESP subscriber.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Specifies the unique project tag of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.

Parameter	Description
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream from which the adapter subscribes to data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.

Formatter Module: ESP to Object List Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **EspToObjectListFormatterParameters** parameter.

Parameter	Description
EspToObjectListFormatterParameters	(Required) Section containing the ESP to Object List formatter parameters.
OutputAsSQLDatetimeFormat	Type: `boolean` (Optional) Specify whether the Event Stream Processor `date`, `timestamp`, and `bigdatetime` datatypes are output as `java.sql.Date` or `java.sql.Timestamp`. The default output is `java.util.Date`.

Transporter Module: JMS Output Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **JMSOutputTransporterParameters** parameter.
JMSOutputTransporterParameters	(Required) Section containing the JMS Output transporter parameters.
ConnectionFactory	Type: `string` (Required) Specify the connection factory class name. No default value.

Parameter	Description
JndiContextFactory	Type: `string` (Required) Specify the context factory for JNDI context initialization. No default value.
JndiUrl	Type: `string` (Required) Specify the JNDI URL. No default value.
Destination Type	Type: `string` (Required) Specify the destination type. Valid values are: QUEUE and TOPIC. The default value is QUEUE.
DestinationName	Type: `string` (Required) Specify the destination name. No default value.
MessageType	Type: `string` (Required) Specify the message type you want the JMS transporter to process. These types are supported: • TEXT - for receiving and sending messages in text string • OBJARRAY - for receiving and sending messages in custom format No default value.
DeliveryMode	Type: `string` (Optional) Specify the delivery mode type. Valid values are: • PERSISTENT • NON_PERSISTENT Default value is PERSISTENT.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the JMS Object Array Output adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.

Parameter	Description
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using key-store • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

JMS Object Array Output Adapter Studio Properties

Adapter type: toolkit_jms_objlist_output. Set these properties for the JMS Object Array Output adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Connection Factory	Property ID: **connectionFactory** Type: `string` (Required) Connection factory name.
JNDI Context Factory	Property ID: **jndiContextFactory** Type: `string` (Required) Context factory for JDNI context initialization.
JNDI URL	Property ID: **jdniURL** Type: `string` (Required) JDNI URL.

Property Label	Description
JMS Destination Type	Property ID: **destinationType** Type: `string` (Required) Specify the destination type. Valid values: QUEUE, TOPIC.
Destination Name	Property ID: **destinationName** Type: `string` (Required) Specify the destination name.
Delivery Mode	Property ID: **deliveryMode** Type: `string` (Optional) Specify the delivery mode. Valid values: PERSISTENT, NON_PERSISTENT.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: JMS Object Array Output Adapter

Sample adapter configuration file for the JMS Object Array Output adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>jms_objlist_output</Name>
   <Description>An adapter which converts ESP data into ObjectArray
format and sends to JMS server.</Description>
    <Log4jProperty>./log4j.properties</Log4jProperty>
    <Modules>
        <Module type="espconnector">
            <InstanceName>MyOutStream_Subscriber</InstanceName>
            <Name>EspSubscriber</Name>
            <Next>EspToObjectListFormatter</Next>
            <Parameters>
```

```
                <EspSubscriberParameters>
                    <ProjectName>EspProject2</ProjectName>
                    <StreamName>BaseOutput</StreamName>
                </EspSubscriberParameters>
            </Parameters>
        </Module>

        <Module type="formatter">
            <InstanceName>EspToObjectListFormatter</InstanceName>
            <Name>EspToObjectListFormatter</Name>
            <Next>JmsOutputTransporter</Next>
            <Parallel>true</Parallel>
            <Parameters>
            </Parameters>
        </Module>

        <Module type="transporter">
            <InstanceName>JmsOutputTransporter</InstanceName>
            <Name>JmsOutputTransporter</Name>
            <Parameters>
                <JMSOutputTransporterParameters>
                    <ConnectionFactory>ConnectionFactory</
ConnectionFactory>

<JndiContextFactory>org.apache.activemq.jndi.ActiveMQInitialContext
Factory</JndiContextFactory>
                    <JndiURL>tcp://localhost:61616</JndiURL>
                 <DestinationName>queue.array.test</DestinationName>
                    <DestinationType>QUEUE</DestinationType>
                    <MessageType>OBJARRAY</MessageType>
                </JMSOutputTransporterParameters>
            </Parameters>
        </Module>
    </Modules>


    <EspProjects>
        <EspProject>
            <Name>EspProject2</Name>
            <Uri>esp://localhost:19011/sample_workspace/
jms_objlist_output</Uri>
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Adapter Controller Parameters

The Adapter Controller port listens for commands. The `controller.xml` file is located in `%ESP_HOME%/adapters/framework/config` directory. This file is shared among all the adapter within the `%ESP_HOME%/adapters/framework/instances` directory.

Parameter	Description
ControlPort	(Required) Section containing the **MinPort** and **MaxPort** parameters.
MinPort	Type: `int` (Required) The minimum port number that can act as a control port number. The adapter framework allocates this number automatically. A recommended value is 19082.
MaxPort	Type: `int` (Optional) The maximum port number that can act as a control port number. The adapter framework allocates this number automatically. The default value is 65535.
Statistics_Update_Interval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework publishes statistics data to the ESP project to which the adapter is connected. A recommended value is 5000.
MonitorInterval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework checks the status of internal resource allocation. The framework logs a message in the log file if a resource is about to be exhausted. A recommended value is 3000.

Logging

The JMS Object Array Input and Output adapter use the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file

containing the logging configuration is located in the `%ESP_HOME%\adapters\framework\config` directory.

Specify the location of the logging file you wish to use in the **Log4jProperty** parameter within the adapter configuration file. You can modify the logging levels within this file or the `$ESP_HOME/adapters/framework/config/log4j.properties`, which is used by default. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
```

```
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Starting the JMS Object Array Adapter

To start the JMS Object Array adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

   ```
   cd %ESP_HOME%\cluster\nodes\node1
     %ESP_HOME%\bin\esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   %ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

   UNIX:

   a. Start the example cluster.

```
cd $ESP_HOME/cluster/nodes/node1
  $ESP_HOME/bin/esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
$ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

2. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: For the JMS Object Array Input adapter: `cd $ESP_HOME/adapters/framework/instances/jms_objlist_input` For the JMS Object Array Output adapter: `cd $ESP_HOME/adapters/framework/instances/jms_objlist_output` `./start_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the JMS Object Array Input adapter: `cd %ESP_HOME%/adapters/framework/instances/jms_objlist_input` For the JMS Object Array Output adapter: `cd %ESP_HOME%/adapters/framework/instances/jms_objlist_output` `start_adapter.bat <adapter configuration file name>`

Stopping the JMS Object Array Adapter

To stop the JMS Object Array adapter from the command line, execute the **stop** command.

When you are running the adapter from the command line, stop the adapter first before stopping the project.

Operating System	Step
UNIX	Open a terminal window and enter: For the JMS Object Array Input adapter: `cd $ESP_HOME/adapters/framework/instances/jms_objlist_input` For the JMS Object Array Output adapter: `cd $ESP_HOME/adapters/framework/instances/jms_objlist_output` `./stop_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the JMS Object Array Input adapter: `cd %ESP_HOME%/adapters/framework/instances/jms_objlist_input` For the JMS Object Array Output adapter: `cd %ESP_HOME%/adapters/framework/instances/jms_objlist_output` `stop_adapter.bat <adapter configuration file name>`

JMS XML Input and Output Adapter

The JMS XML Input adapter obtains XML list string messages from a JMS server and publishes them to Event Stream Processor. The JMS XML Output adapter takes XML data from Event Stream Processor, formats it to XML list format, and sends it to a JMS server.

JMS XML Input Adapter Configuration

Configure the JMS XML Input adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

Transporter Module: JMS Input Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
BufferMaxSize	Type: `integer` (Advanced) Specify the capacity of the buffer queue between the transporter module and the next module. The default value is 10240.
Parameters	(Required) Section containing the **JMSInputTransporterParameters** element.
JMSInputTransporterParameters	(Required) Section containing parameters for the JMS Input transporter.
ConnectionFactory	Type: `string` (Required) Specify the connection factory class name. No default value.
JndiContextFactory	Type: `string` (Required) Specify the context factory for JNDI context initialization. No default value.

Parameter	Description
JndiUrl	Type: `string` (Required) Specify the JNDI URL. No default value.
Destination Type	Type: `string` (Required) Specify the destination type. Valid values are: QUEUE and TOPIC. The default value is QUEUE.
DestinationName	Type: `string` (Required) Specify the destination name. No default value.
MessageType	Type: `string` (Required) Specify the message type you want the JMS transporter to process. These types are supported: • TEXT - for receiving and sending messages in text string • OBJARRAY - for receiving and sending messages in custom format No default value.
SubscriptionMode	Type: `string` (Optional) Specify the subscription mode for TOPIC (see the **Destination Type** parameter). Valid values are DURABLE and NONDURABLE. Default value is NONDURABLE.
ScanDepth	Type: `integer` (Optional) Specify the depth of the schema discovery. The adapter reads the number of rows specified by this parameter value when discovering the input data schema. The default value is three.

Parameter	Description
ClientID	Type: `string` (Required for DURABLE subscription mode only) Specifies the client identifier for a JMS client. Required for creating a durable subscription in JMS. Can be any string, but must be unique for each topic. No default value. Example: `client1`.
SubscriptionName	Type: `string` (Required for DURABLE subscription mode only) Specifies a unique name identifying a durable subscription. Required for creating a durable subscription in JMS. Can be any string, but must be unique within a given client ID. No default value. Example: `subscription1`.

Formatter Module: XML String to ESP Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.

Parameter	Description
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **XmlString-ToEspFormatterParameters** parameter.
XmlStringToEspFormatterParameters	(Required) Section containing the XML String to ESP formatter parameters.
DateFormat	Type: `string` (Optional) The format string for parsing date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.
TimestampFormat	Type: `string` (Optional) Format string for parsing timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

ESP Connector Module: ESP Publisher

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **EspPublisherParameters** parameter.

Parameter	Description
EspPublisherParameters	(Required) Section containing parameters for the ESP publisher.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream to which the adapter publishes data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
MaxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.

Parameter	Description
MaxPubPoolTime	Type: `positive integer` (Optional) Specifies the maximum period of time, in milliseconds, for which records are pooled before being published. If not set, pooling time is unlimited and the pooling strategy is governed by **maxPubPoolSize**. No default value.
UseTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.
SafeOps	Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT, and converts DELETE to SAFEDELETE. The default value is false.
SkipDels	Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. The default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the JMS XML Input adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.

Parameter	Description
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.

Parameter	Description
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

See also

- *Adapter Support for Schema Discovery* on page 1005

JMS XML Input Adapter Studio Properties

Adapter type: toolkit_jms_xmllist_input. Set these properties for the JMS XML Input adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Date Format	Propterty ID: **xmllistDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **xmllistTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.
Connection Factory	Property ID: **connectionFactory** Type: `string` (Required) Connection factory name.
JNDI Context Factory	Property ID: **jndiContextFactory** Type: `string` (Required) Context factory for JDNI context initialization.
JNDI URL	Property ID: **jdniURL** Type: `string` (Required) JDNI URL.
JMS Destination Type	Property ID: **destinationType** Type: `string` (Required) Specify the destination type. Valid values: QUEUE, TOPIC.

Property Label	Description
Destination Name	Property ID: **destinationName** Type: `string` (Required) Specify the destination name.
Subscription Mode	Property ID: **subscriptionMode** Type: `string` (Optional) Specify the subscription mode. Valid values: DURABLE, NONDURABLE.
Scan Depth	Property ID: **scanDepth** Type: `int` (Advanced) Specify the depth of the schema discovery. The adapter reads the number of rows specified by this parameter value when discovering the input data scheme.
Client ID	Property ID: **clientID** Type: `string` (Advanced) Specify the client identifier (required when durable subscription is enabled).
Subscription Name	Property ID: **subscriptionName** Type: `string` (Advanced) Specify a unique name identifying a durable subscription.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: JMS XML Input Adapter

Sample adapter configuration file for the JMS XML Input adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>jms_xmllist_input</Name>
    <Description>An adapter which receives XML list string mesage
from JMS server, transforms to ESP data format, and publishes to ESP
stream.</Description>
    <Log4jProperty>./log4j.properties</Log4jProperty>
    <Modules>
        <Module type="transporter">
            <InstanceName>JmsInputTransporter</InstanceName>
            <Name>JmsInputTransporter</Name>
            <Next>XmlStringToEspFormatter</Next>
            <BufferMaxSize>10240</BufferMaxSize>
            <Parameters>
                <JMSInputTransporterParameters>
                    <ConnectionFactory>ConnectionFactory</
ConnectionFactory>

<JndiContextFactory>org.apache.activemq.jndi.ActiveMQInitialContext
Factory</JndiContextFactory>
                    <JndiURL>tcp://localhost:61616</JndiURL>
                    <DestinationName>queue.jmsxml.test</
DestinationName>
                    <DestinationType>QUEUE</DestinationType>
                    <MessageType>TEXT</MessageType>
                    <ScanDepth>3</ScanDepth>
                </JMSInputTransporterParameters>
            </Parameters>
        </Module>

        <Module type="formatter">
            <InstanceName>XmlStringToEspFormatter</InstanceName>
            <Name>XmlStringToEspFormatter</Name>
            <Next>MyInStream_Publisher</Next>
            <Parallel>true</Parallel>
            <Parameters>
            </Parameters>
        </Module>

        <Module type="espconnector">
            <InstanceName>MyInStream_Publisher</InstanceName>
            <Name>EspPublisher</Name>
            <Parameters>
                <EspPublisherParameters>
                    <ProjectName>EspProject1</ProjectName>
                    <StreamName>BaseInput</StreamName>
                    <MaxPubPoolSize>1</MaxPubPoolSize>
                    <UseTransactions>false</UseTransactions>
                </EspPublisherParameters>
            </Parameters>
            <BufferMaxSize>10240</BufferMaxSize>
```

```
        </Module>
    </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject1</Name>
            <Uri>esp://localhost:19011/sample_workspace/
jms_xmllist_input</Uri>
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

JMS XML Output Adapter Configuration

Configure the JMS XML Output adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

ESPConnector Module: ESP Subscriber

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, espconnector.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	Section containing the **EspSubscriberParameters** parameter.
EspSubscriberParameters	(Required) Section containing parameters for the ESP subscriber.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Specifies the unique project tag of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.

Parameter	Description
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream from which the adapter subscribes to data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.

Formatter Module: ESP to XML String Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **EspToXmlStringFormatterParameters** parameter.

Parameter	Description
EspToXmlStringFormatterParameters	(Required) Section containing the ESP to XML String formatter parameters.
DateFormat	Type: `string` (Optional) The format string for date values from an ESP project. For example, `yyyy-MM-dd'T'HH:mm:ss.`
TimestampFormat	Type: `string` (Optional) Format string for timestamp values from an ESP project. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

Transporter Module: JMS Output Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **JMSOutputTransporterParameters** parameter.
JMSOutputTransporterParameters	(Required) Section containing the JMS Output transporter parameters.
ConnectionFactory	Type: `string` (Required) Specify the connection factory class name. No default value.

Parameter	Description
JndiContextFactory	Type: `string` (Required) Specify the context factory for JNDI context initialization. No default value.
JndiUrl	Type: `string` (Required) Specify the JNDI URL. No default value.
Destination Type	Type: `string` (Required) Specify the destination type. Valid values are: QUEUE and TOPIC. The default value is QUEUE.
DestinationName	Type: `string` (Required) Specify the destination name. No default value.
MessageType	Type: `string` (Required) Specify the message type you want the JMS transporter to process. These types are supported: • TEXT - for receiving and sending messages in text string • OBJARRAY - for receiving and sending messages in custom format No default value.
DeliveryMode	Type: `string` (Optional) Specify the delivery mode type. Valid values are: • PERSISTENT • NON_PERSISTENT Default value is PERSISTENT.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the JMS XML Output adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.

Parameter	Description
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using key-store • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

JMS XML Output Adapter Studio Properties

Adapter type: toolkit_jms_xmllist_output. Set these properties for the JMS XML Output adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Date Format	Propterty ID: **xmllistDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **xmllistTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.
Connection Factory	Property ID: **connectionFactory** Type: `string` (Required) Connection factory name.

Property Label	Description
JNDI Context Factory	Property ID: **jndiContextFactory** Type: `string` (Required) Context factory for JDNI context initialization.
JNDI URL	Property ID: **jdniURL** Type: `string` (Required) JDNI URL.
JMS Destination Type	Property ID: **destinationType** Type: `string` (Required) Specify the destination type. Valid values: QUEUE, TOPIC.
Destination Name	Property ID: **destinationName** Type: `string` (Required) Specify the destination name.
Delivery Mode	Property ID: **deliveryMode** Type: `string` (Optional) Specify the delivery mode. Valid values: PERSISTENT, NON_PERSISTENT.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: JMS XML Output Adapter

Sample adapter configuration file for the JMS XML Output adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>jms_xmllist_output</Name>
    <Description>An adapter which transforms ESP data to XML list
format and sends out the data to JMS server.</Description>
    <Log4jProperty>./log4j.properties</Log4jProperty>
    <Modules>
        <Module type="espconnector">
            <InstanceName>MyOutStream_Subscriber</InstanceName>
            <Name>EspSubscriber</Name>
            <Next>EspToXmlStringFormatter</Next>
            <Parameters>
                <EspSubscriberParameters>
                    <ProjectName>EspProject2</ProjectName>
                    <StreamName>BaseOutput</StreamName>
                </EspSubscriberParameters>
            </Parameters>
        </Module>

        <Module type="formatter">
            <InstanceName>EspToXmlStringFormatter</InstanceName>
            <Name>EspToXmlStringFormatter</Name>
            <Next>JmsOutputTransporter</Next>
            <Parallel>true</Parallel>
            <Parameters>
            </Parameters>
        </Module>

        <Module type="transporter">
            <InstanceName>JmsOutputTransporter</InstanceName>
            <Name>JmsOutputTransporter</Name>
            <Parameters>
                <JMSOutputTransporterParameters>
                    <ConnectionFactory>ConnectionFactory</
ConnectionFactory>

<JndiContextFactory>org.apache.activemq.jndi.ActiveMQInitialContext
Factory</JndiContextFactory>
                    <JndiURL>tcp://localhost:61616</JndiURL>
                    <DestinationName>queue.jmsxml.test</
DestinationName>
                    <DestinationType>QUEUE</DestinationType>
                    <MessageType>TEXT</MessageType>
                </JMSOutputTransporterParameters>
            </Parameters>
        </Module>
    </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject2</Name>
```

```
            <Uri>esp://localhost:19011/sample_workspace/
jms_xmllist_output</Uri>
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Adapter Controller Parameters

The Adapter Controller port listens for commands. The `controller.xml` file is located in `%ESP_HOME%/adapters/framework/config` directory. This file is shared among all the adapter within the `%ESP_HOME%/adapters/framework/instances` directory.

Parameter	Description
ControlPort	(Required) Section containing the **MinPort** and **MaxPort** parameters.
MinPort	Type: `int` (Required) The minimum port number that can act as a control port number. The adapter framework allocates this number automatically. A recommended value is 19082.
MaxPort	Type: `int` (Optional) The maximum port number that can act as a control port number. The adapter framework allocates this number automatically. The default value is 65535.
Statistics_Update_Interval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework publishes statistics data to the ESP project to which the adapter is connected. A recommended value is 5000.

Parameter	Description
MonitorInterval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework checks the status of internal resource allocation. The framework logs a message in the log file if a resource is about to be exhausted. A recommended value is 3000.

Logging

The JMS XML Input and Output adapter use the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing the logging configuration is located in the `%ESP_HOME%\adapters\framework\config` directory.

Specify the location of the logging file you wish to use in the **Log4jProperty** parameter within the adapter configuration file. You can modify the logging levels within this file or the `$ESP_HOME/adapters/framework/config/log4j.properties`, which is used by default. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Starting the JMS XML Adapter

To start the JMS XML adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

```
cd %ESP_HOME%\cluster\nodes\node1
  %ESP_HOME%\bin\esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
%ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
%ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
%ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

UNIX:

a. Start the example cluster.

```
cd $ESP_HOME/cluster/nodes/node1
  $ESP_HOME/bin/esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
$ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

2. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: For the JMS XML Input adapter: `cd $ESP_HOME/adapters/framework/instances/jms_xmllist_input` For the JMS XML Output adapter: `cd $ESP_HOME/adapters/framework/instances/jms_xmllist_output` `./start_adapter.sh <adapter configuration file name>`

Operating System	Step
Windows	Open a command window and enter: For the JMS XML Input adapter: `cd %ESP_HOME%/adapters/framework/instances/jms_xmllist_input` For the JMS XML Output adapter: `cd %ESP_HOME%/adapters/framework/instances/jms_xmllist_output` `start_adapter.bat <adapter configuration file name>`

Stopping the JMS XML Adapter

To stop the JMS XML adapter from the command line, execute the **stop** command.

When you are running the adapter from the command line, stop the adapter first before stopping the project.

Operating System	Step
UNIX	Open a terminal window and enter: For the JMS XML Input adapter: `cd $ESP_HOME/adapters/framework/instances/jms_xmllist_input` For the JMS XML Output adapter: `cd $ESP_HOME/adapters/framework/instances/jms_xmllist_output` `./stop_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the JMS XML Input adapter: `cd %ESP_HOME%/adapters/framework/instances/jms_xmllist_input` For the JMS XML Output adapter: `cd %ESP_HOME%/adapters/framework/instances/jms_xmllist_output` `stop_adapter.bat <adapter configuration file name>`

KDB Input and Output Adapter

Adapter types: KDBInput, KDBOutput. The SAP Sybase Event Stream Processor KDB Input and Output adapters allow data to be loaded from a kdb+ database into an Event Stream Processor project, and for output from an Event Stream Processor project to be stored in a kdb+ database.

Note: The KDB Input and Output adapters do not support the Solaris AMD platform.

See also

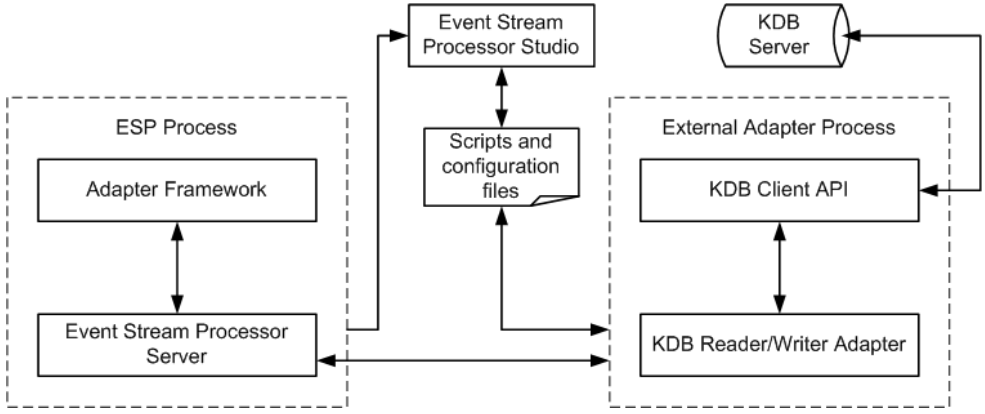
- *Adapter Support for Schema Discovery* on page 1005

Control Flow

KDB Input and Output adapters are operated by a set of script files.

The adapter scripts use these basic commands:

Figure 9: KDB Adapter Control Flow



Start Command

The **start** command starts the KDB Input or Output adapter, the KDB client API, and the KDB database server, and connects them to Event Stream Processor via the SDK interface.

Stop Command

The **stop** command stops the KDB Input or Output adapter, and closes the connection between the KDB database server and Event Stream Processor.

Datatype Mapping for the KDB Adapter

Event Stream Processor datatypes map to KDB datatypes, and KDB datatypes map to Event Stream Processor datatypes.

KDB Datatypes to ESP Datatypes

KDB datatypes map to Event Stream Processor datatypes.

KDB Datatype	Character	ESP Datatype
boolean	b	boolean
byte	x	integer
short	h	integer
int	i	integer
long	j	long
real	e	float
float	f	float
symbol	s	string
date	d	date
datetime	z	timestamp
time	t	timestamp
month	m	date
minute	u	interval
second	v	interval

ESP Datatypes to KDB Datatypes

Event Stream Processor datatypes map to KDB datatypes.

ESP Datatype	Character	KDB Datatype
integer	i	int
long	j	long
float	f	float

ESP Datatype	Character	KDB Datatype
money	f	float
string	s	symbol
date	z	datatime
timestamp	z	datatime
bigdatetime	j	long
interval	j	long
boolean	b	boolean

KDB Input Adapter

Adapter type: KDBInput. The KDB Input adapter reads kdb or kdb+tick database tables.

By default, the adapter matches the field names (in a case-insensitive manner) to determine the mapping between the source kdb+tick table and the target stream. The KDB Output adapter also supports custom field-mapping.

This adapter supports schema discovery.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
KDB Server	Property ID: **server** Type: string (Required for adapter operation and schema discovery) Specifies the name or IP address of the database server machine. Default value is localhost.
KDB Port	Property ID: **port** Type: range (Required for adapter operation and schema discovery) Specifies the IP port of the database listener. Default value is 5001. Minimum value is 0, maximum value is 65535.

Property Label	Description
Event Stream Processor User ID	Property ID: **espUser** Type: `string` (Optional) Specifies the user name for connecting to the Event Stream Processor. Default value is t.
Event Stream Processor Password	Property ID: **espPassword** Type: `password` (Optional) Specifies the password for the Event Stream Processor user ID. If you are using RSA authentication, you can leave this property blank. Default value is t.
Authentication	Property ID: **authentication** Type: `choice` (Optional) Specifies the method used to authenticate to the kdb or kdb+tick database tables. Valid values are: • `value="user_password" label="username/password"` • `value="rsa" label="RSA"` • `value="kerberos" label="Kerberos V5"` No default value.
Project URI	Property ID: **projectUri** Type: `string` (Optional) Specifies the URI to connect to a project in cluster environment. No default value.
RSA Key File	Property ID: **rsaKeyFile** Type: `filename` (Optional) Specifies the RSA private-key file name and location. No default value.

Property Label	Description
KDB User	Property ID: **user** Type: `string` (Optional for adapter operation; required only for schema discovery) Specifies the user ID for the database connection. No default value.
KDB Password	Property ID: **password** Type: `password` (Optional for adapter operation; required only for schema discovery) Specifies the password for the database connection. No default value.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.
Source Table	Property ID: **table** Type: `tables` (Advanced) Specifies the name of the source table to retrieve data from. This property supports SQL query statements.No default value.
Field Mapping	Property ID: **permutation** Type: `string` Mapping between Event Stream Processor and external fields, for example: <esp_columnname>=<database_columnname>:<esp_columnname>=<database_columnname>. No default value.

Property Label	Description
Streaming Mode	Property ID: **mode** Type: `choice` (Advanced) Specifies if the adapter should connect to a kdb +tick database and read in streaming data or execute the supplied query and feed the result to Event Stream Processor. Valid value list: • `value="stream", label ="Stream"` • `value="pull", label="One time pull"` Default value is "stream".
Polling Interval	Property ID: **pollInterval** Type: `int` (Advanced) Specifies the number of seconds to wait before reexecuting query in pull mode. If set to 0, the query is not reexecuted. Default value is 0.
Block Size	Property ID: **blockSize** Type: `int` (Advanced) Specifies the number of records to block into one pseudotransaction. Default value is 64.
Async Mode	Property ID: **async** Type: `boolean` (Advanced) If set to true, the adapter does not wait for an acknowledgement from Event Stream Processor that is received the data. Default value is false.
Encrypt Connection	Property ID: **encrypt** Type: `boolean` (Advanced) If set to true, the traffic between Event Stream Processor and the adapter is encrypted. Default value is false.

Property Label	Description
Print Debug Info	Property ID: **debug** Type: `boolean` (Advanced) Specifies if the adapter prints additional debugging information to the console. Default value is false.
Use Transaction Blocks	Property ID: **useTransaction** Type: `boolean` (Advanced) For better performance, use transaction blocks instead of envelopes while sending data to Event Stream Processor. Default value is false.
Connection Retries	Property ID: **connectionRetries** Type: `int` (Advanced) Specifies the number of times to retry connection if it breaks. Default value is 1.
Event Stream Processor Host Name	Property ID: **gatewayHost** Type: `string` (Advanced) Specifies the explicit gateway host name. No default value.
Parameter File	Property ID: **x_paramFile** Type: `filename` (Advanced) Specifies the file to write the parameters into, to pass to the external process. No default value.
Parameter File Format	Property ID: **x_paramFormat** Type: `choice` (Advanced) Specifies the format in which the external process expects the parameters. Valid value list: • `value = prop, label = "Java properties"` • `value= shell, label="Unix shell assignments"` • `value = xml, label = "Simple XML"` Default value is "prop".

Known limitations:

- If the kdb+tick databases are not running when the adapter tries to make a connection, the adapter waits indefinitely until the kdb+tick database is started.

  Note: This issue occurs only if the kdb+tick database and Event Stream Processor are running on different machines.

- If the connection to the database is broken, any updates that happen between the time the connection is broken and reestablished are lost.

KDB Output Adapter

Adapter type: KDBOutput. The KDB Output adapter publishes stream data from Event Stream Processor to a KDB+tick database table.

By default, the adapter matches the field names (in a case-insensitive manner) to decide the mapping between the source KDB+tick table and the target stream. The KDB Output adapter supports custom field-mapping.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
KDB Server	Property ID: **server** Type: `string` (Required for adapter operation and schema discovery) Name or IP address of the database server machine. No default value.
KDB Port	Property ID: **port** Type: `range` (Required for adapter operation and schema discovery) IP port of the database listener. Default value is 5001. Minimum value is 0, maximum value is 65535.
Event Stream Processor User ID	Property ID: **espUser** Type: `string` (Optional) User name for connecting to the Event Stream Processor. No default value.

Property Label	Description
Event Stream Processor Password	Property ID: **espPassword** Type: `password` (Optional) Password for the Event Stream Processor user ID. Can be empty if using RSA authentication. No default value.
Authentication	Property ID: **authentication** Type: `choice` (Optional) Authentication mechanism to use. Valid values are: • `value="user_password" label="username/password"` • `Value value="rsa" label="RSA"` • `Value value="kerberos" label="Kerberos V5"` No default value.
Project URI	Property ID: **projectUri** Type: `string` (Optional) Specifies the URI to connect to a project in cluster environment. No default value.
RSA Key File	Property ID: **rsaKeyFile** Type: `filename` (Optional) RSA private-key file name and location. No default value.
KDB User	Property ID: **user** Type: `string` (Optional for adapter operation; required only for schema discovery) User ID for the database connection. No default value.

Property Label	Description
KDB Password	Property ID: **password** Type: `password` (Optional for adapter operation; required only for schema discovery) Password for the database connection. No default value.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.
Target Table	Property ID: **table** Type: `tables` (Advanced) Name of the target table to update. No default value.
SQL Query	Property ID: **query** Type: `string` (Advanced) The SQL query to filter incoming data. No default value.
Field Mapping	Property ID: **permutation** Type: `string` Mapping between Event Stream Processor and external fields, for example: <esp_columnname>=<database_columnname>:<esp_columnname>=<database_columnname>. No default value.

Property Label	Description
Streaming Mode	Property ID: **mode** Type: `choice` (Advanced) Streaming mode. Valid values are: • `value="stream", label="user.u.upd"` • `value="push", label=”use update"` No default value.
Async Mode	Property ID: **async** Type: `boolean` (Advanced) If set to true, the adapter does not wait for an acknowledgement from Event Stream Processor that is received the data. Default value is false.
Connection Retries	Property ID: **connectionRetries** Type: `int` (Advanced) Number of times to retry connection if it breaks. Default value is 1.
KDB Batch size	Property ID: **blockSize** Type: `int` (Advanced) Maximum number of records in a single KDB write batch. Default value is 5000.
BASE Data Only	Property ID: **baseOnly** Type: `boolean` (Advanced) If set to true, does not retrieve data existing at connection time. Default value is false.
Lossy Subscriber	Property ID: **lossy** Type: `boolean` (Advanced) If set to true, the Event Stream Processor drops records if connection slows down. Default value is false.

Property Label	Description
Pulse Interval	Property ID: **pulsed** Type: `int` (Advanced) If set as a non-zero value, the subscription is created in pulsed mode with the specified period. Default value is 0.
Shine Through	Property ID: **shine** Type: `boolean` (Advanced) If set to true, subscribe send data with shine through. Default value is false.
Droppable Subscription	Property ID: **droppable** Type: `boolean` (Advanced) If set to true, Event Stream Processor drops the subscription if data is being backed up. Default value is false.
Preserve Transaction Blocks	Property ID: **originalBlocks** Type: `boolean` (Advanced) If set to true, Event Stream Processor preserves transaction boundaries in subscribed data. Default value is false.
Debug	Property ID: **debug** Type: `boolean` (Advanced) If set to true, a debug message of connection between KDB database server and Event Stream Processor is shown together with other Event Stream Processor logs. Default value is false.
Omitted Fields	Property ID: **omitFields** Type: `string` (Advanced) Comma-delimited list of KDB fields to omit (not send). No default value.

Property Label	Description
Ignored Fields	Property ID: **ignoreFields** Type: `string` (Advanced) Comma-delimited list of KDB fields to ignore, these are set to NULL. No default value.
Quiet Mode	Property ID: **quietMode** Type: `boolean` (Advanced) If true, a KDB log message is not show in standard error output. Default value is false.
Encrypt Connection	Property ID: **encrypt** Type: `boolean` (Advanced) If set to true, the traffic between Event Stream Processor and the KDB adapter is encrypted. Default value is false.
Parameter File	Property ID: **x_paramFile** Type: `filename` (Advanced) File to write the parameters into, to pass to the external process. Default value is `<temp>\PARAMETER_FILE.txt`.
Parameter File Format	Property ID: **x_paramFormat** Type: `choice` (Advanced) Format in which the external process expects the parameters. Valid values are: • `value = prop, label = "Java properties"` • `value = shell, label= "Unix shell assignments"` • `value = xml, label = "Simple XML"` Default value is “prop”.

Enabling Kerberos Authentication for the KDB Input and Output Adapters

Enable Kerberos authentication for the KDB input and output adapters by setting the necessary environment variables and specifying the **Authentication** and **espUser** parameters.

1. Set the following environment variables:
   a) Set the ESP_SERVICE_NAME environment variable to set the service principal name.
   b) Set the ESP_GSSAPI_LIB environment variable to point to the shared library provided by the Kerberos install. The library contains the GSSAPI function implementations.

      Note: If using a Kerberos library that depends on additional libraries, set the PATH environment variable for Windows or the LD_LIBRARY_PATH environment variable for Solaris and Linux.

   c) Set the KRB5CCNAME environment variable to point to the ticket cache.
   d) Set the KRB5_CONFIG environment variable to point to the configuration file used by the Kerberos library.
2. Specify the following parameters:
   - Set the **Authentication** parameter to kerberos.
   - Set the **espUser** parameter.

Log File Input Adapter

The SAP Sybase Event Stream Processor Log File Input adapter reads from a log file and sends data to a stream.

For each log record read from the file, the adapter sends one row (message) to the stream. The log file can be in any of these formats:

- Common log format
- Combined log format
- Extended common log format

The choice of formats allows you to read log files from Apache, Tomcat, IIS, and other datasources that write their log files in one of these formats. You can customize the appropriate `.properties` file to read the log files in other formats.

This adapter is written in Java and uses JavaBeans. SAP assumes, if you use this adapter, that you have extensive knowledge and experience with JavaBeans and `.properties` files.

You can use this adapter to read either live or historical log files. Historical files are complete and can be read once from beginning to end. Live files are those to which data continues to be added.

There are two types of live files, rotating and advancing. For rotating files, when the log file is full, the program that writes to the file renames the existing file and starts a new file with the original name. Typically the new file name is based on the original file name with a suffix appended, for example, if the original file name is "access.log", the renamed files are "access.log.1", "access.log.2", and so on. The Log File Input adapter always reads the original file name. The adapter starts over at the beginning when the old file has been renamed and a new one created.

For advancing files, when one file is full, the log file writer creates a new file and starts writing to that new file. The naming convention is typically a base name plus a suffix, where the suffix might be based on date/time or a sequential number (for example, "access-log.2007-01-01", "access-log.2007-01-02", and so on, or "access.log.1", "access.log.2", and so on.) Regardless of the naming convention, the adapter opens these log files in chronological order by the most recent modification date.

Configuration

Configure your Log File Input adapter by setting values in the `.properties` file for the adapter.

For example, specify the name of the log file to read by setting the **Input.Filename** property. An example `.properties` file is included in the product.

You can use this adapter to read either live or historical log files.

- To read a historical file, specify the file name in the **Input.Filename** property, and set the **Input.WaitForGrowth** property to false.
- To read a live file, whether rotated or advancing, set the **Input.WaitForGrowth** property to true. The Log File Input adapter goes to the end of the file and then reads as new data is appended to the file. When the file size shrinks to zero (after the old log file is renamed and a new, empty one is created), the Log File Input adapter continues reading from the beginning of the new file.

Properties

Lists and briefly describes some of the crucial properties in the `.properties` file. See the `example.properties` file for a complete, up-to-date list of configurable properties for the adapter.

The `example .properties` file contains commented-out examples of the column lists for the common log format and the combined log format. (See the "Parse.Class" section of the file.) To use the Extended Common log format, add column names and datatypes to the Parse.Format.Common.Columns property. Since the log file reader is extensible and modifiable, you can define your own formats, and modify the existing formats to match your configurations. For example, if your Apache server has changed the format it writes for common log format or combined log format, you can modify the properties file to match. You can either modify existing entries or create new entries. For example, you can create a new format named "MyUncommonFormat" and define the columns for that format.

Property Name	Description
Input.Filename	Type: `string` List as the name of the log file to read from.
Input.WaitForGrowth	Type: `string` If set to false, the adapter reads until it reaches the end of the file and then exits. If set to true, the adapter continues reading from the log file indefinitely at intervals of 100 milliseconds (the default interval). A value in the format milliseconds or milliseconds, nanoseconds tells the server to continue reading from the log file indefinitely at the specified interval, for example, 10 tells the adapter to read every 10 milliseconds and 0,500000 tells the adapter to check every 500000 nanoseconds, or 2000 times per second. The actual interval between reads is approximately the interval you specify; the exact interval depends upon your system's hardware and operating system and load.
Parse.Class	Type: `string` Specifies the JavaBean used to parse columns in the log file.
Parse.FormatName	Type: `string` Specifies the name of the format of the data in the log file. This may be the name of one of the predefined formats, such as common or combined, or it may be the name of a custom format.
Parse.{FormatName}.Columns (where {FormatName} is replaced with the name provided by Parse.FormatName)	Type: `string` Contains the names and datatypes of the column names in the log file. You may need to customize this property.
Parse.{FormatName}.DateColumn (where {FormatName} is replaced with the name provided by Parse.FormatName)	Type: `string` Contains the name of the date column in the log file. This value must match the actual name of your date column.
Parse.{FormatName}.TimeColumn (where {FormatName} is replaced with the name provided by Parse.FormatName)	Type: `string` (Optional) Contains the name of the time column in the log file. This column is needed only if the date column does not include the time.
Output.Uri	Type: `string` URI of the stream to send the rows to.

Property Name	Description
Output.Columns	Type: `string` The string should contain a list of column names separated by spaces. This allows you to match the output to the schema of the stream that you are writing to.
Admin.Input	Type: `string` If Admin.Input is set to stdin, then the log file adapter reads stdin looking for administrative commands. The only administrative command it currently supports is exit, which can be requested by any of the input lines: exit, quit, x, or q. The primary intent of this is as a debugging aid. The default is for no administrative request bean.
Output.AuthType	Type: `string` (Optional) Specifies authentication type. Valid values are: UserPassword, Kerberos, or ServerRSA. Default value is UserPassword.
Output.username	Type: `string` (Required) For UserPassword, this specifies the username for the server. For ServerRSA authentication method, this specifies the keystore alias.
Output.password	Type: `string` (Dependent required) Required by UserPassword authentication method. Specifies password for the Server.
Output.passwordEncrypted	Type: `string` (Dependent required) Specifies whether password in Output.password property is encrypted. Valid values are true or false. Default value is false.
Output.RSAKeyStore	Type: `string` (Dependent required) Specifies location of keystore file. Required by ServerRSA authentication method or UserPassword (if password is encrypted).
Output.RSAKeyStoreAlias	Type: `string` (Dependent required) Specifies the keystore alias. Required by UserPassword authentication method if password is encrypted.
Output.RSAKeyStorePassword	Type: `string` (Dependent required) Specifies password for the RSA keystore. Required by ServerRSA authentication method or UserPassword (if password is encrypted).

Property Name	Description
Output.KerberosKDC	Type: `string` (Dependent required) Specifies host name for the Kerberos key distribution center. Required by Kerberos authentication method.
Output.KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required by Kerberos authentication method.
Output.KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required by Kerberos authentication method.
Output.KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required by Kerberos authentication method.
Parse.BigDatetime.Format	Type: `string` Specifies format of bigdatetime column. Uses java.text.SimpleDateFormat, which does not support microseconds. To display microseconds, ensure the string UUUUUU is present. If month comes after microseconds, and the number of characters in the month are variable, the data is parsed incorrectly. For example, if the full month name is used, September has more characters than July, and July has more than May.

Note: Each input stream has a property (see the stream's Properties tab in Studio) that can specify whether to use the current server timestamp value instead of the row timestamp set by the adapter. If this stream property is set to true, it overrides any row timestamp set by the adapter.

Starting the Adapter from the Command Line

To start the Log File Input adapter from the command line, set the CLASSPATH environment variable and execute the **start** command.

To set the CLASSPATH environment variable, use `createClasspath.sh` (UNIX, Linux) or `createClasspath.bat` (Windows).

Most UNIX-like operating systems place start-up scripts in `/etc/init.d`. To start the adapter on such a system, SAP recommends that you copy the script `logfile_input.rc` to the `/etc/init.d` directory and make any necessary edits.

Note: The `logfile_input.rc` script is a sample, and works only on Linux. To use this on another platform, you must customize this script.

On systems without `/etc/init.d`, you can use `logfile_input.rc` as the basis for writing your own script to start the Log File Input adapter.

The commands that you can execute through the `logfile_input.rc` file are:

- **Start**
- **Stop**
- **Status**
- **Restart**

Attention: The `logfile_input.rc` script requires you to have write permissions to `/etc/sysconfig`, `/var/run`, and `/var/lock/sybsys`.

1. Run the command **source createClasspath.bat** for Windows or **source createClasspath.sh** for UNIX to set the CLASSPATH environment variable.
2. To run the adapter from the command line:

   ```
   java -cp $CP -Dproperties=FILE.PROPERTIES
   com.sybase.esp.adapters.logFileInput.Main
   ```

   There is no space between the "D" and the word "properties".

NYSE Technologies Input Adapter

Adapter type: wombatplugin. The Event Stream Processor adapter for NYSE Technologies MAMA (NYSE adapter) connects to the Wombat market data infrastructure.

The NYSE adapter does not support the Solaris SPARC platform.

The NYSE adapter:

- Connects to the Wombat data feed, opens sessions, and creates and signs off subscriptions.
- Translates Wombat messages into Event Stream Processor records.

The NYSE adapter requires a separately purchased license that you can obtain from the SAP Product Download Site. It does not support the standard SySAM grace period, meaning it cannot run without a valid license.

If you purchased your product from SAP or an authorized SAP reseller, go to the secure SAP Product Download Center (SPDC) at *https://sybase.subscribenet.com* and log in to generate license keys. The license generation process may vary slightly, depending on whether you ordered directly from SAP or from an SAP reseller.

If you ordered your product under an SAP contract and were directed to download from SAP Service Marketplace (SMP), you can use SMP at *http://service.sap.com/licensekeys* to generate license keys for SAP products that use SySAM 2-based licenses.

See also

- *Adapter Support for Schema Discovery* on page 1005

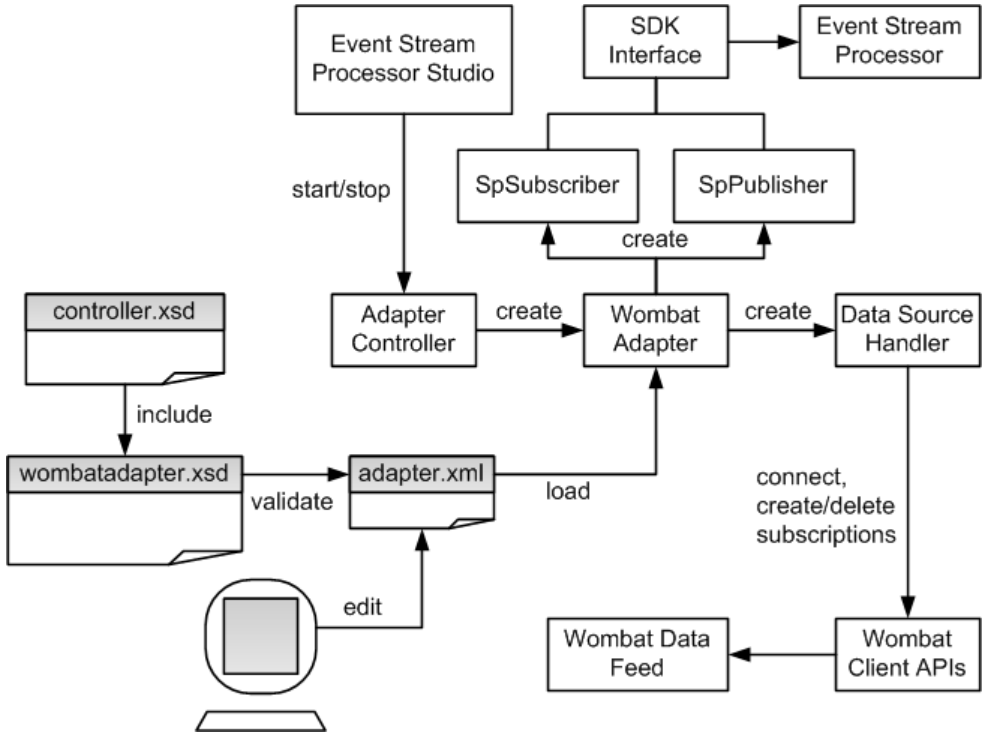
Control Flow

The adapter loads its configuration from a file (for example, `adapter.xml`) and validates it against the adapter schema (`wombatadapter.xsd`), which includes the API-wide controller schema (`controller.xsd`).

You cannot edit schemas.

The Adapter Controller creates an instance of the adapters, and receives and executes **start**, **stop**, and **status** commands.

Figure 10: NYSE Technologies Adapter Control Flow



Start Command

The **start** command configures and starts the adapter command and control interface.

The Data Source Handler, which implements the NYSE MAMA and MAMDA Client APIs, connects to the NYSE data feed, initiates a session with it, and downloads the data dictionary. The SpSubscriber and SpPublisher components connect to Event Stream Processor via the

SDK interface. SpSubscriber starts listening to the watchlists and SpPublisher is ready to publish data to data streams.

The adapter ignores the **start** command if it is executed when there is a running instance of the adapter, and sends a warning.

See also

- *Starting the NYSE Adapter* on page 463

Stop Command

The **stop** command disconnects the SpPublisher and SpSubscriber from Event Stream Processor, causes the Data Source Handler to close the session and disconnect from the datasource, stops the Adapter Controller from listening to user commands, and terminates the adapter process.

If the **stop** command is executed when there is no instance of a running adapter, the command is ignored and a warning is sent.

See also

- *Stopping the NYSE Adapter* on page 464

Status Command

The **status** command reports the adapter status, and the Adapter Controller prints out its status: either running or stopped.

See also

- *Checking the NYSE Adapter Status* on page 464

Watchlists

Dynamically control message flow through the adapter using two watchlist streams: market data and order book.

The adapter subscribes, over different transports, to various symbols from available namespaces. The symbol+namespace+transport triads are called subscription keys. The watchlists map subscription keys to data streams. The watchlist stream names are defined in the adapter configuration file.

The adapter supports group subscriptions for market data. Group subscriptions are identified by a group symbol, and each group symbol is associated with various individual symbols. Data coming on symbols from the same group are stored in the same data stream. Data stream records are keyed by individual symbols. For example, if symbols X1, X2, and X3 are associated with GROUPX, the data records are keyed using X1, X2, and X3 as opposed to GROUPX.

Subscription keys relate to streams as many to one. Subsequently, a subscription key may not target more than one market data stream and one order book stream per side (Buy or Sell).

However, a data stream may be targeted by multiple subscription keys. Order book sides (Buy and Sell) may be hosted on the same or separate streams.

Watchlist inserts and deletes are user-controlled, while updates are interpreted as error conditions. The adapter reacts to changes in watchlists as follows:

Insert	Activates subscription to symbol from specified namespace over specified transport.
Delete	Deactivates subscription.
Update	Logs an error. Sends an alert to operator (as configured). Continues to send data to old data streams. The valid way to modify the target streams is to shut down and restart the adapter.

See also

- *Watchlist Operation* on page 465
- *Insert* on page 465
- *Delete* on page 466
- *Watchlist Stream Configuration Parameters* on page 456

Market Data Watchlists

Example of market data watchlist structure and content.

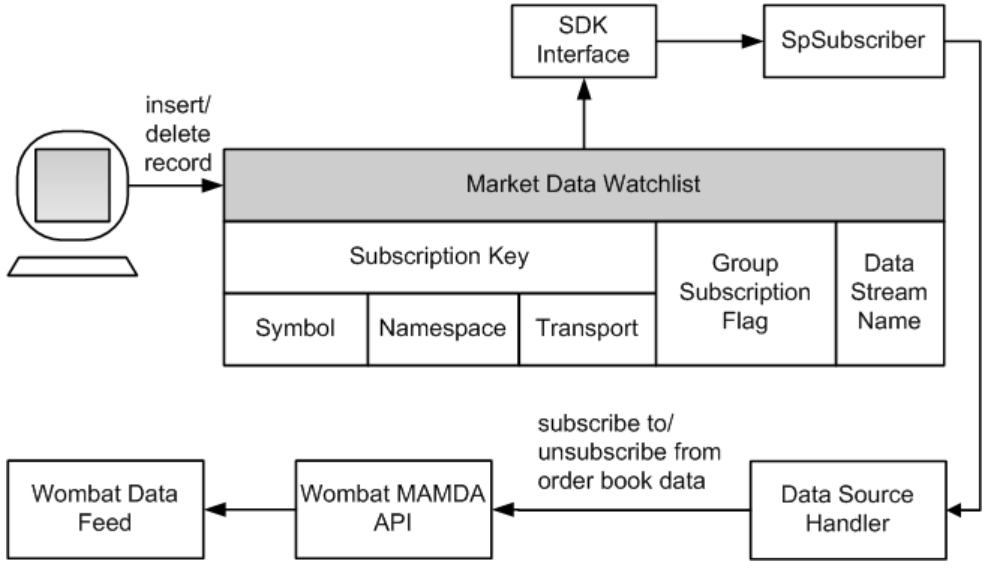


Table 1. Market Data Watchlist Structure and Sample Content

Subscription Key			Group Subscription Flag	Data Stream Name
Symbol	Namespace	Transport		
BDK	NASDAQ	T1	false	MarketB
MSFT	NYSE	T1	false	MarketA
IBM	NASDAQ	T2	false	MarketA
GROUP X	NYSE	T2	true	MarketC

See also

- *Order Book Watchlists* on page 446
- *Watchlist Stream Configuration Parameters* on page 456

Order Book Watchlists

Example of order book watchlist structure and content.

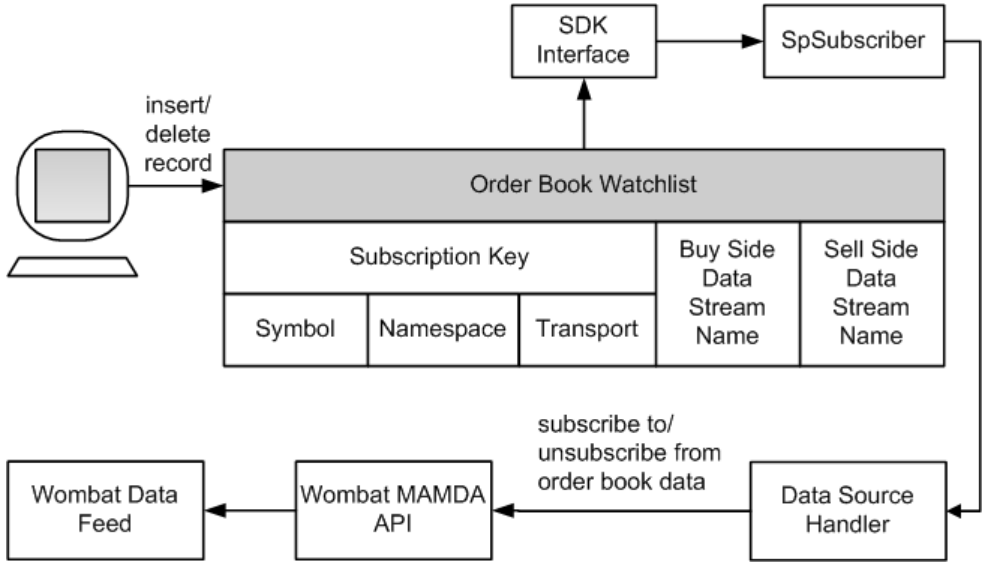


Table 2. Order Book Watchlist Structure and Sample Content

Subscription Key			Buy Side Data Stream	Sell Side Data Stream
Sym-bol	Namespace	Transport		
APPL	NASDAQ	T1	BookABuy	-
BDK	NASDAQ	T1	BookABuy	BookASell
MSFT	NYSE	T1	BookB	-
IBM	NASDAQ	T2	BookB	BookB

See also

- *Market Data Watchlists* on page 445
- *Watchlist Stream Configuration Parameters* on page 456

Data Streams

There are two types of data streams: market data and order book.

See also

- *Data Stream Configuration* on page 456

Market Data Streams

A market data stream contains the record key, one or more fields, and the stale flag.

The record key consists of the symbol, namespace (if present), and transport (if present). If you omit the namespace, the transport, or both, incoming symbol updates from different namespaces and transports are stored in the same record.

A market data stream column may have the same name as the hosted field, for example, wBidSize, wBidPrice. Custom-named columns (for example, MyTimestamp) are mapped to field names in the adapter configuration file.

Table 3. Sample Market Data Stream Content

Record Key			wBid-Size	wBid-Price	My Time-stamp	Stale
Sym-bol	Name-space	Trans-port				
MSFT	NYSE	T1	550	33.67	31--12--2008T 10:32:10.536	false

Record Key			wBid-Size	wBid-Price	My Time-stamp	Stale
Sym-bol	Name-space	Trans-port				
IBM	NASDAQ	T2	430	51.89	31--12--2008T 10:32:44.993	true
X1	NYSE	T2	850	133.63	31--12--2008T 10:27:58.563	false
X2	NYSE	T2	440	74.36	31--12--2008T 10:29:03.755	false
X3	NYSE	T2	180	21.53	31--12--2008T 10:31:55.001	false

See also

- *Data Stream Configuration* on page 456

Order Book Data Streams

An order book data stream contains the record key, entry ID, price, total size, timestamp, and stale flag.

The number of price levels is unlimited.

The record key consists of the symbol, namespace (if present), transport (if present), side indicator (if present), and price. If you omit the namespace, transport, or both, incoming symbol updates from different namespaces and transports are consolidated.

Valid values for the side indicator are B (Bid) and A (Ask). The side indicator column is mandatory if the data stream is targeted for both sides of the order book. The adapter ignores the value of the side indicator column for subscriptions that target separate buy and sell streams.

Table 4. Sample Order Book Stream Content

Record Key				EntryID	Size	Timestamp	Stale
Sym-bol	Name-space	Side	Price				
IBM	NASDAQ	B	106.23	25345	200	31--12--2008 T10:32:10.5 36	false
IBM	NASDAQ	B	106.22	74558	300	31--12--2008 T10:32:09.2 11	false

Record Key				EntryID	Size	Timestamp	Stale
Sym-bol	Name-space	Side	Price				
IBM	NASDAQ	B	106.19	12347	600	31--12--2008 T10:32:07.8 40	false
IBM	NASDAQ	A	108.73	53298	200	31--12--2008 T10:32:05.2 66	false
IBM	NASDAQ	A	108.11	53749	300	31--12--2008 T10:32:03.7 54	false
MSFT	NYSE	-	55.93	65228	400	31--12--2008 T10:31:53.9 22	false
MSFT	NYSE	-	55.87	54349	700	31--12--2008 T10:31:46.7 25	false

Note: Transport is ignored when storing data records. Also, the side indicator is empty when its value is derived from the watchlist.

See also

- *Data Stream Configuration* on page 456

Stale Records

Data stream records are marked stale when certain adapter functions fail.

When the adapter starts, it sends finalization instructions to Event Stream Processor. These instructions are run if communication with the adapter is lost or if the adapter fails to deliver two consecutive heartbeats. The finalization procedure sets the stale flag to 1 (true) in all records of all configured adapter data streams.

Data stream records are also marked stale if:

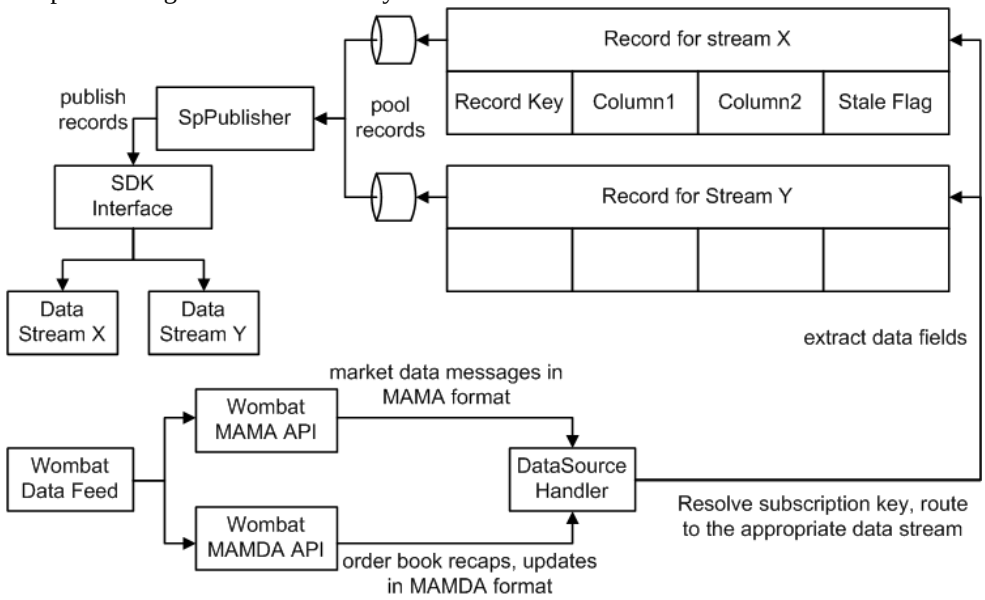
- A subscription is deactivated.
- The adapter receives a market data gap message.
- The adapter receives an order book gap message.
- The adapter receives a transport disconnect message from the NYSE data feed.
- The adapter is about to shut down due to a user command or fatal error condition.

See also

- *Event Stream Processor Parameters* on page 453

Message Flow

Adapter message flow is initiated by the **start** command.



The Data Source Handler receives real-time market data messages in MAMA format. It receives order book recaps and updates, such as deltas and complex deltas, in MAMDA format. It resolves the subscription key and routes the message to the appropriate data stream, where the data fields extract and convert into stream records.

A record is then ready to be published to Event Stream Processor but is not published immediately. Records are queued, then picked up by SpPublisher. Set the queue capacity in the adapter configuration file. A larger queue is less likely to overflow if a message burst occurs. When the queue is three-quarters full, a warning is logged. Another warning is logged when the queue returns to three-quarters empty. If the queue is full, the adapter waits until room becomes available before it places the next record.

The adapter uses record pooling for performance considerations. The dequeued records are pooled according to user preferences in the adapter configuration file. Messages are published when the pool size reaches the Maximum Pool Size or after a Maximum Pooling Time since the previous publication, whichever occurs first. The Maximum Pooling Time drives the adapter's latency. If the Maximum Pooling Time is too short or Maximum Pool Size is too small, messages are published to Event Stream Processor in undersized batches, resulting in poor overall performance.

When a pooled record batch is ready for publication, the SpPublisher uses the Pub/Sub API to send the records to Event Stream Processor. Records are published asynchronously. The

adapter receives no feedback from Event Stream Processor. In the event of a failover, the Pub/Sub API switches to the spare Event Stream Processor instance without message loss.

Datatype Mapping for the NYSE Adapter

Event Stream Processor datatypes map to NYSE datatypes.

Event Stream Processor Datatype	MAMA API Datatype
`integer`	`long`
`long`	`long`
`float, money, money1-money15`	`double`
`string`	`string`
`bigdatetime`	`com.wombat.mama.MamaDateTime`
`interval`	`com.wombat.mama.MamaDateTime`
`binary`	`string`

Setting the JAVA_HOME Environment Variable

Set the JAVA_HOME environment variable to point to the Java directory.

Prerequisites

- Install Java Runtime Environment version 1.7.0_1 or higher. To see if you have a suitable version of Java installed, go to *http://www.java.com/en/download/installed.jsp*.
- Place NYSE Java and binary libraries under `$ESP_HOME/adapters/wombat/lib/wombat`.

Task

Set the JAVA_HOME environment variable to the directory path where Java Runtime Environment 1.7.0_1 or higher is installed.

Next

Verify that the ESP_HOME environment variable is set correctly.

Configuration

Configuration information for the NYSE Technologies adapter.

NYSE Adapter Directory

The adapter directory contains all files, such as configuration files, templates, examples, and JAR files, relating to the adapter.

```
README.txt  Quick Guide
ReleaseNotes.txt  Release Notes

bin/
    start_adapter.bat  Standalone adapter startup script
    start_adapter.sh      Standalone adapter startup script
    adapter-plugin.bat Plug-in connector startup script
    adapter-plugin.sh  Plug-in connector startup script

config/
    controller.xsd     Controller schema
    log4j.properties   Sample logging configuration
    wombatadapter.xsd  Adapter schema
    dictionary.txt     The configuration file to map wombat fields
with ESP datatypes. This is used for data dictionary.
    login.config  Authentication configuration

discovery/        Data discovery templates

examples/        Working example

lib/
    wombat/        wombat java and binary libraries

libj/

    commons-codec-1.3.jar Required by SDK API
    commons-collections-3.2.1.jar
    commons-configuration-1.6.jar
    commons-lang-2.6.jar
    commons-logging-1.1.jar Logging library
    esp_adapter_api.jar Adapter API code
    esp_adapter_wombat.jar  Wombat adapter library
    esp_i18n.jar
    esp_license.jar
    esp_sdk.jar ESP SDK library
    log4j-1.2.16.jar Logging library
    postgresql.jar
    sylapi.jar
    ws-commons-util-1.0.2.jar Required by ESP SDK
    xerces-impl-2.9.1.jar XML parser library
    xmlrpc-client-3.1.3.jar Required by ESP SDK
    xmlrpc-common-3.1.3.jar Required by ESP SDK
```

Schema and Configuration File

The adapter configuration is loaded from a file and validated against the adapter schema.

The adapter working example includes a sample `adapter.xml` file. You can edit this file or write a new one. Ensure that the adapter configuration validates against the schema. You see an error message if the configuration does not validate.

Adapter Controller Parameter

The **controllerPort** parameter specifies the adapter command and control port.

Parameter Name	Description
controllerPort	Type: `positive integer` (Required) Specifies the adapter command and control port. User commands are sent to this port on localhost.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the NYSE adapter.

These parameters are defined in the `controller.xsd` file in the `config` directory.

Parameter Name	Description
espAuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **espAuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
espUser	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **espAuthType**). No default value.

Parameter Name	Description
espPassword	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **espPassword** value is encrypted. Default value is false. If set to true, the password value is decrypted using **espRSAKeyStore** and **espRSAKeyStorePassword**.
espProjectUri	Type: `string` (Required) Specifies the total project URI to connect to Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
pulseInterval	Type: `non-negative integer` (Optional) Specifies the time interval, in seconds, during which outbound record changes are coalesced by Event Stream Processor, then received by the adapter as a single event. If not set or set to 0, record changes are received individually as they occur.
espHeartbeatPeriod	Type: `positive integer` (Optional) Specifies the length of time, in seconds, that adapter waits before sending the next heartbeat to Event Stream Processor. If Event Stream Processor fails to receive two consecutive heartbeats, all records published by the adapter are marked stale. The default value is 10.
recordQueueCapacity	Type: `positive integer` (Optional) Specifies capacity of the record queues. Default value is 4096.

Parameter Name	Description
maxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.
maxPubPoolTime	Type: `positive integer` (Optional) Specifies the maximum period of time, in milliseconds, for which records are pooled before being published. If not set, pooling time is unlimited and the pooling strategy is governed by **maxPubPoolSize**. No default value.
useTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.
espRSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **espAuthType** is set to server_rsa, or the encrypted attribute for **espPassword** is set to true, or both.
espRSAKeyStorePassword	Type: `string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **espAuthType** is set to server_rsa, or the encrypted attribute for **espPassword** is set to true, or both.
espKerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **espAuthType** is set to kerberos.
espKerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **espAuthType** is set to kerberos.

Parameter Name	Description
espKerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **espAuthType** is set to kerberos.
espKerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **espAuthType** is set to kerberos.
espEncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **espPassword** is set to true. If left blank, RSA is used as default.

See also

- *Stale Records* on page 449

Watchlist Stream Configuration Parameters

Watchlist stream configuration parameters specify the names of the market data and order book watchlists.

Parameter Name	Description
marketDataWatchlist	Type: `string` (Required) Specifies name of the market data watchlist.
orderBookWatchlist	Type: `string` (Required) Specifies name of the order book watchlist.

See also

- *Market Data Watchlists* on page 445
- *Order Book Watchlists* on page 446
- *Watchlists* on page 444

Data Stream Configuration

Use the `marketDataStreams` section in the configuration file to provide data stream parameters.

Indicate the stream name for each data stream.

Data stream columns and the corresponding MAMA fields may have the same or different names. If the names are different, explicitly map the column and its corresponding data field. In the example below, the MyTimestamp column is mapped to the wSrcTime MAMA field.

```
<column>
<name>MyTimestamp</name>
<field>wSrcTime</field>
</column>
```

Ensure columns have the same datatype as their corresponding fields. Some columns may correspond to no field. Column names Symbol, Namespace, Transport, and Stale are reserved.

See also

- *Data Streams* on page 447
- *Market Data Streams* on page 447
- *Order Book Data Streams* on page 448

Datafeed Parameters

Datafeed parameters configure the datafeed for the NYSE adapter.

Refer to the *MAMA Developer's Guide* for detailed information about these parameters.

Parameter Name	Description
middleware	Type: `string` (Required) Specifies the name of the middleware API. Currently only Configuration 9 wmw (NYSE TCP Middleware) is supported. Default value is wmv or lbm.
subscriptionTimeout	Type: `positive integer` (Required) Specifies number of seconds the adapter waits, without receiving an initial value, before it resends a market data subscription request.
subscriptionRetries	Type: `positive integer` (Required) Specifies the number of attempts to make to obtain an initial image for a subscription.
dictionaryTransport	Type: `string` (Required) Specifies transport over which the MAMA dictionary is requested on adapter start-up.
dictionaryNamespace	Type: `string` (Required) Specifies namespace from which the MAMA dictionary is requested on adapter start-up.

Parameter Name	Description
dictionaryTimeout	Type: `positive integer` (Required) Specifies the number of seconds the adapter waits, without receiving a response, before it resends a MA-MA dictionary request.
dictionaryRetries	Type: `positive integer` (Required) Specifies the number of attempts to make to obtain the MAMA dictionary.

Sample NYSE Configuration File

Sample configuration file (`adapter.xml`) for the NYSE adapter.

This file is in the `example` folder.

```
<adapter>

<!-- Adapter Controller -->
 <controller>
  <controllerPort>13579</controllerPort>
 </controller>

<!-- Event Stream Processor Settings -->
 <esp>
   <espConnection>
        <espProjectUri>esp://localhost:19011/w1/p1</espProjectUri>
   </espConnection>

   <espSecurity>
        <espUser>espuser</espUser>
        <espPassword encrypted="false">espuser</espPassword>
        <espAuthType>none</espAuthType>
<!--
        <espRSAKeyStore>/keystore/keystore.jks</espRSAKeyStore>
       <espRSAKeyStorePassword>Sybase123</espRSAKeyStorePassword> --
>
        <espEncryptionAlgorithm>RSA</espEncryptionAlgorithm>
   </espSecurity>
  <maxPubPoolSize>1</maxPubPoolSize>
 </esp>

 <watchlists>
  <marketDataWatchlist>MarketDataWatchlist</marketDataWatchlist>
  <orderBookWatchlist>OrderBookWatchlist</orderBookWatchlist>
 </watchlists>

 <marketDataStreams>
  <stream>
   <name>MyMarketDataStream</name>
   <column>
```

```
    <name>MyTimestamp</name>
    <field>wSrcTime</field>
   </column>
  </stream>
 </marketDataStreams>

<datafeed>
 <middleware>wmw</middleware>
 <subscriptionTimeout>5</subscriptionTimeout>
 <subscriptionRetries>1</subscriptionRetries>
 <dictionaryTransport>demo</dictionaryTransport>
 <dictionaryNamespace>WOMBAT</dictionaryNamespace>
 <dictionaryTimeout>10</dictionaryTimeout>
 <dictionaryRetries>1</dictionaryRetries>
</datafeed>

</adapter>
```

NYSE Input Adapter

The NYSE Input adapter connects to a NYSE data feed to receive real-time level 1 and level 2 market data.

You can configure the adapter on any source stream as an input data location. The authentication method is set to that of Event Stream Processor: rsa, kerberos, or Native OS (user name/password). This adapter supports schema discovery.

To use this adapter, ensure NYSE adapter version 1 or later is installed.

Property Label	Description
Connector Directory Path	Property ID: **baseDir** Type: `directory` (Required) Specify the path to the adapter installation directory. This property is ignored if the **Connector Remote Directory Path** property is supplied. Default value is `/mydir/NYSEAdapter`. Use a forward slash for both UNIX and Windows paths.
Configuration File Path	Property ID: **configFilePath** Type: `configFilename` (Required) Specify the absolute path to the adapter configuration file. This property is ignored if the **Remote Configuration File Path** property is supplied. Default value is `/mydir/NYSEAdapter/config/adapter.xml`. Use a forward slash for both UNIX and Windows paths.

Property Label	Description
Discovery Directory Path	Property ID: **discDirPath** Type: `directory` (Required) Specify the absolute path to the adapter discovery directory. Default value is `/mydir/NYSEAdapter/discovery`. Use a forward slash for both UNIX and Windows paths.
Connector Remote Directory Path	Property ID: **remoteBaseDir** Type: `string` (Advanced) Specify the path to the adapter remote base directory (for remote execution only). If this property is supplied, the **Connector Directory Path** property is ignored. Use a forward slash for both UNIX and Windows paths.
Remote Configuration File Path	Property ID: **remoteConfigFilePath** Type: `string` (Advanced) Specify the absolute path to the adapter configuration file (for remote execution only). If this property is supplied, the **Configuration File Path** property is ignored. Use a forward slash for both UNIX and Windows paths.
Discovered Table (runtime)	Property ID: **table** Type: `tables` (Advanced) Name of the discovered table. This is filled in by Studio.

Property Label	Description
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Logging

The NYSE Technologies Input adapter uses the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing the logging configuration is part of the NYSE Technologies Input adapter distribution.

You can modify the logging levels of the `log4j.properties` configuration file which is located in the `%ESP_HOME%\adapters\<adapter-name>\config` directory. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Operation

Operate the NYSE adapter from the command line.

Ensure the project to be run contains the market data and order book watchlists. Check that the names of the watchlist streams correspond to the **marketDataWatchlist** and **orderBookWatchlist** parameters respectively.

Set the desired logging level in `log4j.properties`.

Starting the NYSE Adapter

To start the NYSE adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

   ```
   cd %ESP_HOME%\cluster\nodes\node1
     %ESP_HOME%\bin\esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   %ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

   UNIX:

   a. Start the example cluster.

   ```
   cd $ESP_HOME/cluster/nodes/node1
     $ESP_HOME/bin/esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   $ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   $ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   $ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

2. Start the adapter.

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/wombat/bin`
	`./adapter.sh <configuration file path> start`

Operating System	Step

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/wombat/bin`
	`adapter.bat <configuration file path> start`

You can use the **esp_subscribe** utility to ensure that NYSE messages are successfully published to Event Stream Processor.

See also

- *Start Command* on page 443

Checking the NYSE Adapter Status

To check the NYSE adapter status from the command line, execute the **status** command.

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/wombat/bin`
	`./adapter.sh <configuration file path> status`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/wombat/bin`
	`adapter.bat <configuration file path> status`

You see the adapter status: running or stopped.

See also

- *Status Command* on page 444

Stopping the NYSE Adapter

To stop the NYSE adapter from the command line, execute the **stop** command.

Prerequisites

When you are running the adapter from the command line, stop the adapter first before stopping the project.

Task

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/wombat/bin`
	`./adapter.sh <configuration file path> stop`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/wombat/bin`
	`adapter.bat <configuration file path> stop`

See also

- *Stop Command* on page 444

Watchlist Operation

You can use inserts and deletes to modify watchlists.

Watchlist updates are interpreted as error conditions and no action is taken.

Modifying the market data watchlist causes the adapter to subscribe to or unsubscribe from real-time data on symbols. Modifying the order book watchlist causes the adapter to subscribe to or unsubscribe from order book data on symbols.

See also

- *Watchlists* on page 444

Insert

A watchlist insert triggers two actions in the adapter: subscribing and publishing.

- The adapter subscribes to the specified symbol from the specified namespace over the specified transport.
- The adapter receives real-time market data messages or order book recaps and updates, and publishes them to the corresponding data streams.

See also

- *Watchlists* on page 444

Delete
A watchlist delete triggers two actions in the adapter: unsubscribing and marking records stale.

- The adapter unsubscribes the specified symbol from the specified namespace over the specified transport.
- Market data stream records that result from the canceled subscription are marked stale.

See also

- *Watchlists* on page 444

Example: Subscribing to and Publishing Data

Subscribe to real-time market data on two symbols and order book data on one symbol, and publish the incoming data to Event Stream Processor.

Prerequisites

A network connection to the NYSE datafeed.

Task

1. Set the username and password in the example environment:

Operating System	Step
UNIX	**1.** Edit the `set_example_env.sh` script **2.** Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	**1.** Edit the `set_example_env.bat` script **2.** Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

2. Edit the `start_adapter.sh` script.
3. Set the JAVA_HOME environment variable to the directory where Java Runtime Environment (JRE) is installed.

   Note: The NYSE libraries are available in both 32- and 64-bit versions. If your libraries are 32-bit, use a 32-bit JRE. If your libraries are 64-bit, use a 64-bit JRE.
4. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

5. Edit the `mama.properties` file in the adapter `lib/wombat` directory to ensure the `subscribe_address` and `subscribe_port` properties point to a NYSE data feed.
6. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: `start_adapter.sh`
Windows	Open a command window and enter: `start_adapter.bat`

7. Wait five to ten seconds for the adapter to initialize.
8. Upload a stream record.

Operating System	Step
UNIX	`./upload.sh`
Windows	`upload.bat`

9. Start the subscriber utility to view data stream content.

Operating System	Step
UNIX	`./esp-subscribe.sh`
Windows	`esp-subscribe.bat`

ODBC Driver for SAP Sybase Event Stream Processor

The ODBC driver allows you to access and view ESP content within 32-bit Windows, 64-bit Windows, and Linux environments.

The ESP ODBC driver is installed by default. It is included in the ESP distributions for 32-bit Windows, 64-bit Windows, and Linux operating systems. For Windows, the binaries are placed in the *%ESP_HOME%*\obdc folder. For Linux, the binaries are placed in the *$ESP_HOME*/odbc folder.

For Linux, the ESP ODBC driver must be configured using the unixODBC 2.3.0 driver manager, which must be downloaded and installed manually.

Event Stream Processor ODBC Driver for Windows

The ESP ODBC driver for Windows allows you to access and view ESP content using 32-bit or 64-bit external applications within the corresponding Windows platform.

If you run the Windows 32-bit installer, only the 32-bit ODBC driver is included in the %ESP_HOME%\odbc folder. If you run the 64-bit installer, the 32-bit ODBC driver is included in the %ESP_HOME%\odbc folder, as well as the 64-bit ODBC driver under %ESP_HOME%\odbc\odbc64. Using this driver, C/C++ programs and external applications can send SQL queries or statements to ESP. For example, Microsoft Excel® can pull data from ESP into a spreadsheet.

Configuring Data Source Names

Configuring a Data Source Name (DSN) on a Windows computer makes that DSN available for use by third-party tools.

1. Start the ODBC Data Source Administrator utility:
   - If running on Windows 64-bit, this will bring up the 64-bit ODBC Data Source Administrator utility, which can only be used to manage datasources for 64-bit ODBC drivers.
   - To add a datasource for the 32-bit ESO ODBC driver on Windows 64-bit, you need to bring up the 32-bit ODBC DATA Source Administrator utility by running the odbcad32.exe that is under C:\Windows\SysWOW64.

   a) Select **Windows Control Panel** > **Adminstrative Tools** > **Data Sources (ODBC)**.

   The **ODBC Data Source Administrator** menu is displayed.
2. Click **Add**.
   The **Create New Data Source** menu is displayed.
3. Select **Sybase ESP ODBC Driver** and click **Finish**.
   The **SYBASE ODBC DRIVER ANSI** screen is displayed.

4. In the **Data Source** field, enter the name you wish to use for this data source.
5. In the **Ws/Proj[/Auth]** field, enter the name of the workspace followed by a slash and the name of the project. If you are using Kerberos authentication, add `/krb` after the project name. If you are using RSA authentication, add `/rsa/` followed by the full path (such as `C:/home/mykey.private`) of your RSA key file.
6. In the **Server** field, enter the name of the node in the cluster on which the specified project is running.
7. In the **User Name** field, enter the loginid that you use when accessing that node.
8. In the **Description** field, describe the data available from this source.
9. In the **Ssl Mode** field, enter `require` if the cluster uses SSL or `disable` if it doesn't.
10. In the **Port** field, enter the port to use when communicating with the specified node.
11. In the **Password** field, enter the password for the userid you use when accessing the specified node.
12. Click **OK** to dismiss that screen and click **OK** again to dismiss the **ODBC Data Source Administrator** screen.

Connecting to ESP Using the ODBC Driver

Follow the procedures specific to the third-party application from which you wish to connect to ESP using the ODBC Driver for ESP.

The ODBC Driver for ESP is a general purpose utility. Once you have defined a data source name (DSN) for the instance of ESP to which you wish to connect, you can use any application you choose.

Note: You must have a DSN defined before using this procedure to connect to ESP using the ODBC driver.

The specific procedure for making the connection will depend on the application you use. The example shown here makes a connection from Microsoft Excel to pull data from ESP into a spreadsheet.

1. Start the ESP project to which you want to connect, if it is not already running.
2. Open Excel®.
3. Click **Data > From Other Sources > From Microsoft Query**
4. Select **Data Source**.
   A list of the DSNs configured on your system is displayed.
5. Select the DSN for the instance of ESP to which you wish to connect, and click **OK**.
   The **Query Wizard - Choose Columns** screen is displayed.
6. Select the name of the stream you want from the list in the left pane of the screen, and click on the button between the two panes.
   The columns in your stream are listed in the right pane of the screen.
7. Click **Next**.

The **Query Wizard - Filter Next** screen is displayed.

8. Click **Next**.
   The **Query Wizard - Sort Order** screen is displayed.
9. Click **Next**.
   The **Query Wizard - Finish** screen is displayed.
10. Click **Finish**.
    The **Import Data** screen is displayed.
11. Click **OK**.
    The Excel spreadsheet is redisplayed, with data from ESP populating it.

Event Stream Processor ODBC Driver for Linux

The ESP ODBC driver for Linux allows you to access and view ESP content within external applications running on a Linux machine.

The ESP ODBC driver for Linux operates only on a Linux machine; however, it connects to any ESP on any platform. It allows external applications, such as SAP HANA, to send SQL queries or statements to ESP.

Connect to ESP using the Linux ODBC Driver

Allows you to access ESP content by connecting to ESP using Linux ODBC.

Prerequisites

- Install unixODBC 2.3.0 driver on the Linux machine.

Task

Connect to the Linux ESP ODBC driver.

1. Set *LD_LIBRARY_PATH* to `<unixODBC2.3>/lib:$ESP_HOME/odbc`
2. Set the environment variable *PATH* to `<unixODBC2.3>/bin`
3. Configure a data source name (DSN) by defining the DSN parameters:

Table 5. DSN Parameters

Property	Description
Servername	Property ID: **Servername** Type: `string` (Required) Cluster host name.

Property	Description
Driver	Property ID: **Driver** Type: `string` (Required) Path to the ESP ODBC Driver.
Port	Property ID: **Port** Type: `integer` (Required) Cluster port.
Database	Property ID: **Database** Type: `string` (Required) String specifying project and authentication. For example, /ws1/project1/user /ws1/project1/rsa/<rsakeyfile> /ws1/project1/krb
Username	Property ID: **Username** Type: `string` (Required) Cluster login user.
Password	Property ID: **Password** Type: `string` (Required) Cluster login password.
SSLMode	Property ID: **SSLMode** Type: `Enum 'enable | disable'` (Optional) Cluster SSL Mode.

4. Start an ESP project and enter some data in the project stream or window.
5. Use isql or iusql (for Unicode) to run an SQL statement to extract the data from ESP. For example, here Trades is a stream or window in the ESP project that you started:

```
isql -v myesp
```

```
Sql > select * from Trades
```

Now you can access and read data from an ESP project running on a Linux machine.

Open Input and Output Adapter

The SAP Sybase Event Stream Processor Open adapter is a version of the open-source openadapter™ (openadapter.org).

A range of adapters is available for common applications and middleware environments, such as Web services and various file types. You can use each adapter with a variety of readers and writers to parse and format different types of messages (for example, delimited field records or XML documents). The records coming in through the adapter can include an ESP_OPS column that indicates the database operation to perform with the record.

- i, I, insert, or INSERT indicates an insert.
- p, P, upsert, or UPSERT indicates an upsert.
- u, U, update, or UPDATE indicates an update.
- s, S, safedelete, or SAFEDELETE indicates a safedelete.
- d, D, delete, or DELETE indicates a delete.

If you use the ESP_OPS column, ensure every record in this column has a value.

An Open adapter is defined by an adapter properties file, and includes a number of components that move data from one or more source components to one or more sink components. You may also configure intermediate components (pipes) to perform additional processing on the data. In a system where a number of possible adapters can run, each adapter runs as a separate instance that you start and control individually.

Note: On Microsoft Windows, use a forward slash as a separator in paths, class paths, and URLs.

The Open adapter requires a separately purchased license that you can obtain from the SAP Product Download Site. It supports the standard SySAM grace period, meaning it can run unlicensed for 30 days. After this period, the adapter cannot run without a valid license.

If you purchased your product from SAP or an authorized SAP reseller, go to the secure SAP Product Download Center (SPDC) at *https://sybase.subscribenet.com* and log in to generate license keys. The license generation process may vary slightly, depending on whether you ordered directly from SAP or from an SAP reseller.

If you ordered your product under an SAP contract and were directed to download from SAP Service Marketplace (SMP), you can use SMP at *http://service.sap.com/licensekeys* to generate license keys for SAP products that use SySAM 2-based licenses.

Tips for Migrating Your Open Adapter Scripts

Tips for migrating your Open Adapter scripts from Aleri to Event Stream Processor.

When migrating your scripts from Aleri to Event Stream Processor:

- Set the $ESP_ADAPTER_HOME environment variable to `<ESP_HOME>\adapters\open_adaptor`.
- Add `-Djava.library.path=$ESP_ADAPTER_HOME/lib` to the Java command when you invoke the adapter. For example:

```
$JAVA_HOME/bin/java
$AUTH -Djava.library.path=$ESP_ADAPTER_HOME/lib -cp $CP
org.openadaptor.adaptor.RunAdaptor fileToAsap.props adaptor
```

- In the AsapSource, SpPersistentSubscriberSource, and AsapSink property files, enable authentication by setting (only) one of these three properties to true:
  - `UseUserPassword`
  - `UseKerberos`
  - `UseServerRSA`

  See *AsapSource Properties*, *SpPersistentSubscriberSource Properties* , and *AsapSink Properties* for additional details on these properties.

See also

- *AsapSource Properties* on page 477
- *SpPersistentSubscribeSource Properties* on page 480
- *AsapSink Properties* on page 482

Datatype Mapping for the Open Adapter

Event Stream Processor datatypes map to Open adapter datatypes.

Event Stream Processor Datatype	Open Adapter Datatype
integer	integer
long	long
float	double
date	datetime
timestamp	datetime
bigdatetime	long
boolean	short
interval	long
binary	string
money	double

Event Stream Processor Datatype	Open Adapter Datatype
`money(1-n)`	`double`
`string`	`string`

Setting the JAVA_HOME Environment Variable

Set the JAVA_HOME environment variable to point to the Java directory.

Prerequisites

Install Java Runtime Environment version 1.7.0_1 or higher. To see if you have a suitable version of Java installed, go to *http://www.java.com/en/download/installed.jsp*.

Task

Set the JAVA_HOME environment variable to the directory path where Java Runtime Environment 1.7.0_1 or higher is installed.

Next

Verify that the ESP_HOME environment variable is set correctly.

Configuration

The adapter properties (`.props`) files are text files that contain configuration information for the components to be invoked for an adapter.

You can create properties files by using a text editor or the Adapter Framework Editor. A configuration can contain any number of source, sink, and pipe components, and their respective readers and formatters. The SAP Open adapter can also read properties from XML documents.

Each adapter properties file contains the configuration for a single adapter. An adapter property specifies the adapter, component, and property names:

```
AdapterName. ComponentName.PropertyName=PropertyValue
```

For example:

A # character in the first column denotes a comment line.

```
#
# Adaptor 'Dynamic2' Component 'BalanceInMQ
#
Dynamic2.BalanceInMQ.QueueManager=QM_Test
Dynamic2.BalanceInMQ.Queue=TEST.BALANCE.IN … #
# Adaptor 'Dynamic2' Component 'EventInMQ'
#
Dynamic2.EventInMQ.QueueManager=QM_Tes t
Dynamic2.EventInMQ.Queue=TEST.EVENT.IN
```

Properties may be qualified to more levels. Ensure that properties that define a field are defined for both the field name and the field type. For example:

```
Dynamic2.EventInMQ.Field1=Date
Dynamic2.EventInMQ.Date.Name=CurrentDate Dynamic2.
EventInMQ.Date.Type=Datetime
```

The Open adapter supports properties for individual components. You can use a number of statements in the adapter properties file to simplify the definition of properties.

Open Adapter Directory

The adapter directory contains all files, such as configuration files, templates, examples, and JAR files, relating to the adapter.

```
$ESP_HOME/adapters/open_adaptor  Root directory for the Open
adapter. This directory contains the log4j.xml configuration file

    bin/ Example scripts for starting the adapters and editor.

    config/

    examples/ Various component examples.

    libj/ All adapter and third-party distributable jar files.

    repo/ Standard location for all property files.

    sysam/
```

Include Files Syntax

Syntax for including an additional properties file.

```
#include other.props
```

The file name can be preceded by a prefix, which is added to each property name in the included file:

```
#include A.comp other.props
```

Where `other.props` contains:

```
property1=foo
```

The Open adapter reads:

```
A.comp.property1=foo
```

Variable Substitution

Defines a variable within the properties file or an included properties file.

You can define a variable within the properties file or an included properties file if the variable is defined before it is used:

```
NUM_TO_SEND=1000 … A.comp.MaxRecords=${NUM_TO_SEND}
```

You can also define the variable using the "-D" option to the Java Virtual Machine when the adapter is started:

```
java –DNUM_TO_SEND=1000 org.openadapter.adapter.RunAdapter
config.props A
```

Wildcard Property Names

If a component initializes and attempts to get property values for a property that is defined with a wildcard name, the `SuperProperties` class returns the value for the wildcard property unless there is a more specific property setting that matches the request.

For example, this matches any adapter and component names:

```
*.QueueManager=QM_Test
```

This matches exactly one component in the name:

```
A.?.QueueManager=QM_Test
```

`A.B.QueueManager` matches; however, `A.B.C.QueueManager` does not.

Autoincremented Property Names

You can autoincrement property names if there is a long list of properties to be specified.

For example:

```
A.comp.field1=foo A.comp.field2=bar A.comp.field3=hello
A.comp.field4=world
```

You can autoincrement these fields:

```
A.comp.field++=foo A.comp.field++=bar A.comp.field++=hello
A.comp.field++=world
```

Note: If you use the Adaptor Framework Editor, the autoincrement fields convert to the corresponding numbered fields on reading the configuration file. The Adaptor Framework Editor cannot revert the fields to autonumbered fields on rewriting the properties file.

XML Properties Files

Specify adapter properties files as XML documents.

```
<?xml version=”1.0” encoding=”UTF-8” ?>
<openadapter>
<A>
<Component1>
<Name>C1</Name>
</Component1>
<Component2>
<Name>C2</Name>
</Component2>
<C1>
<ClassName>com.sybase.adapter.ibm.MqSource</ClassName>
</C1>
<C2>
```

```
…
…
</A>
</openadapter>
```

Open Adapter Components

Add at least one source and one sink component to use the Open adapter. Source components read provided data, and sink components write to associated output.

Each component has its own required properties. Set the `DOStringReader` and `DOStringWriter` properties for the source and sink components to enable data passing through these components to be parsed and formatted by various parsers and formatters.

Event Stream Processor does not support multibyte character sets, such as UTF-16. However, an external source may contain non-ASCII characters. By default, the adapter interprets them as 1-byte or 2-byte Unicode characters, which may lead to data corruption. To set the encoding explicitly, add the `TextEncoding` property to the configuration file. For example:

```
Adapter. Component.TextEncoding=ASCII
```

If a property for defining multiple table names is specified as:

```
Adapter. Component.Table++=TableName
```

the configuration file contains:

```
Adapter. Component.Table1=TableName1
Adapter. Component.Table2=TableName2
```

You can define properties with a number of levels separated by a period. For example, a property specific to Table1 can be represented as:

```
Adapter. Component.TableName1.Field1=FieldName1
```

Source Components

The Open adapter has two source components: AsapSource and SpPersistentSubscribeSource.

AsapSource Properties

The AsapSource component subscribes to data from the Event Stream Processor stream name specified in the adapter configuration.

```
ClassName=com.sybase.esp.adapter.asap.AsapSource
```

Property	Description
`ProjectUri`	(Dependent required) Connect to the Server running in cluster mode. For example, `esp://localost:19011/ws1/p1`.

Property	Description
`User`	(Required) The initial connection between AsapSource and Event Stream Processor requires authentication. Enter a valid user name known to Event Stream Processor.
`Passwd`	(Required) The initial connection between AsapSource and Event Stream Processor requires authentication. Enter a valid password for the user name. If the `IsEncrypted` property is set to true, the user and password information is passed to Event Stream Processor before the SSL connection is set up. These details are passed in plain text.
`IsEncrypted`	(Optional) If this property is set to true, AsapSource attempts to use an SSL socket connection to Event Stream Processor.
`UseServerRSA`	(Optional) If true, server RSA authentication is used to connect to the Server. If you specify this property, also provide the `KeyStore` and `KeyStorePassword` properties. The Open adapter uses the Bouncy Castle Java security implementation. Ensure that Bouncy Castle is listed among other security providers in the `java.security` file of your Java Runtime Environment directory: `=org.bouncycastle.jce.provider.BouncyCastleProvider`
`KeyStore`	(Optional) Used for server RSA authentication. Specifies the location of the keystore (`.jks` file). Set this property if you specify `UseServerRSA`.
`KeyStorePassword`	(Optional) Specify the keystore password. This is used for server RSA authentication. Set this property if you specify `UseServerRSA`.
`UseKerberos`	(Optional) If true, Kerberos authentication is used to connect to the Server. If you specify this property, also provide the `KerberosKDC`, `KerberosRealm`, `KerberosService`, and `KerberosTicketCache` properties.
`KerberosKDC`	(Optional) Used for Kerberos authentication. Specifies the host name of Kerberos key distribution center. Set this property if you specify `UseKerberos`.
`KerberosRealm`	(Optional) Used for Kerberos authentication. Specifies the Kerberos realm setting. Set this property if you specify `UseKerberos`.
`KerberosService`	(Optional) Used for Kerberos authentication. Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Set this property if you specify `UseKerberos`.

Property	Description
KerberosTicketCache	(Optional) Used for Kerberos authentication. Specifies the location of the Kerberos ticket cache. Set this property if you specify UseKerberos.
UseUserPassword	(Optional) If set to true, User and Password authentication is used to connect to the Server. If you specify this, provide values for User and Password.
Stream++	(Optional) Name of stream to subscribe to.
MaxBlockBuildTime (milliseconds)	(Optional) AsapSource sends records in blocks for better performance. A block is built of individual records until the MaxBlockSize is reached or MaxBlockBuildTime elapses, whichever occurs first. Then, the block is sent along the adapter pipeline. If set to 0, there is no limit on the block building time. Default value is 1000 (milliseconds).
MaxBlockSize	(Optional) AsapSource sends records in blocks for better performance. A block is built of individual records until the MaxBlockSize is reached or MaxBlockBuildTime elapses, whichever occurs first. Then the block is sent along the adapter pipeline. Default value is 256.
RecordBufferCapacity	(Optional) If set to 0, records are sent along the adapter pipeline one at a time. If set to a positive number, AsapSource queues the records made available by the Event Stream Processor in an internal buffer. Buffered records are added to ongoing blocks on a dedicated thread so that the Pub/Sub subscription thread can continue buffering the records. When the buffer capacity is exceeded, the queue blocks until the buffer capacity becomes available again. Default value is 4096.
PulseInterval	(Optional) The number of seconds to wait until AsapSource gets the next record from Event Stream Processor. All updates to a record that are made on the Event Stream Processor during the pulse interval are coalesced and only the resulting record is sent. By default, records are received by the adapter instantaneously, rather than being pulsed.

See also

- *Example: Using the AsapSource Component* on page 524
- *Tips for Migrating Your Open Adapter Scripts* on page 472
- *SpPersistentSubscribeSource Properties* on page 480
- *AsapSink Properties* on page 482

SpPersistentSubscribeSource Properties

The SpPersistentSubscribeSource component subscribes to a stream in Event Stream Processor and sends records on to other components.

```
ClassName =
com.sybase.esp.adapter.asap.SpPersistentSubscribeSource
```

The stream the component subscribes to does not explicitly remove records until asked by the subscriber. Once records are processed, SpPersistentSubscribeSource publishes tags back to the Event Stream Processor to remove rows from the subscribed stream.

SpPersistentSubscribeSource has two additional streams: log stream and truncate stream. For example, you can have three streams named Stream1, Stream1_log, and Stream1Truncate. The log stream has two additional columns: sequence number and opcode. Records pass from Stream1 to Stream1_log, as well as increasing sequence number values. The opcode values in the opcode column in Stream1_log are "insert". After SpPersistentSubscribeSource subscribes to a batch of data (or a single record), the last sequence number of the records is published to Stream1Truncate, which then removes any records prior to that sequence number from the Stream1_log and persistent store (for example, file on hard disk).

Property	Description
`Host`	(Required) Host name for the Event Stream Processor command and control process.
`Port`	(Required) Port number for the Event Stream Processor command and control process.
`ProjectUri`	(Dependent required) Connect to the Server running in cluster mode. For example, `esp://localost:19011/ws1/p1`.
`IsEncrypted`	(Optional) If this property is set to true, AsapSource attempts to use an SSL connection to Event Stream Processor.
`UseServerRSA`	(Optional) If true, server RSA authentication is used to connect to the Server. If you specify this property, also provide the `KeyStore` and `KeyStorePassword` properties.
`KeyStore`	(Optional) Used for server RSA authentication. Specifies the location of the keystore (`.jks` file). Set this property if you specify `UseServerRSA`.
`KeyStorePassword`	(Optional) Specify the keystore password. This is used for server RSA authentication. Set this property if you specify `UseServerRSA`.

Property	Description
`UseKerberos`	(Optional) If true, Kerberos authentication is used to connect to the Server. If you specify this property, also provide the `KerberosKDC`, `KerberosRealm`, `KerberosService`, and `KerberosTicketCache` properties.
`KerberosKDC`	(Optional) Used for Kerberos authentication. Specifies the host name of Kerberos key distribution center. Set this property if you specify `UseKerberos`.
`KerberosRealm`	(Optional) Used for Kerberos authentication. Specifies the Kerberos realm setting. Set this property if you specify `UseKerberos`.
`KerberosService`	(Optional) Used for Kerberos authentication. Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Set this property if you specify `UseKerberos`.
`KerberosTicketCache`	(Optional) Used for Kerberos authentication. Specifies the location of the Kerberos ticket cache. Set this property if you specify `UseKerberos`.
`UseUserPassword`	(Optional) If set to true, `User` and `Password` authentication is used to connect to the Server. If you specify this, provide values for `User` and `Password`.
`SyncBaseStreams`	(Optional) Asks Event Stream Processor for SYNC_BASE_STREAMS. When set to true, the `publisher.commit()` method is called after each batch is published to the Event Stream Processor. Default value is true.
`CommitLimit`	(Optional) Max size of batch to process in one call. Default value is 100.
`User`	(Required) The initial connection between SpPersistentSubscribeSource and the Event Stream Processor requires authentication. Enter a valid user name known to the Event Stream Processor.
`Passwd`	(Required) The initial connection between SpPersistentSubscribeSource and the Event Stream Processor requires authentication. Enter a valid password for the user name.
`Stream++`	(Optional) Name of stream to subscribe to.
`<Stream++>.TruncateStream`	(Required) Stream responsible for truncating data.

Property	Description
`<Stream++>.OpcodeColumn`	(Required) Name for the column where the opcode value is stored.
`<Stream++>.SequenceColumn`	(Required) Name for the column where the sequence number is stored.
`QueueCapacity`	(Optional) SpPersistentSubscribeSource queues the records made available by the Event Stream Processor. The queued records are consumed by a separate thread. This property sets the capacity of the internal queue. When the queue is full, the adapter waits for space to become available. The default value is 4096. If the IsEncrypted property is set to true, the user and password information is passed to the Event Stream Processor before the SSL connection is set up. These details are passed in plain text.

See also

- *Example: Using the SpPersistentSubscribeSource Component* on page 531
- *Tips for Migrating Your Open Adapter Scripts* on page 472
- *AsapSource Properties* on page 477
- *AsapSink Properties* on page 482

Sink Components

The Open adapter has two sink components: AsapSink and WSSink.

AsapSink Properties

The AsapSink component takes records from the source and delivers them to Event Stream Processor.

Ensure that every input adapter configuration includes exactly one AsapSink component.

```
ClassName=com.sybase.esp.adapter.asap.AsapSink
```

Property	Description
`ProjectUri`	(Dependent required) Connect to the Server running in cluster mode. For example, `esp://localost:19011/ws1/p1`.
`User`	(Required) The initial connection between AsapSink and Event Stream Processor requires authentication. Enter a valid user name known to Event Stream Processor.

Property	Description
`Passwd`	(Required) The initial connection between AsapSink and Event Stream Processor requires authentication. Enter a valid password for the user name. If the `UseSSL` property is set to true, the user and password information is passed to the Event Stream Processor before the SSL connection is set up. These details are passed in plain text.
`Stream++`	(Required) The name of the stream to which the data is delivered.
`IsEncrypted`	(Optional) If present and set to true, AsapSink attempts to use an SSL connection to Event Stream Processor.
`UseServerRSA`	(Optional) If true, server RSA authentication is used to connect to the Server. If you specify this property, also provide the `KeyStore` and `KeyStorePassword` properties.
`KeyStore`	(Optional) Used for server RSA authentication. Specifies the location of the keystore (`.jks` file). Set this property if you specify `UseServerRSA`.
`KeyStorePassword`	(Optional) Specify the keystore password. This is used for server RSA authentication. Set this property if you specify `UseServerRSA`.
`UseKerberos`	(Optional) If true, Kerberos authentication is used to connect to the Server. If you specify this property, also provide the `KerberosKDC`, `KerberosRealm`, `KerberosService`, and `KerberosTicketCache` properties.
`KerberosKDC`	(Optional) Used for Kerberos authentication. Specifies the host name of Kerberos key distribution center. Set this property if you specify `UseKerberos`.
`KerberosRealm`	(Optional) Used for Kerberos authentication. Specifies the Kerberos realm setting. Set this property if you specify `UseKerberos`.
`KerberosService`	(Optional) Used for Kerberos authentication. Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Set this property if you specify `UseKerberos`.
`KerberosTicketCache`	(Optional) Used for Kerberos authentication. Specifies the location of the Kerberos ticket cache. Set this property if you specify `UseKerberos`.
`UseUserPassword`	(Optional) If set to true, `User` and `Password` authentication is used to connect to the Server. If you specify this, provide values for `User` and `Password`.

Property	Description
`SHINE`	(Optional) Used for update and upsert operations. If a field is set to shine through, then an update to an existing record does not affect the value of that field.
`PublishMethod`	(Optional) Determines how the data is published to the Event Stream Processor: RECORD, COLLECTION, ENVELOPE, TRANSACTION. With the RECORD method, records are published individually. The other three types are bulk publisher methods. The ENVELOPE and TRANSACTION methods behave similarly. They create blocks per streams and then publish those blocks once the required number of records accumulates. With TRANSACTION, if any single record in the block fails to publish, the entire block fails to publish. However, with ENVELOPE, the remaining records have other attempts at being published. The COLLECTION type is similar to ENVELOPE but it creates blocks regardless of streams and publishes all records in the block simultaneously.
`SyncBaseStreams`	(Optional) Asks Event Stream Processor for SYNC_BASE_STREAMS. When set to true, the `publisher.commit()` method is called after each batch is published to the Event Stream Processor. Default value is false.
`DiscardFile`	(Optional) Name of the file for discarded SDOs.
`TruncateDiscardFile`	(Optional) If set to true, the file is truncated.
`EspOpsColumn`	(Optional) If set, the value of the ESP_OPS attribute in the incoming records is written to the corresponding column and the record is treated as an UPSERT regardless of the ESP_OPS value.
`MaxRecordsPerBlock`	(Optional) An integer value specifying the maximum number of records to send as a transactional unit to Event Stream Processor. If no value is specified, the default is 0 and all records in the collection get published in a single transaction block to Event Stream Processor.

See also

- *Example: Using the AsapSink Component* on page 522
- *Tips for Migrating Your Open Adapter Scripts* on page 472

- *AsapSource Properties* on page 477
- *SpPersistentSubscribeSource Properties* on page 480

WSSink Properties

The WSSink component is a client implementation of a Web service, allowing communication with remote services.

WSSink is consistent with the WSDL descriptor in `lib/WEB-INF/WSAdapterSource.wsdl`.

Note: Build server-site Web services based on the WSDL descriptor located in `lib/WEB-INF/WSAdapterSource.wsdl` of the adapter installation directory.

The Web service client uses simple objects called data transfer objects (DTOs) as data containers. The classes used are:

1. `com.sybase.adapter.soap.DataTransferObject`

   ```
   public class DataTransferObject {
   private String name;
   private int opcode;
   private Object[] data;
   }
   ```

   Ensure the structure of the `data` field is the same as that defined in the Web service. You can obtain metadata for DTOs.
2. `com.sybase.adapter.soap.DTOMetaData`

   ```
   public class DTOMetaData {
   private String name;
   private DTOAttribute[] attributes;
   }
   ```

   which uses class:`com.sybase.adapter.soap.DTOAttribute`

   ```
   public class DTOAttribute {
   private String name;
   private String xsdType;
   }
   ```

   DataTransferObject, DTOMetaData, and DTOAttribute all offer getter and setter methods. You can also obtain object definitions from the WSDL descriptor of the service.

Property	Description
`URL`	(Required) URL string of the server Web service. Example of valid value is "http://eult121.mycompany.com:9085/services/WSAdapter-Source".
`TypeN`	(Required) Name of the message type with which the source component transmits data. `TypeN` is also the name of the exposed DTO.

Property	Description
`TypeN.<DOType>`	(Required) Name of the DTO configured on the remote service.
`ManualMapping`	(Optional) If true, sink uses mapping AttName->DtoAttName given in the configuration file. If false, sink gets DTO information from the service and assumes that all attribute names defined in DTO are also present in the incoming adapter message.
`TypeN.AttName++`	(Dependent optional) Name of the field as passed by the source within the adapter. It is also the name of the DTO attribute.
`TypeN.<AttName++>. DTOAttName`	(Dependent optional) Name of attribute defined by remote Web service for the selected DTO type.
`DiscardedLogger-Name`	(Optional) Name of logger responsible for logging any records that have not processed successfully.

See also

- *Example: Using the WSSink Component* on page 533

Pipe Components

The Open adapter has two pipe components: BeanShellPipe and JDBCLookupPipe.

BeanShellPipe Properties

BeanShellPipe is a scriptable pipe used to modify a message.

You can script the entire message or each field individually in Java, and use this component in incoming and outgoing flows. The scripting is implemented through BeanShell. Refer to *http://www.beanshell.org* for details.

`ClassName: com.sybase.esp.adapter.scripting.BeanShellPipe`

Property	Description
`MsgPreProcessor`	(Required) BeanShell script for this message. The script is applied to the message before any fields are processed. The script has access to a message object type. The name of the variable is *message*. For example: `System.out.println("Message received; SDO array size= " + message.peekDataOb-jects().length);`

Property	Description
`MsgPostProcessor`	(Required) BeanShell script for this message. The script is applied to the message after all fields have been processed. The script has access to a message object type. The name of the variable is message. For example: `System.out.println("Message sent; SDO array size= " + message.peekDataObjects().length);`
`Type++`	(Required) The type of the data object that is received from the source component. For an incoming flow (one flowing into Event Stream Processor through an AsapSink component), this is the Event Stream Processor Base Stream name to be updated with the message. For an outgoing flow (one originating from published data from an Event Stream Processor stream), this is the Event Stream Processor stream name that is publishing the data.
`.PreProcessor`	(Optional) BeanShell script for this message type. The script is applied to the message before any fields are processed. The script has access to a SimpleDataObject object type. The name of the variable is *sdo*. `Typen` is the name of the message type or stream as defined in the associated `Type` property. For example: `System.out.println("Got data for message type: " + sdo.getType().getName());`
`.PostProcessor`	(Optional) BeanShell script for this message type. The script is applied to the message after all fields have been processed. The script has access to a SimpleDataObject object type. The name of the variable is *sdo*. Typen is the name of the message type or stream as defined in the associated `Type` property. For example: `System.out.println("Sending data for message type: " + sdo.getType().getName());`
`.AttName++`	(Required) Field names contained in the associated message type. Bean-Shell scripting is required for this. If the field name does not exist in the received message type, a new field is created. `Typen` is the name of the message type or stream as defined in the associated `Type` property.

Property	Description
`.Script`	(Optional) BeanShell script for this field. The script has access to a SimpleDataObject object type. The name of the variable is *sdo*. `Typen` is the name of the message type/stream as defined in the associated `Type` property.
`AttNamex`	(Optional) Field name as defined in the associated `AttName` property. For example: `if (sdo.isPresent("Amount ") && sdo.getAttributeValue ("Amount")>0 {value="AC";} else {value="CO"}`

See also

- *Example: Using the BeanShellPipe Component* on page 526

JDBCLookupPipe Properties

The JDBCLookupPipe component queries a database at start-up and uses the cached result set as a lookup table.

`ClassName: com.sybase.esp.adapter.jdbc.JDBCLookupPipe`

Each record in the lookup table consists of a unique lookup key and an array of added attributes. The lookup key consists of one or more attributes. When a data object arrives from the source:

- The values of the key attributes are matched against a record in the lookup table.
- If no record matches, the data object is passed on to the sink without any transformation.
- If a record in the lookup table does match the value of the key attributes, the added attributes from the lookup table are added to the record, and the record result is passed on to the sink.

Property	Description
`JdbcDriver`	(Required) The JDBC driver that connects to the database. For example: `oracle.jdbc.OracleDriver`
`JdbcUrl`	(Required) The location of the database. For example: `jdbc:oracle:thin:@myhost.com:1521:mydatabase`
`DBProperty++`	(Optional) Name of a database property that the pipe sets when connecting to the database. For example, the user name, password, database name, and so on.

Property	Description
DBPropertyn.Value	(Dependent optional) Value for the associated DBProperty. Set this property if the DBProperty++ property is set.
Table	(Required) Name of the database table where lookup is performed.
KeyAttName++	(Required) Attribute names that make up the lookup key.
KeyDbCol++	(Required) Names of the database columns that correspond to KeyAttNames.
ValueAttName++	(Required) Names of the attributes used for added values.
ValueDbCol++	(Required) Names of the database columns that correspond to ValueAttNames.
WhereClause	(Optional) The WHERE clause that is part of the lookup SELECT query. The lookup query uses this form: `SELECT KeyDbCol1, KeyDbCol2, ... , ValueDbCol1, ValueDbCol2, ... FROM Table WHERE WhereClause`

Example

The Oracle database table "MyTable = (SYMBOL, ID, PRICE)" is used for lookup. Each data object has four attributes: AttA, AttB, AttC and AttD. AttA and AttB correspond to SYMBOL and ID respectively and are used as a lookup key, and AttD corresponds to PRICE and is added to the data object received from the source. Here is an example of the pipe configuration:

```
adapter.LOOKUPPIPE.ClassName=
com.sybase.esp.adapter.jdbc.JdbcLookupPipe
adapter.LOOKUPPIPE.JdbcUrl = jdbc:oracle:thin:@myhost.com:
1521:mydatabase
adapter.LOOKUPPIPE.JdbcDriver = oracle.jdbc.OracleDriver
adapter.LOOKUPPIPE.DBProperty1 = user
adapter.LOOKUPPIPE.DBProperty1.Value = MyUser
adapter.LOOKUPPIPE.DBProperty2 = password
adapter.LOOKUPPIPE.DBProperty2.Value = MyPassword
adapter.LOOKUPPIPE.Table = MyTable
adapter.LOOKUPPIPE.KeyDbCol1 = SYMBOL
adapter.LOOKUPPIPE.KeyAttName1 = AttA
adapter.LOOKUPPIPE.KeyDbCol2 = ID
adapter.LOOKUPPIPE.KeyAttName2 = AttB
adapter.LOOKUPPIPE.ValueDbCol1 = PRICE
adapter.LOOKUPPIPE.ValueAttName1 = AttD
adapter.LOOKUPPIPE.WhereClause = SYMBOL LIKE 'A%'
```

See also

- *Example: Using the JDBCLookupPipe Component* on page 527

Reader Components

The Open adapter has four reader components: MultiFlatXmlStringReader, XPathXmlStreamReader, XPathMultiTypeXmlReader, and EspDelimitedStringReader.

MultiFlatXmlStringReader Properties

The MultiFlatXmlStringReader component handles messages quickly, provides the flexibility to set defaults based on message content, and splits data into multiple tables in Event Stream Processor.

This reader uses a simple XML format, where the name of the table is the tag and the fields are the attributes.

If MultiFlatXmlStringReader is selected as the parsing method, sources can populate multiple tables (defined streams). Specify an internal table or tables that Event Stream Processor updates based on data from the source. Also define the fields within each source record by specifying the name and datatype for each field.

Source records for MultiFlatXmlStringReader have this format:

```
<TableName field1='field1 data' field2='field2 data' … />
```

Property	Description
`AcceptAmper-sand`	(Default required) Enter a true or false value to indicate whether the adapter accepts non-XML uses of the ampersand (&). True indicates that the adapter accepts non-XML uses of the ampersand. For example, the adapter converts "&" , "<" , and so on, but it also accepts values such as "Marks & Spencer". False indicates that the adapter rejects non-XML uses of the ampersand.
`Type++`	(Required) Type the name of the base stream or streams that Event Stream Processor updates based on the data in the source. Repeat this process for each stream you are updating. Ensure each system table has its own `Typen` property.
`Typen.AttName++`	(Required) Type the name of the table field that Event Stream Processor updates based on the data in the source. This field is case-sensitive. `Typen` is the name of the related table. Specify a name for each record field in the source data.

Property	Description
`Typen.AttType++`	(Default required) The system defaults the datatype for the field. `Typen` is the name of the related table. Event Stream Processor supports these datatypes: • `string` – for strings • `datetime` – for dates • `float`– for floating-point numbers • `short` – for 16-bit signed integers • `integer` – for 32-bit signed integers • `long` – for 64-bit signed integers Specify a datatype for each record field in the source data.
`Typen.Format++`	(Default optional) If you created a field with a datetime datatype, enter the format that the adapter understands when reading data for that field. The adapter rejects any data that is not in this format. `Typen` is the name of the related table. If you do not specify a value, the adapter understands only datetime values with the format `yyyyMMdd` or `yyyyMMdd HH:mm:ss` for the field. It rejects any other datetime data.
`Typen.Match`	(Required) Enter the regular expression to match records for this table. `Typen` is the name of the related table. Provide a regular expression for each table. For example: `.*table_is_x.*`
`Typen.UTCTime-Zone++`	(Default optional) If you created a field with a datetime datatype, you can enter the time zone for the field. `Typen` is the name of the related table. The adapter converts and normalizes the corresponding datetime value from its originating time zone value to an equivalent UTC value. The UTC value is then passed to Event Stream Processor for storage. You can enter any time zone that Java recognizes (for example, Europe/London or America/New_York). If there is no specified value, the datetime value passes through as local time to Event Stream Processor for storage.

See also

- *Example: Using the MultiFlatXmlStringReader Component* on page 529
- *Valid Time Zones for the Open Adapter* on page 502

XPathXmlStreamReader Properties

The XPathXmlStreamReader component handles XML documents using XPath properties. Select XPathXmlStreamReader as the parsing method to get sources to populate a number of tables.

```
DOStringReader=com.sybase.esp.adapter.xml.xpath.XPathXmlStreamReade
r
```

Specify a base stream that Event Stream Processor updates based on data from the source. Also, define the fields within each source record by specifying the name and datatype for each field.

You can populate fields with data from an XML document by specifying tag data or attribute values. Specify nested tags by using a forward slash (/) to separate the tag names:

For example, the field data is set to xyz.

```
XPath=/env/body/tag
<env>
<body>
<tag> xyz </tag>
</body>
</env>
```

Attributes are specified by [@attributeName].

In this case, the field data is set to abc.

```
XPath=/env/body/tag[@attr]
<env>
<body>
<tag attr= abc />
</body>
</env>
```

The XPathXmlStreamReader handles collections.

```
XPath=/env/body/tag
<env>
<body>
<tag> xyz </tag>
<tag> abc </tag>
</body>
</env>
```

By default, the command above inserts both values into the field, separated by the collection separator character: xyz|abc.

If a specific tag value is required, use an index operator to specify the position in the collection:

```
XPath=/env/body/tag[2]
<env>
<body>
<tag> xyz </tag>
```

```
<tag> abc </tag>
</body> </env>
```

This command inserts the value of only the second tag.

Property	Description
`XmlRoot`	(Required) Enter the root node of the XML document. For example: `env`
`DateFormat`	(Optional) If you create fields with a datetime datatype, you can type the default format that the adapter understands when reading data for that field. Unless overridden by the field's `Format` property, the adapter rejects any data that is not in this format. If you do not enter a value, the adapter only understands datetime values with the format `yyyyMMdd` or `yyyyMMdd HH:mm:ss` for the field. It rejects any other datetime data.
`Type++`	(Required) Name of the base streams that Event Stream Processor updates based on the data in the source.
`XPath`	(Required) Enter an XPath-style expression for the root node of the table. For example: `/env`
`.AttName++`	(Required) Names of the table fields that Event Stream Processor updates based on the data in the source. This property is case-sensitive. Specify a name for each record field in the target Event Stream Processor base stream.

`.XPath`	(Required) Enter an XPath-style expression for the data to be inserted into this field. If the expression begins with "/", it is taken as a full path. Otherwise, it is relative to the `Typen.XPath` property. For example:
	`tag`
	or:
	`/env/body/tag`
	If you specify a full path, you cannot access a field at a higher level than the `Typen.XPath` property.

`.DefaultValue`	(Optional) If the field is empty, for example, it is an empty tag or the tag is not present in the XML document, this value is substituted.

Property	Description
`.Format`	(Dependent optional) If you created a field with a datetime datatype, you may type the format that the adapter understands when reading data for that field. The adapter rejects any data that is not in this format. If you do not enter a value, the adapter understands datetime values only with the format `yyyyMMdd`, `yyyyMMdd HH:mm:ss` for the field, or with the default value from the `DateFormat` property. It rejects any other datetime data.
`.Match`	(Optional) If necessary, type a regular expression match for the adapter to perform on the record. AttNamex is the field name as defined in the `Att-Name` property. If the regular expression is matched in the field data, the string defined in AttNamex.MatchReplace is substituted. For example: `.*char_is_(.).*`
`.MatchReplace`	(Dependent optional) Type the replacement value for the `.Match` property that the adapter may use when the corresponding regular expression match is successful.
`.AttType`	(Default required) Type the datatype for the field. `AttNamex` is the field name as defined in the `AttName` property. Event Stream Processor supports these datatypes: • `string` – for strings • `datetime` – for dates • `float` – for floating-point numbers • `short` – for 16-bit signed integers • `integer` – for 32-bit signed integers • `long` – for 64-bit signed integers
`.UTCTimeZone`	(Dependent optional) If you created a field with a datetime datatype, you may type the time zone for the field. The adapter converts and normalizes the corresponding datetime value from its originating time zone value to an equivalent UTC value. The UTC value is then passed to Event Stream Processor for storage. You may type any time zone that Java recognizes (for example, Europe/London or America/New_York). If no value is set, the datetime value passes through as local time to Event Stream Processor for storage.

Property	Description
`BadRecordLoggerName`	(Optional) The name of the logger responsible for writing bad records. The behavior depends on implementation. If the name is provided, also provide the implementation class. If this property is left blank, the adapter warns only about bad records but the original message is lost. `<BadRecordLoggerName>.ClassName - Logger implementation`

See also

- *Example: Using the XPathXmlStreamReader Component* on page 537
- *Valid Time Zones for the Open Adapter* on page 502

XPathMultiTypeXmlReader Properties

The XPathMultiTypeXmlReader component handles XML messages.

```
DOStringReader=com.sybase.esp.adapter.xml.xpath.XPathMultiTypeXmlReader
```

This reader uses XPathXmlStreamReader, depending on the message type provided in XML. Once the message type is obtained, this component uses the standard XPathXmlStreamReader component to handle incoming messages. All configuration property files for the XPathXmlStreamReader component are stored in separate files called parsing rules. The list of properties in parsing rules are similar to XPathXmlStreamReader except that they require prefix parsing rules.

Property	Description
`MSGTypeXPath`	(Required) Location of the message type in the XML message expressed as XPath. This can also be provided as an attribute of the element.
`MSGTypeN`	(Required) Name of message type. One of the message type names must match the value obtained by MSGTypeXPath from XML message.
`MSGTypeN.ParsingRules`	(Required) Name of the file where XPathXmlStreamReader stores its properties.

Property	Description
`BadRecordLogger-Name`	(Optional) The name of the logger responsible for writing bad records. The behavior depends on implementation. If you provide a name, also provide the implementation class. If you leave this property blank, the adapter only warns about bad records, and the original message is lost. `<BadRecordLoggerName>.ClassName - Logger implementation.`

See also

- *Example: Using the XPathMultiTypeXmlReader Component* on page 536
- *Valid Time Zones for the Open Adapter* on page 502

EspDelimitedStringReader

The EspDelimitedStringReader component handles delimited (for example, comma separated) messages. You can use it to send incorrect records to a bad record file.

```
DOStringReader=com.sybase.esp.adapter.dostrings.EspDelimitedStringReader
```

Property	Description
`BadRecordLog-gerName`	(Optional) Name of the logger responsible for writing bad records. The behavior depends on the implementation. If you provide a name, also provide the implementation class. If you leave this property blank, the adapter warns only about bad records, and the original message is lost. `<BadRecordLoggerName>.ClassName - Logger implementation.`

`EmptyStringAs-Null`	(Optional) If set to true, empty string values are translated to null values. For example:
	`NULL`
	If the input record contains:
	`A,B,NULL,D`
	Then the resulting field values is:
	`A, B,{null},D`

`FieldDelimiter`	(Default required) Character number for a single-character field delimiter. Use either "uXXXX" for a hexadecimal unicode value, or "DDD" for a decimal ASCII value. Use this property if the field is a nonprintable or whitespace character (for example, "space", "tab", or "null"). The default delimiter is the comma (ASCII 44).

Property	Description
`NullString`	(Optional) A string, which if encountered as a field value, causes the adapter to insert a null string in the resulting message for this field.
`StripQuotes`	(Optional) If set to true and a field is "quoted", the quote characters are stripped from the beginning and end of the field value. Default value is true.

See also

- *Valid Time Zones for the Open Adapter* on page 502

Writer Component

The Open adapter has one writer component, the XPathXmlWriter.

XPathXmlStringWriter Properties

The XPathXmlWriter component uses an XPath-style syntax to format XML documents from published Event Stream Processor stream data.

The formatter formats XML tags and attributes. To use this writer, ensure the sink specifies this property:

```
DOStringWriter = com.sybase.esp.adapter.xml.xpath.XmlStringWriter
```

Specify nested tags by separating the tag names by / :

```
/env/body/tag
<env>
<body>
<tag>xyz</tag>
</body>
</env>
```

Attributes are specified by [@attributeName].

```
/env/body/tag[@attr]
<env>
<body>
<tag attr='xyz'/>
</body>
</env>
```

By default, the formatter creates collections. For example, a new nested tag is created for each occurrence of a tag name:

```
XPath1=/env/body/tag
XPath2=/env/body/tag
<env>
<body>
<tag>xyz</tag>
</body>
<body>
<tag>abc</tag>
```

```
</body>
</env>
```

If tags are added within a nesting, use an index operator to specify the position in the collection:

```
XPath1=/env/body[1]/tag
XPath2=/env/body[1]/tag
<env>
<body>
<tag>xyz</tag>
<tag>abc</tag>
</body>
</env>
```

XML content encoding for this component is iso-8859-1.

Property	Description
`Type++`	(Required) Name of the Event Stream Processor stream to be exported.

`.XPath`	(Required) The XPath–style description of the top-level tags for this table.
	`/env/body`
	`<env/>`
	`<body/>`

`.AttName++`	(Required) Name of a field within the `Type` (stream).

`.XPath`	(Required) The XPath–style description of the XML tag or attribute to be formatted. `Typen` is the table name and `AttNamen` is the field name specifying the source of the data to insert into the tag or attribute.
	If the description specifies a tag, the field is output as tag data:
	`/env/body/tag`
	`<env>`
	`<body>`
	`<tag>field data from Typen.AttNamex</tag>`
	`</body>`
	`</env>`
	If the description specifies an attribute, the attribute value is set to the field data:
	`/env/body/tag[@attr]`
	`<env>`
	`<body>`
	`<tag attr= field data from Typen.AttNamex />`
	`</body>`
	`</env>`

See also

- *Example: Using the XPathXmlStringWriter Component* on page 538

Specifying Datetime Formats

You can specify the acceptable format for dates in a file, if you are using a system that reads data from a file.

The Open adapter rejects dates that are not in the specified format. If you do not specify an acceptable datetime format, the adapter understands datetime values only with the format `yyyyMMdd` or `yyyyMMdd HH:mm:ss`, and rejects any other datetime data.

If you specify the format in the form of a template string, use special identifiers for day, year, month, and so on, along with formatting characters. Use uppercase H for hour to ensure the use of a 24-hour clock.

Table 6. Datetime Format Identifiers

Character	Description	Typical Usage
y	A digit of year	yyyy
M	A digit of month	MM
d	A digit of day	dd
H	A digit of hour	HH
m	A digit of minute	mm
s	A digit of second	ss
S	A digit of millisecond	SS

If your input data contains letters, enter them in single quotation marks. For example, if the input has strings like `Day:2003-12-29 Time:10:22-00`, specify a datetime format of `'Day' :yyyy-MM-dd 'Time' :HH:mm:ss`. Entering the format in this way prevents the Open adapter from mistaking the letters as formatting instructions.

Examples of specifying datetime formats:

- If you read dates from a file formatted as `2003/06/29`, where the year is 2003, the month is 06 (June), and the day is 29, enter the datetime format as `yyyy/MM/dd`.
- If you read dates from a file formatted as `29-06-2003 19:12:45`, where the day is 29, the month is 06 (June), the year is 2003, and the time is 7:12:45 PM, enter the datetime format as `dd-MM-yyyy HH:mm:ss`.
- If MQ-Series passes a value for MQPutDateTime in the format of `2003-06-29 19:12:45.493`, where the year is 2003, the month is 06 (June), the day is 29, and the time is 7:12:45 PM and 493 milliseconds, enter the datetime format as `yyyy-MM-dd HH:mm:ss.SSS`.

Note: The Open adapter strips off any milliseconds that it reads through datetime.

See also

- *Valid Time Zones for the Open Adapter* on page 502

Third-Party JAR Files

The Open adapter distribution includes a number of third-party distributable JAR files.

Note: The distribution does not contain the MSSQL JDBC driver, which you can download from *http://www.microsoft.com/downloads*. Search for 'mssql jdbc driver'. The Open adapter supports the SQL Server 2000 driver for JDBC SP3.

File	Description	License Information
`.../libj/esp_adapter_openadaptor.jar`	Contains SAP Open adapter components	SAP
`.../libj/openadaptor.jar`	SAP build of Open adaptor sources based on version 1.7	*https://www.openadaptor.org/licence.html*
`.../libj/esp_sdk.jar`	SAP Event Stream Processor Java SDK library	SAP

`.../libj/jetty-6.0.1.jar`
`.../libj/jetty-util-6.0.1.jar`
`.../libj/servlet-api-2.5-6.0.1.jar`
	Jetty client libraries	*http://www.apache.org/licenses*

`.../libj/bsh-2.0b4.jar`	Library containing BeanShell scripting implementation.	LGPL *http://www.beanshell.org/license.html*
`.../libj/commons-codec-1.3.jar`	Part of Apache Commons project	Apache license *http://jakarta.apache.org/site/jspa-position.html*
`.../libj/commons-collections-3.2.1.jar`	Part of Apache Commons project	Apache license *http://jakarta.apache.org/site/jspa-position.html*
`.../libj/commons-configuration-1.6.jar`	Part of Apache Commons project	Apache license *http://jakarta.apache.org/site/jspa-position.html*

File	Description	License Information
.../libj/commons-lang-2.6.jar	Part of Apache Commons project	Apache license *http://jakarta.apache.org/site/jspa-position.html*
.../libj/commons-logging-1.1.jar	Part of Apache Commons project	Apache license *http://jakarta.apache.org/site/jspa-position.html*
.../libj/dom4j-1.5.jar	DOM XML implementation	BSD style *http://www.dom4j.org/license.html*
.../libj/jakarta-oro-2.0.8.jar	Java classes that provide Perl5 compatible regular expressions, AWK-like regular expressions, glob expressions, and utility classes for performing substitutions, splits, filtering file names, and so on.	Apache license *http://svn.apache.org/repos/asf/jakarta/oro/trunk/LICENSE*
.../libj/log4j-1.2.16.jar	Logging implementation for Java	Apache license *http://logging.apache.org/log4j/2.x/*
.../libj/xerces-impl-2.9.1.jar	Xerces2 XML implementation	Apache license *http://xerces.apache.org/xerces-j/*
.../libj/xmlrpc-client-3.1.3.jar .../libj/xmlrpc-common-3.1.3.jar	XML RPC implementation for Java	Apache license *http://ws.apache.org/xmlrpc/*
.../libj/esp_adapter_api.jar	External Java adapter framework API	SAP
.../libj/axis.jar	An implementation of the SOAP submission to W3C	Apache
.../libj/esp_i18n.jar	Internationalization of messages	SAP

File	Description	License Information
`.../libj/esp_license.jar`	Event Stream Processor licensing API	SAP
`.../libj/sylapi.jar`	Required by SAP Sybase Event Stream Processor licensing	SAP
`.../libj/ws-commons-util-1.0.2.jar`	Apache Web service common utilities	Apache

Valid Time Zones for the Open Adapter

Examples of possible valid time zones for `UTCTimeZone` properties in various adapter reader components.

In the `UTCTimeZone` property, you can set any time zone that Java recognizes. Currently, there are over 500 time zones. You can retrieve the comprehensive list in Java through the TimeZone object's `getAvailableIDs()` method:

```
TimeZone.getAvailableIds()
```

See also

- *Specifying Datetime Formats* on page 499
- *MultiFlatXmlStringReader Properties* on page 490
- *XPathXmlStreamReader Properties* on page 492
- *XPathMultiTypeXmlReader Properties* on page 495
- *EspDelimitedStringReader* on page 496

Africa Time Zones

Valid time zones to specify for Africa in the `UTCTimeZone` property.

Country	Time Zone
Algeria	Africa/Algiers
Angola	Africa/Luanda
Benin	Africa/Porto-Novo
Botswana	Africa/Gaborone
Burkina Faso	Africa/Ouagadougou
Burundi	Africa/Bujumbura
Cameroon	Africa/Douala

Country	Time Zone
Cape Verde	Atlantic/Cape_Verde
Central African Republic	Africa/Bangui
Chad	Africa/Ndjamena
Comoros	Indian/Comoro
Democratic Republic of Congo	Africa/Kinshasa Africa/Lubumbashi
Republic of the Congo	Africa/Brazzaville
Cote D'Ivoire	Africa/Abidjan
Djibouti	Africa/Djibouti
Egypt	Africa/Cairo
Equatorial Guinea	Africa/Malabo
Eritrea	Africa/Asmera
Ethiopia	Africa/Addis_Ababa
Gabon	Africa/Libreville
Gambia	Africa/Banjul
Ghana	Africa/Accra
Guinea	Africa/Conakry
Guinea-Bissau	Africa/Bissau
Kenya	Africa/Nairobi
Lesotho	Africa/Maseru
Liberia	Africa/Monrovia
Libya	Africa/Tripoli
Madagascar	Indian/Antananarivo
Malawi	Africa/Blantyre
Mali	Africa/Bamako Africa/Timbuktu
Mauritania	Africa/Nouakchott

Country	Time Zone
Mauritius	Indian/Mauritius
Mayotte	Indian/Mayotte
Morocco	Africa/Casablanca
Western Sahara	Africa/El_Aaiun
Mozambique	Africa/Maputo
Namibia	Africa/Windhoek
Niger	Africa/Niamey
Nigeria	Africa/Lagos
Reunion	Indian/Reunion
Rwanda	Africa/Kigali
St Helena	Atlantic/St_Helena
Sao Tome and Principe	Africa/Sao_Tome
Senegal	Africa/Dakar
Seychelles	Indian/Mahe
Sierra Leone	Africa/Freetown
Somalia	Africa/Mogadishu
South Africa	Africa/Johannesburg
Sudan	Africa/Khartoum
Swaziland	Africa/Mbabane
Tanzania	Africa/Dar_es_Salaam
Togo	Africa/Lome
Tunisia	Africa/Tunis
Uganda	Africa/Kampala
Zambia	Africa/Lusaka
Zimbabwe	Africa/Harare

Asia Time Zones

Valid time zones to specify for Asia in the `UTCTimeZone` property.

Country	Time Zone
Afghanistan	Asia/Kabul
Armenia	Asia/Yerevan
Azerbaijan	Asia/Baku
Bahrain	Asia/Bahrain
Bangladesh	Asia/Dacca
Bhutan	Asia/Thimbu
British Indian Ocean Territory	Indian/Chagos
Brunei	Asia/Brunei
Burma / Myanmar	Asia/Rangoon
Cambodia	Asia/Phnom_Penh
China	Asia/Harbin Asia/Shanghai Asia/Chungking Asia/Urumqi Asia/Kashgar
Hong Kong	Asia/Hong_Kong
Taiwan	Asia/Taipei
Macao	Asia/Macao
Cyprus	Asia/Nicosia
Georgia	Asia/Tbilisi
India	Asia/Calcutta
Indonesia	Asia/Jakarta Asia/Ujung_Pandang Asia/Jayapura
Iran	Asia/Tehran

Country	Time Zone
Iraq	Asia/Baghdad
Israel	Asia/Jerusalem
Japan	Asia/Tokyo
Jordan	Asia/Amman
Kazakhstan	Asia/Almaty Asia/Aqtobe Asia/Aqtau
Kirgizstan	Asia/Bishkek
Korea (North and South)	Asia/Seoul Asia/Pyongyang
Kuwait	Asia/Kuwait Asia/Vientiane
Lebanon	Asia/Beirut
Malaysia	Asia/Kuala_Lumpur Asia/Kuching
Maldives	Indian/Maldives
Mongolia	Asia/Dariv Asia/Ulan_Bator Asia/Baruun-Urt
Nepal	Asia/Katmandu
Oman	Asia/Muscat
Pakistan	Asia/Karachi
Palestine	Asia/Gaza
Philippines	Asia/Manila
Qatar	Asia/Qatar
Saudi Arabia	Asia/Riyadh
Singapore	Asia/Singapore

Country	Time Zone
Sri Lanka	Asia/Colombo
Syria	Asia/Damascus
Tajikistan	Asia/Dushanbe
Thailand	Asia/Bangkok
Turkmenistan	Asia/Ashkhabad
United Arab Emirates	Asia/Dubai
Uzbekistan	Asia/Samarkand Asia/Tashkent
Vietnam	Asia/Saigon
Yemen	Asia/Aden

Australasia Time Zones

Valid time zones to specify for Australasia in the `UTCTimeZone` property.

Country	Time Zone
Australia	Australia/Adelaide Australia/Brisbane Australia/Broken_Hill Australia/Darwin Australia/Hobart Australia/Lindeman Australia/Lord_Howe Australia/Melbourne Australia/Perth Australia/Sydney
Christmas	Indian/Christmas
Cook Islands	Pacific/Rarotonga
Cocos	Indian/Cocos
Fiji	Pacific/Fiji

Country	Time Zone
French Polynesia	Pacific/Gambier Pacific/Marquesas Pacific/Tahiti
Guam	Pacific/Guam
Kiribati	Pacific/Tarawa Pacific/Enderbury Pacific/Kiritimati
North Mariana Islands	Pacific/Saipan
Marshall Island	Pacific/Majuro Pacific/Kwajalein
Micronesia	Pacific/Yap Pacific/Truk Pacific/Ponape Pacific/Kosrae
Nauru	Pacific/Nauru
New Caledonia	Pacific/Noumea
New Zealand	Pacific/Auckland Pacific/Chatham
Niue	Pacific/Niue
Norfolk	Pacific/Norfolk
Palau (Belau)	Pacific/Palau
Papua New Guinea	Pacific/Port_Moresby
Pitcairn	Pacific/Pitcairn
American Samoa	American Samoa
W Samoa	Pacific/Apia
Solomon Islands	Pacific/Guadalcanal
Tokelau Islands	Pacific/Fakaofo

Country	Time Zone
Tonga	Pacific/Tongatapu
Tuvalu	Pacific/Funafuti
US minor outlying islands	Pacific/Johnston Pacific/Midway Pacific/Wake
Vanuatu	Pacific/Efate
Wallis and Futuna	Pacific/Wallis

Europe Time Zones

Valid time zones to specify for Europe in the `UTCTimeZone` property.

Country	Time Zone
Andorra	Europe/Andorra
Austria	Europe/Vienna
Belarus	Europe/Minsk
Belgium	Europe/Brussels
Britain / Ireland	Europe/London Europe/Belfast Europe/Dublin
Bulgaria	Europe/Sofia
Czech Republic	Europe/Prague
Denmark, Faeroe Islands, and Greenland	Europe/Copenhagen Atlantic/Faeroe
Estonia	Europe/Tallinn
Finland	Europe/Helsinki
France	Europe/Paris
Germany	Europe/Berlin
Gibraltar	Europe/Gibraltar
Greece	Europe/Athens

Country	Time Zone
Hungary	Europe/Budapest
Iceland	Atlantic/Reykjavik
Italy	Europe/Rome Europe/San_Marino Europe/Vatican
Latvia	Europe/Riga
Liechtenstein	Europe/Vaduz
Lithuania	Europe/Vilnius
Luxembourg	Europe/Luxembourg
Malta	Europe/Malta
Moldova	Europe/Chisinau
Monaco	Europe/Monaco
Netherlands	Europe/Amsterdam
Norway	Europe/Oslo
Poland	Europe/Warsaw
Portugal	Europe/Lisbon Atlantic/Azores Atlantic/Madeira
Romania	Europe/Bucharest

Country	Time Zone
Russia	Europe/Kaliningrad Europe/Moscow Europe/Samara Asia/Yekaterinburg Asia/Oms Asia/Krasnoyarsk Asia/Irkutsk Asia/Yakutsk Asia/Vladivostok Asia/Magadan Asia/Kamchatka Asia/Anadyr
Spain	Africa/Ceuta Atlantic/Canary Europe/Madrid
Sweden	Europe/Stockholm
Switzerland	Europe/Zurich
Turkey	Europe/Istanbul Asia/Istanbul
Ukraine	Europe/Kiev Europe/Simferopol
Yugoslavia	Europe/Belgrade Europe/Ljubljana Europe/Sarajevo Europe/Skopje Europe/Zagreb

North America Time Zones

Valid time zones to specify for North America in the `UTCTimeZone` property.

Country	Time Zone
Anguilla	America/Anguilla
Antigua and Barbuda	America/Antigua
Bahamas	America/Nassau
Barbados	America/Barbados
Belize	America/Belize
Bermuda	Atlantic/Bermuda
British Virgin Islands	America/Tortola

Country	Time Zone
Canada	America/Dawson America/Dawson_Creek America/Edmonton America/Glace_Bay America/Goose_Bay America/Halifax America/Inuvik America/Iqaluit America/Montreal America/Nipigon America/Pangnirtung America/Rainy_River America/Rankin_Inlet America/Regina America/St_Johns America/Swift_Current America/Thunder_Bay America/Vancouver America/Whitehorse America/Winnipeg America/Yellowknife
Cayman Islands	America/Cayman
Costa Rica	America/Costa_Rica
Cuba	America/Havana
Dominica	America/Dominica
Dominican Republic	America/Santo_Domingo
El Salvador	America/El_Salvador
Grenada	America/Grenada

Country	Time Zone
Guadeloupe	America/Guadeloupe
Guatemala	America/Guatemala
Haiti	America/Port-au-Prince
Honduras	America/Tegucigalpa
Jamaica	America/Jamaica
Martinique	America/Martinique
Mexico	America/Cancun America/Chihuahua America/Ensenada America/Mazatlan America/Mexico_City America/Tijuana
Montserrat	America/Montserrat
Nicaragua	America/Managua
Panama	America/Panama
Puerto Rico	America/Puerto_Rico
St Kitts-Nevis	America/St_Kitts
St Lucia	America/St_Lucia
St Pierre and Miquelon	America/Miquelon
St Vincent and the Grenadines	America/St_Vincent
Turks and Caicos	America/Grand_Turk
United States	America/Chicago America/Denver America/Honolulu America/Los_Angeles America/New_York

Country	Time Zone
United States (Alaska)	America/Adak America/Anchorage America/Juneau America/Nome America/Yakutat
United States (exceptions)	America/Boise America/Detroit America/Indiana/Knox America/Indiana/Marengo America/Indiana/Vevay America/Indianapolis America/Louisville America/Menominee America/Phoenix
Virgin Islands	America/St_Thomas

South America Time Zones

Valid time zones to specify for South America in the `UTCTimeZone` property.

Country	Time Zones
Argentina	America/Buenos_Aires America/Catamarca America/Cordoba America/Jujuy America/Mendoza America/Rosario
Aruba	America/Aruba
Bolivia	America/La_Paz

Country	Time Zones
Brazil	America/Araguaina America/Belem America/Cuiaba America/Fortaleza America/Maceio America/Manaus America/Noronha America/Porto_Acre America/Porto_Velho America/Sao_Paulo
Chile	America/Santiago Pacific/Easter
Colombia	America/Bogota
Curacao	America/Curacao
Ecuador	America/Guayaquil Pacific/Galapagos
Falklands	Atlantic/Stanley
French Guiana	America/Cayenne
Guyana	America/Guyana
Paraguay	America/Asuncion
Peru	America/Lima
South Georgia	Atlantic/South_Georgia
Suriname	America/Paramaribo
Trinidad and Tobago	America/Port_of_Spain
Uruguay	America/Montevideo
Venezuela	America/Caracas

Starting the Open Adapter

Start an Open adapter instance via a `bootstrap` class.

Prerequisites

1. Install Java Runtime Environment version 1.7.0_1 or higher. To see if you have a suitable version of Java installed, go to *http://www.java.com/en/download/installed.jsp*.
2. Set the JAVA_HOME environment variable to the root of the JRE 1.7.0_1 directory.

Task

Start the Open adapter using a `bootstrap` class that reads the configuration file and starts the adapter components:

```
java - Xmx768M –cp ClassPath Bootstrap PropertyFile
AdapterName
```

where:

- **`Xmx768M`** – Java parameter specifying the size of the memory heap. You can increase the memory size from 768 for adapter configurations that require more memory.
- **`ClassPath`** – Java class path containing the JAR files or classes required by the Open adapter. This includes third-party JAR files. Refer to the component property files within the `examples` directory for more info on the class path used.
- **`Bootstrap`** – the adapter bootstrap and a Java class that can be run as a command line program. The name of the class provided is `org.openadapter.adapter.RunAdaptor`.
- **`PropertyFile`** – name of the properties file containing the component configuration for the adapter `<adapterName>.props`.
- **`AdapterName`** – name of the adapter to start. When creating adapter configurations, SAP recommends that you provide adapters with descriptive names to simplify identifying and monitoring adapter processes.

Monitoring the Open Adapter

Monitor a running adapter using RemoteControl and RemoteLogger interfaces.

RemoteControl and RemoteLogger are optional adapter implementations. Specify the interfaces in the adapter properties.

A running adapter contains a controller that supports a dynamic control interface, that can be invoked by a RemoteControl. A RemoteControl provides an interface for operators to communicate with running adapters to establish the current status and resolve problems. To specify a RemoteControl, use this properties syntax:

```
adapterName. Controller.RemoteControl.ClassName = Class
```

Remote logging filters the log output of the adapter and generates alerts or messages. You can configure remote loggers to send all log lines with a specific log level such as FATAL, WARN,

and so on. You can also use a regular expression to pattern match for explicit errors. To specify a RemoteLogger, use this syntax:

```
adapterName. Logging.RemoteLogger.ClassName = Class
```

SAP provides some standard implementations for remote control and remote logging.

Table 7. Remote Control and Remote Logging Interface Implementations

Interface	Description
HTTPRemote-Control	Turns adapter into a Web server. Point a browser at the machine on which the adapter is running, and the port (default value is 80) on which the HTTPRemoteControl is listening, and HTTPRemote-Control returns a simple control interface.
RvRemote-Control	Allows you to use TIBCO Rendezvous to communicate with the adapter without knowing where the adapter is running.
JMXRemote-Control	A JMX-compliant remote control.
RMIRemote-Control	Registers as an RMI service and allows clients to support, administer, and configure an adapter using RMI.
RvRemoteLogger	Publishes log lines on a specified Rendezvous subject
MailRemote-Logger	Sends e-mail messages using the Java Mail API.

Remote Control Interface

The remote control implementation expects to receive requests as DataObjects that contain specific attributes.

Table 8. Remote Control Interface Attributes

Attribute	Description
Method	Name of the operation.
UserName	Name of the user sending the request.
HostName	Host name the request originates from.
Comment	Any comment.
ControllerName	Name of the Adapter Controller. For example, Adapter.Controller.
Password	If set, ensure this matches the `ControlPassword` property for the adapter.
Arg1…ArgN	Arguments for the method.

Table 9. Remote Control Methods (Operations)

Operation	Description
pause	Invokes `pause()` on all adapter components. No messages are processed until the adapter is told to resume.
resume	Invokes `resume()` on all adapter components. Messages are processed again.
terminate	Invokes `terminate()` on all adapter components. All source components inform the controller that they are exiting, and the controller then exits.
kill	Invokes `System.exit(0)`, which ends any controller processes.
logLines	Returns the last N lines from the adapter output logger. The OutputLogger caches the last N lines it writes. The default is 10, but you can change this by using the `LogLinesToCache` property.
status	Invokes `getStatus()`, which returns a string on all components and publishes a consolidated status message on the control interface reply subject. The control utility can then display the results. If you write your own component, you can override the `getStatus()` method.
setLogLevel	Invokes `setLogLevel(arg1,arg2)` on the adapter OutputLogger. `setLogLevel` assumes that arg1 and arg2 in the request DataObject are loglevel and scope. An example is INFO DEFAULT.
customControl	Assumes that arg1 in the request DataObject is the name of an adapter component. The remote control forwards the entire request DataObject to the component by calling `customControl()` on the component. You can edit this.

HTTPRemoteControl

The HTTPRemoteControl implementation provides an HTTP-based remote control, turning your adapter into a simple Web server.

The Remote control listens for HTTP requests on a defined port (the default is 80) and replies with a simple HTML interface. This interface represents a control panel, and the buttons generate HTTP get requests. The Remote control parses these URLs into remote control requests.

For adapter A, set:

```
A.Controller.RemoteControl.ClassName =
org...standard.HTTPRemoteControl
A.Controller.RemoteControl.HTTPPort = ?
A.Controller.RemoteControl.ControlPassword = ?
```

The `HTTPort` and `ControlPassword` properties are optional. They default to port 80 and no password.

To test this remote control, type this URL into a browser: `http://<hostname>:<port>`

You see a control panel that supports the dynamic control interface. The control interface is based on parsing the URL so that you may cut and paste the URLs and use them on existing Web sites. Add `&reply=false` to the URL to disable the control interface in the reply.

The syntax for the URL is:

```
 http:// HostName :Port/ ?name= ControllerName &method= Method
&password= ControlPassword &arg1= Arg-Value ... &argN= ArgValue
&reply={true|false}
```

The HTTPRemoteControl parses this URL and creates a request DataObject, which the AbstractRemoteControl processes.

MailRemoteLogger

The MailRemoteLogger implementation uses the Java Mail API.

The CLASSPATH requires `mail.jar` and `activation.jar`. For adapter A, set:

```
A.Logging.RemoteLogSetting = FATAL A.Logging.RemoteLogger.ClassName
=
org.openadapter.adapter.mail.MailRemoteLogger
A.Logging.RemoteLogger.Mailhost = mailhost@foo.com
A.Logging.RemoteLogger.To = fred@openadaptor.org
A.Logging.RemoteLogger.FilterPattern = failed to connect
```

Note: `FilterPattern` is an optional property but requires a regular expression. Refer to Java documentation for additional properties.

The standard Open Adapter Controller offers an OASecurityManager interface that is responsible for all security related issues. Select an implementation of OASecurityManager by setting the controller property `SecurityManager.ClassName` in the controller property file.

PasswordEncryptor

The PasswordEncryptor component ensures that there are no plain text passwords in the Open adapter components.

The Event Stream Processor Extension for Open adapter provides sample keystores with the pairs of private and public keys. The default location of keystores is `$ESP_HOME/adapters/esp_open/lib/security`. There are three samples:

- `jksKeyStore` – a Java native keystore containing an RSA key pair generated with 512 encryption strength. The password for keystore is "changeit", and the key pair alias is "adaptor".
- `pkcs8KeyStore.der` – a keystore in the form PKCS#8 standard, and encoded using DER. It does not expect a password and alias. It contains an RSA key pair, and is generated using 512 encryption strength.
- `pkcs12KeyStore.p12` – keystore in the form PKCS#12 standard. The password is "changeit" and alias is"adaptor".

Note: The keystores above are samples only. In a production system, use your own keys.

The Open adapter offers a simple tool to encrypt password strings. In `$ESP_HOME/adapters/esp_open/bin`, the `pwdenc.sh` and `pwdenc.bat` files allow you to encrypt passwords. The tool requires two parameters:

- **-t** – the type of keystore. Valid values are JKS, PKCS8, and PKCS12.
- **- k** – keystore location.

If you provide no settings, the tool uses these default values:

```
pwdenc -t JKS -k ../lib/security/jksKeyStore.der
```

Depending on the keystore type, the tool asks further questions. Encrypted passwords are stored in the `encrytedPwd.txt` file of the directory where the shell script is executed. For example, `$ESP_HOME/adapters/esp_open/bin`. The string is also encoded using base64 algorithm. A limitation is that all characters should be in one line of the adapter property file. Passwords in encrypted form should be copied to the related password field of the component in the adapter property file.

Property	Description
`KeyStore`	(Required) Location of the keystore file.
`KeyStoreType`	(Optional) The standard used to store the Keystore file. Valid values are: JKS (default), PKCS8, PKCS12.
`KeyAlias`	(Optional) If keystore type is JKS or PKCS12, provide an alias name for the key pair. This property is not used in PKCS8.
`KeyStorePassword`	(Optional) If keystore type is JKS or PKCS12, provide a password. This property is not used in PKCS8.

Generating Self-Signed RSA Keys Using Java Keytool

Use the sample `jksKeyStore` file in the `$ESP_HOME/adapters/esp_open/lib/security` directory to generate self-signed RSA keys using Java keytool.

In a command prompt, execute:

```
keytool -genkey -keyalg rsa -keysize 512 -alias adaptor -keystore
jksKeyStore
```

Generating Self-Signed RSA Keys Using OpenSSL

Use the `PKCS12 Keystore` file in the `$ESP_HOME/adapters/esp_open/lib/security` directory to generate self-signed RSA keys using OpenSSL.

1. Generate CA private key.
   ```
   openssl genrsa -rand -des3 -out ca.key 512
   ```
2. Use that key to create the CA certificate.
   ```
   openssl req -new -x509 -days 365 -key ca.key -out ca.pem -outform
   PEM
   ```
3. Export the CA certificate so it can be imported into clientTrustStore.
   ```
   openssl x509 -in ca.pem -out caCert.pem -outform PEM -signkey
   ca.key
   ```
4. Generate the server private key.
   ```
   openssl genrsa -rand -des3 -out server.key 512
   ```
5. Create a server certificate.
   ```
   openssl req -new -days 365 -key server.key -out server.crs
   ```
6. Sign the server certificate with your CA certificate.
   ```
   openssl ca -in server.crs -out signedServerCert.pem -keyfile
   ca.key -cert caCert.pem
   ```
7. Export the certificate to PKCS#12 format so it can be imported to Queue Manager store.
   ```
   openssl pkcs12 -export -in signedServerCert.pem -out
   pkcs12KeyStore.p12 -inkey server.key -name adaptor
   ```

Generating Self-Signed RSA Keys Using OpenSSL (PKCS8 Keystore)

Use the sample `pkcs8KeyStore.der` file in the `$ESP_HOME/adapters/esp_open/lib/security` directory to generate self-signed RSA keys using OpenSSL.

In a command prompt, enter:

```
openssl pkcs8 -nocrypt -in server.key -out pkcs8KeyStore.der -
outform DER -topk8
```

Examples

Different Open adapter components.

Example: Using the AsapSink Component

Associate the FilePollSource (reader) component with the AsapSink (writer) component. The FilePollSource component reads records from file on the disk (`insert.txt`) and transfers those records to the AsapSink component. The AsapSink component then publishes those records to Event Stream Processor.

1. Set the username and password in the example environment:

Operating System	Step
UNIX	**1.** Edit the `set_example_env.sh` script **2.** Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	**1.** Edit the `set_example_env.bat` script **2.** Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

2. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

3. Start **esp_subscriber** to subscribe to the project that is running on the cluster.

Operating System	Step
UNIX	Open a terminal window and enter: `subscribe.sh`
Windows	Open a command window and enter: `subscribe.bat`

4. Start the FilePollSource and AsapSink components.

Operating System	Step
UNIX	Open a terminal window and enter: `./fileToAsap.sh`

Operating System	Step
Windows	Open a command window and enter: `fileToAsap.bat`

The FilePollSource components reads from the `insert.txt` file and passes records to AsapSink to publish to the Server.

5. In the `fileToAsap.props` file, change `adaptor.FILESOURCE.InputFileName` to `insert_withNULL.txt`, and run again.

See also

- *AsapSink Properties* on page 482

Example: Using the AsapSource Component

Associate the AsapSource (reader) component with the FileSink (writer) component. AsapSource reads records from Event Stream Processor and passes those to FileSink, which then writes those records to the `out.txt` file.

1. Set the username and password in the example environment:

Operating System	Step
UNIX	1. Edit the `set_example_env.sh` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	1. Edit the `set_example_env.bat` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

2. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Operating System	Step

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

3. Start **esp_subscriber** to subscribe to the project that is running on the cluster.

Operating System	Step
UNIX	Open a terminal window and enter: `subscribe.sh`
Windows	Open a command window and enter: `subscribe.bat`

4. Starts the AsapSource and FileSink components.

Operating System	Step
UNIX	Open a terminal window and enter: `./asapToFile.sh`
Windows	Open a command window and enter: `asapToFile.bat`

5. Upload data from the `esp_insert.txt` file to the Server.

Operating System	Step
UNIX	Open a terminal window and enter: `./esp_upload.sh`
Windows	Open a command window and enter: `esp_upload.bat`

AsapSource reads this published data from the Server and passes it to FileSink, which writes it to the `out.txt` file.

See also

- *AsapSource Properties* on page 477

Example: Using the BeanShellPipe Component

You can use the BeanShellPipe component between the AsapSource and FileSink components. BeanShellPipe executes some commands in shell after it receives data from AsapSource, and before publishing data to FileSink.

1. Set the username and password in the example environment:

Operating System	Step
UNIX	1. Edit the `set_example_env.sh` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	1. Edit the `set_example_env.bat` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

2. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

3. Start **esp_subscriber** to subscribe to the project that is running on the cluster.

Operating System	Step
UNIX	Open a terminal window and enter: `subscribe.sh`
Windows	Open a command window and enter: `subscribe.bat`

4. Start the AsapSource, FileSink and BeanShellPipe components.

Operating System	Step
UNIX	Open a terminal window and enter: `./asapToFile.sh`
Windows	Open a command window and enter: `asapToFile.bat`

5. Upload data from the `esp_insert.txt` file to Server.

Operating System	Step
UNIX	Open a terminal window and enter: `./esp_upload.sh`
Windows	Open a command window and enter: `esp_upload.bat`

AsapSource reads this data and passes it to BeanShellPipe, which then passes it to FileSink, which writes it to the `out.txt` file. BeanShellPipe outputs the text to the command prompt.

See also

- *BeanShellPipe Properties* on page 486

Example: Using the JDBCLookupPipe Component

AsapSource reads data from Event Stream Processor and passes it to the JDBC lookup pipe. If required, the JDBC lookup pipe modifies the values of the 'charfield' column by using 'replaceValue1', and passes that data to FileSource, which then outputs that data to the `file out.txt` file.

1. Create a table and then create data into the table. For example, for a DB2 database, run the **createTable_DB2.sql** script.

   Modify this script to use it for any other databases.
2. Update the DB properties in the `JdbcLookupPipe.props` file to point to the required database instance.
3. Update the `JdbcLookupPipe.bat` or `JdbcLookupPipe.sh` script, and add JDBC driver JARs in the class path.
4. Set the username and password in the example environment:

Operating System	Step
UNIX	**1.** Edit the `set_example_env.sh` script **2.** Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	**1.** Edit the `set_example_env.bat` script **2.** Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

5. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

6. Start **esp_subscriber** to subscribe to the project that is running on the cluster.

Operating System	Step
UNIX	Open a terminal window and enter: `subscribe.sh`
Windows	Open a command window and enter: `subscribe.bat`

7. Start the AsapSource, FileSink, and JDBCLookupPipe components.

Operating System	Step
UNIX	Open a terminal window and enter: `./JdbcLookupPipe.sh`

Operating System	Step
Windows	Open a command window and enter: `JdbcLookupPipe.bat`

8. Upload data to the Server.

Operating System	Step
UNIX	Open a terminal window and enter: `./esp_upload.sh`
Windows	Open a command window and enter: `esp_upload.bat`

AsapSource reads this data (records) and passes it on to JDBCLookupPipe, which modifies the records according to data available and reference data from the database tables. JDBCLookupPipe then passes that data to FileSink, which then writes the records to file.

- Table "test1" contains data "col1='AttributeKey'" and "col2='replaceValue1'". 'KeyDbCol1' is col1 in the props file, therefore, col1 column contains attribute keys.
- These attribute keys are present in the incoming record column 'textfield'.
- To replace the 'charfield' column value of a record to 'replaceValue1', include 'AttributeKey' as a value in 'textfield' column of a record

See the `esp_insert.txt` file for more details. Records that do not have 'AttributeKey' as the 'textfield' column value are not modified.

9. See contents of the `out.txt` file.

Charfield data for some of the records is updated to 'replaceValue1' value.

See also

- *JDBCLookupPipe Properties* on page 488

Example: Using the MultiFlatXmlStringReader Component

Associate the MultiFlatXmlStringReader component with the FilePollSource component so that it can read records in XML format and pass them to AsapSink, which publishes them to the Server.

1. Set the username and password in the example environment:

Operating System	Step
UNIX	**1.** Edit the `set_example_env.sh` script **2.** Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	**1.** Edit the `set_example_env.bat` script **2.** Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

2. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

3. Start **esp_subscriber** to subscribe to the project that is running on the cluster.

Operating System	Step
UNIX	Open a terminal window and enter: `subscribe.sh`
Windows	Open a command window and enter: `subscribe.bat`

4. Start the MultiFlatXmlStringReader, FilePollSource, and AsapSink components.

Operating System	Step
UNIX	Open a terminal window and enter: `./MultiFlatXMLStringReader.sh`

Operating System	Step
Windows	Open a command window and enter: `MultiFlatXMLStringReader.bat`

FilePollSource reads XML formatted data from the `insert.xml` file using MultiFlatXmlStringReader, and passes it to AsapSink, which publishes data to the Server.

See also

- *MultiFlatXmlStringReader Properties* on page 490

Example: Using the SpPersistentSubscribeSource Component

The SpPersistentSubscribeSource component subscribes to the Server using persistent subscribe (stores subscribed records until it processes them, and then deletes them).

To implement this, a log stream (Stream1_log) and truncate stream (TruncateStream1) are created for stream "Stream1". Stream1_log stores the data and TruncateStream1 has two columns: primary key and sequence number. See the `model.ccl` in the `bin` folder for more details.

Incoming records are transferred to Stream1_log with an additional sequencenumber column. Once records are processed from Stream1_log, the last sequence number is published to TruncateStream1. All records with sequence numbers smaller than or equal to the published sequencenumber are then deleted from the Stream1_log.

1. Set the username and password in the example environment:

Operating System	Step
UNIX	1. Edit the `set_example_env.sh` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	1. Edit the `set_example_env.bat` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

2. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

3. Start **esp_subscriber** to subscribe to Stream1 of the project running on the cluster above.

Operating System	Step
UNIX	Open a terminal window and enter: `./esp-subscribe-Stream1.sh`
Windows	Open a command window and enter: `esp-subscribe-Stream1.bat`

4. Subscribe to the log stream, Stream1_Log.

Operating System	Step
UNIX	Open a terminal window and enter: `./esp-subscribe-Stream1_log.sh`
Windows	Open a command window and enter: `esp-subscribe-Stream1_log.bat`

5. Subscribe to the log stream, Truncate_stream1.

Operating System	Step
UNIX	Open a terminal window and enter: `./esp-subscribe-TruncateStream1.sh`
Windows	Open a command window and enter: `esp-subscribe-TruncateStream1.bat`

6. Start the SpPersistentSubscribeSource and FileSink components.

Operating System	Step
UNIX	Open a terminal window and enter: `./asapToFile.sh`
Windows	Open a command window and enter: `asapToFile.bat`

7. Upload data from the `esp_insert.txt` file to the Server.

Operating System	Step
UNIX	Open a terminal window and enter: `./esp_upload.sh`
Windows	Open a command window and enter: `esp_upload.bat`

SpPersistentSubscribeSource subscribes to the Server and Stream1_log, and passes these records to FileSink, which writes these records to the `out.txt` file. All the subscription script files show the respective subscriptions.

See also

- *SpPersistentSubscribeSource Properties* on page 480

Example: Using the WSSink Component

Use the `WsSink.props` file to associate the WSSink component with the AsapSource component. AsapSource reads data from the Server and passes records to WSSink, which publishes these records to a Web service. A second `WsSource.props` file associates the WSSource component with FileSink. WSSource reads published records to the Web service and passes them to FileSink, which writes those records to file.

1. Set the username and password in the example environment:

Operating System	Step
UNIX	1. Edit the `set_example_env.sh` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

Operating System	Step
Windows	**1.** Edit the `set_example_env.bat` script **2.** Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

2. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

3. Start **esp_subscriber** to subscribe to the project that is running on the cluster.

Operating System	Step
UNIX	Open a terminal window and enter: `subscribe.sh`
Windows	Open a command window and enter: `subscribe.bat`

4. Call the **esp_upload** command, and upload records to the Server.

Operating System	Step
UNIX	Open a terminal window and enter: `./upload.sh`
Windows	Open a command window and enter: `upload.bat`

5. Start the WSSource and FileSink components, and the Web service they are connected to.

Operating System	Step
UNIX	Open a terminal window and enter: `./WsSource.sh`
Windows	Open a command window and enter: `WsSource.bat`

6. Start the WSSink with AsapSource components.

Operating System	Step
UNIX	Open a terminal window and enter: `./WsSink.sh`
Windows	Open a command window and enter: `WsSink.bat`

The `out_wssource.txt` file now contains records. WSSink reads the uploaded records and passes them to the Web service. WSSource reads these records and passes them to FileSink, which writes them to the `out_wssource.txt` file.

See also

- *WSSink Properties* on page 485

Example: Using the WSSource Component

Use the WSSource component to publish data to a Web service using a Web service client, such as soapUI. WSSource reads records from the Web service and passes them to FileSink, which writes those records to file.

1. Start the Web service that WSSource is connected to.

Operating System	Step
UNIX	Open a terminal window and enter: `./WsSource.sh`
Windows	Open a command window and enter: `WsSource.bat`

2. Using any SOAP client, try calls given in the `readme.txt` file in `WSSource` folder, which is located within the `examples` folder. For example, use SOAP client soapUI. This publishes data to a Web service using Web service client. WSSource reads records from the Web service, passes them to FileSink, which writes the records to file.

Example: Using the XPathMultiTypeXmlReader Component

Associate the XPathMultiTypeXmlReader component with the FilePollSource component, which reads XML formatted data, and with the AsapSink component, which publishes records to the Server. Define parsing rules in the `XPathXmlStreamReader.props` file, and use these rules to parse XML records being read from file.

1. Set the username and password in the example environment:

Operating System	Step
UNIX	1. Edit the `set_example_env.sh` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	1. Edit the `set_example_env.bat` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

2. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

3. Start **esp_subscriber** to subscribe to the project that is running on the cluster.

Operating System	Step
UNIX	Open a terminal window and enter: `subscribe.sh`
Windows	Open a command window and enter: `subscribe.bat`

4. Start the FilePollSource (with XPathXmlStreamReader) and AsapSink components.

Operating System	Step
UNIX	Open a terminal window and enter: `./XPathMultiTypeXmlReader.sh`
Windows	Open a command window and enter: `XPathMultiTypeXmlReader.bat`

Data from the `insert.xml` file publishes to the Server.

See also

- *XPathMultiTypeXmlReader Properties* on page 495

Example: Using the XPathXmlStreamReader Component

Use XPathXmlStreamReader with FilePollSource, which reads XML data and parses it using XPath rules. Then pass the records to AsapSink, which publishes them to the Server.

1. Set the username and password in the example environment:

Operating System	Step
UNIX	1. Edit the `set_example_env.sh` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	1. Edit the `set_example_env.bat` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

2. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Operating System	Step

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

3. Start **esp_subscriber** to subscribe to the project that is running on the cluster.

Operating System	Step
UNIX	Open a terminal window and enter: `subscribe.sh`
Windows	Open a command window and enter: `subscribe.bat`

4. Start the FilePollSource (with XPathXmlStreamReader) and AsapSink components.

Operating System	Step
UNIX	Open a terminal window and enter: `./XPathXmlStreamReader.sh`
Windows	Open a command window and enter: `XPathXmlStreamReader.bat`

Data from the `insert.xml` file publishes to the Server.

See also

- *XPathXmlStreamReader Properties* on page 492

Example: Using the XPathXmlStringWriter Component

The XPathXmlStringWriter component writes XML formatted data using XPath rules, and is used with the FileSink component. Associate the AsapSource component, which reads data from the Server and passes it to FileSink, with the FileSink component, which writes out the XML data using XPathXmlStringWriter.

1. Set the username and password in the example environment:

Operating System	Step
UNIX	1. Edit the `set_example_env.sh` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	1. Edit the `set_example_env.bat` script 2. Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

2. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

3. Start **esp_subscriber** to subscribe to the project that is running on the cluster.

Operating System	Step
UNIX	Open a terminal window and enter: `subscribe.sh`
Windows	Open a command window and enter: `subscribe.bat`

4. Start the AsapSource and FileSink (with XPathXmlStringWriter) components.

Operating System	Step
UNIX	Open a terminal window and enter: `./XPathXmlStringWriter.sh`

Operating System	Step
Windows	Open a command window and enter: `XPathXmlStringWriter.bat`

Data from the `insert.xml` file publishes to the Server.

See also

- *XPathXmlStringWriter Properties* on page 497

Random Tuples Generator Input Adapter

Adapter type: randomtuplegen_in. The Random Tuples Generator adapter generates random tuples according to the given schema and sends the rows to the stream.

A tuple is an ordered list of elements, or in other words, a row of data. A row that has two column values is a 2-tuple, and generally, a row that has n column values is an n-tuple. The adapter is primarily used for prototyping and basic testing of Event Stream Processor. You can edit both the schema and configuration file.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Rate	Property ID: **Rate** Type: `uint` (Optional) Number of rows generated per second. Must be between 0 (inclusive) and 1,000,000 (inclusive). If **Rate** is negative, the adapter stops and returns a fatal error message to the Server. If **Rate** is larger than the maximum value, it resets to the maximum and reports a warning message to the Server. If **Rate** property is blank, it resets to the default value and reports a message. Default value is 100.

Property Label	Description
Row Count	Property ID: **RowCount** Type: `uint` (Optional) Specifies number of generated rows. Must be between 0 and 2,000,000,000 inclusive. If **RowCount** is negative, the adapter stops and returns a fatal error message to the Server. If **RowCount** is larger than the maximum value, it resets to the maximum and reports a warning message to the Server. If **RowCount** property is blank, it resets to the default value and reports an information message to the Server. Default value is 0, which represents an infinite number of rows.
Timestamp Base	Property ID: **TimestampBase** Type: `string` (Optional) Initial time for message timestamps. The supported format of **TimestampBase** is %Y-%m-%dT%H:%M:%S. If **TimestampBase** is blank, it resets to default value and the adapter initializes immediately. Similarly, the adapter initializes immediately if the **TimestampBase** value is earlier than current time. If **TimestampBase** value is later than current time, the adapter sleeps until then. If **TimestampBase** has an invalid timestamp format, the adapter stops and returns a fatal error message to the Server. Default value is the current time.
Date Format	Property ID: **DateFormat** Type: `string` (Optional) Format string for parsing date values. Default value is `%Y-%m-%d %H:%M:%S`.

Property Label	Description
Timestamp Format	Property ID: **TimestampFormat** Type: `string` (Optional) Format string for parsing date values. Default value is `%Y-%m-%d %H:%M:%S`.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.
Block Size	Property ID: **blockSize** Type: `int` (Advanced) The number of records to block into one pseudo-transaction. Default value is 0.
Use Envelopes	Property ID: **useEnvelopes** Type: `boolean` (Advanced) Specify the block type the adapter uses to pass data to the engine. If you specify a **blockSize** property greater than zero, by default, the adapter packages rows into transaction blocks to send to the engine. To get the adapter to package rows into envelope blocks instead, set this property to true. Default value is false.

The data this adapter generates is not evenly distributed across the range of possible values for each datatype. This table shows the value range generated for each datatype:

Datatype	Range of values
`boolean`	true/false
`integer`	0 .. 99 inclusive

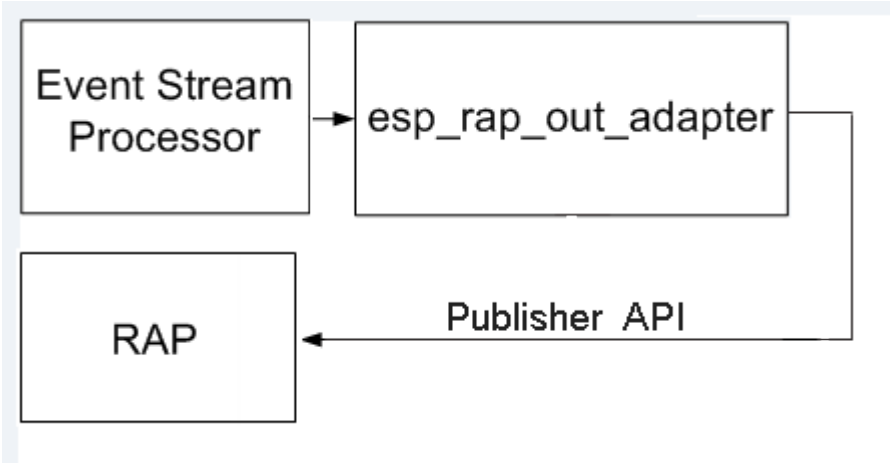
Datatype	Range of values
long	0 .. 99 inclusive
float	0.0 .. 10.0 exclusive (should never get 10.0)
interval	0 .. 9 inclusive
timestamp	Current time with milliseconds
string	2 characters from the following ranges a..z, A..Z, 0..9
binary	2 bytes each with range 0..255
money	0.0000 to 3.2767
money1	0.0 to 3276.7
money2	0.00 to 327.67
money3	0.000 to 32.767
money4	0.0000 to 3.2767
money5	0.00000 to 0.32767
money6	0.000000 to 0.032767
money7	0.0000000 to 0.0032767
money8	0.00000000 to 0.00032767
money9	0.000000000 to 0.000032767
money10	0.0000000000 to 0.0000032767
money11	0.00000000000 to 0.00000032767
money12	0.000000000000 to 0.000000032767
money13	0.0000000000000 to 0.0000000032767
money14	0.00000000000000 to 0.00000000032767
money15	0.000000000000000 to 0.000000000032767
bigdatetime	Current time with microseconds
date	Current time with seconds

Note: Values are not necessarily evenly distributed within these ranges.

RAP Output Adapter

Adapter type: esp_rap_out_adapter. The SAP Sybase Event Stream Processor RAP adapter is an external adapter that publishes data from Event Stream Processor to SAP Sybase RAP using the C SDK.

The RAP adapter supports only Solaris and Linux platforms.



Each stream you want to publish to the RAP platform requires its own adapter. For example, to publish three streams to RAP, configure three adapters. Start and stop these adapters separately.

Note: Since RAP accepts only inserts as input, deletes from the Server are dropped, and updates are converted to inserts.

Start Command

Use the **start.sh** script to publish data from the Server to RAP.

Syntax

To use the `start.sh` script, create the `$ESP_HOME/adapters/rap_out/lib` directory and copy the platform specific `libpublisher.so` to it. Verify that `libodbc.so` is installed with your RAP installation, and if it is not present, install it. Create a symbolic link with the `libodbc.so.1` file from your RAP installation to `libodbc.so` version 1.0.0, and put this file in the `$ESP_HOME/adapters/rap_out/lib` directory.

The Event Stream Processor installer installs the `$ESP_HOME/adapters/rap_out/lang` directory and includes these files:

- `demofeedhandler_en_US.properties`
- `publisher_en_US.properties`
- `templateprocessor_en_US.properties`
- `util_en_US.properties`

If you have more recent versions of these files, copy them over to the `$ESP_HOME/adapters/rap_out/lang` directory. You can also update the `$ESP_HOME/adapters/rap_out/templates` directory if you have more recent versions of its contents.

Note: The `start.sh` is an implementation of the **esp_rap_out_adapter** command.

```
esp_rap_out_adapter -f configFile -t templateDir -p
publisherConfigDir &
```

Required Arguments

Argument	Description
`-f configFile`	The full path of the configuration file that specifies the streams from Event Stream Processor that provide data to RAP. Default value is `../config/espfeedhandler.xml`
`-t templateDir`	The full path to the directory containing the RDS templates that map columns in the Event Stream Processor streams to tables and columns of RAP. Default value is `../templates`
`-p PublisherConfigDir`	The full path to the directory containing the publisher configuration file, which defines the multicast address used by the RAP subscriber. Default value is `../config`
`&`	(Optional) Run the adapter in the background.

See also

- *Starting the RAP Adapter* on page 560

Stop Command

If the RAP adapter is running in the foreground, you can stop it by pressing Ctrl+C.

If the RAP adapter is running in the background, you can stop it by entering `ps -eaf | grep esp_rap_out` to get the process ID of the adapter and enter `kill -15 processID` in the command line.

See also

- *Stopping the RAP Adapter* on page 561

Datatype Mapping for the RAP Adapter

Event Stream Processor datatypes map to RAP, ASE, and IQ datatypes.

The RAP adapter does not support the Event Stream Processor `binary` and `boolean` datatypes.

ESP Datatype	RAP Datatype	ASE Datatype	IQ Datatype
integer	sint32	int	int
long	sint64	bigint	bigint
float	decimal(p,s)	decimal(p,s) or numeric(p,s)	decimal(p,s) or numeric(p,s)
interval	sint64	bigint	bigint
date	datetime	datetime	timestamp
timestamp	datetime	datetime	timestamp
bigdatetime	datetime2	bigdatetime	timestamp
money	decimal(19,4)	numeric(19,4)	numeric(19,4)
money(1)	decimal(19,1)	decimal(19,1)	decimal(19,1)
money(2)	decimal(19,2)	decimal(19,2)	decimal(19,2)
money(3)	decimal(19,3)	decimal(19,3)	decimal(19,3)
money(4)	decimal(19,4)	decimal(19,4)	decimal(19,4)
money(5)	decimal(19,5)	decimal(19,5)	decimal(19,5)
money(6)	decimal(19,6)	decimal(19,6)	decimal(19,6)
money(7)	decimal(19,7)	decimal(19,7)	decimal(19,7)
money(8)	decimal(19,8)	decimal(19,8)	decimal(19,8)
money(9)	decimal(19,9)	decimal(19,9)	decimal(19,9)
money(10)	decimal(19,10)	decimal(19,10)	decimal(19,10)
money(11)	decimal(19,11)	decimal(19,11)	decimal(19,11)
money(12)	decimal(19,12)	decimal(19,12)	decimal(19,12)

ESP Datatype	RAP Datatype	ASE Datatype	IQ Datatype
money(13)	decimal(19,13)	decimal(19,13)	decimal(19,13)
money(14)	decimal(19,14)	decimal(19,14)	decimal(19,14)
money(15)	decimal(19,15)	decimal(19,15)	decimal(19,15)
string	string	varchar(n)	varchar(n)

Configuration

Configuration information for the RAP adapter.

To configure the RAP adapter, you need:

- An adapter configuration file
- A publisher file
- An RDS template file

Adapter Configuration File

Use the `espfeedhandler.xml` configuration file to specify which Event Stream Processor streams provide data to RAP.

Syntax

```
<?xml version="1.0" encoding="UTF-8"?>
<ESPFeedHandler>
    <Logger>
        <LogLevel>warning</LogLevel>
        <LogFile>ESPFeedHandler.log</LogFile>
    </Logger>
    <MainCommandControlServer>
    <MainCCHost>127.0.0.1</MainCCHost>
    <MainCCPort>55555</MainCCPort>
    <Workspace>workspace1</Workspace>
    <Project>project1</Project>
        </MainCommandControlServer>
    <StandbyCommandControlServer>
        <StandbyCCHost/>
        <StandbyCCPort/>
    </StandbyCommandControlServer>
    <UseEncryption/>
    <ESPAuthentication>
        <User></User>
        <Password></Password>
    </ESPAuthentication>
    <Subscription>        <ProjectionSQL></ProjectionSQL>
<SubscriptionStream>ds1</SubscriptionStream>
        <RAPMessageType>69</RAPMessageType>
    </Subscription>
</ESPFeedHandler>
```

Table 10. XML Elements

Element	Description
ESPFeedHandler	(Required) The root element of the file.
Logger	(Required) The root element for logging activities settings.
LogLevel	(Required) The level of logging. Valid values are: • `error` – logs only errors. • `warning` – logs warnings and errors. • `info` – logs informational messages and messages logged at the warning level. • `debug` – logs debugging messages and messages logged at the info level.
LogFile	(Required) The name and location (relative or absoulte path) of the log file.
MainCommandControlServ-er	(Required) The root element of the connection information for the main command control server.
MainCCHost	(Required) The IP address of the main command control server.
MainCCPort	(Required) The port for the main command control server.
Workspace	(Required) The workspace in the cluster that contains the stream.
Project	(Required) The project that contains the stream.
StandbyCommandControl-Server	(Optional) The root element of the connection information for the standby command control server.
StandbyCCHost	(Optional) The IP address of the standby command control server. StandbyCCHost is required for HA.
StandbyCCPort	(Optional) The port for the standby command control server. Stand-byCCPort is required for HA.
UseEncryption	(Optional) The root element for SSL encryption for communications between the RAP adapter and Event Stream Processor.
ESPAuthentication	(Required) The root element for the information necessary for au-thenticating the connection to the command control server.
User	(Optional) The user name to connect to the command control server. If authentication is enabled the user is required.

Element	Description
Password	(Optional) The password to use to connect to the command control server (in encrypted form).If authentication is enabled the password is required.
Subscription	(Required) The root element for information about subscription to a stream.
ProjectionSQL	(Optional) The SQL query projection to be used on the stream.
SubscriptionStream	(Optional) The name of the stream to which you want to subscribe.
RAPMessageType	(Required) The RAP message type number. This is the same number used in the RDS template.

Publisher File

The `publisher.xml` file configures the RAP publisher in Event Stream Processor. It specifies the log file, administration channel, and data stream channels.

For more information about how to configure the RAP publisher, see *Configuring a Publisher* in the *RAP - The Trading Edition R4.1 Operations Console Users Guide*.

Syntax

```
<?xml version="1.0" encoding="UTF-8"?>
<Publisher>
<Logger>
    <LogLevel>...</LogLevel>
    <LogFile>...</LogFile>
</Logger>
<NumMessageBuffers>...</NumMessageBuffers>
<NumPacketBuffers>...</NumPacketBuffers>
<MessageFlushInterval>...</MessageFlushInterval>
<LatencyCheckInterval>...</LatencyCheckInterval>
<AdminChannel>
 <LocalInterface>...</<LocalInterface>
    <AdminPort>...</AdminPort>
    <MaxConnections>...</MaxConnections>
</AdminChannel>
<ResendChannel>
    <ResendPort>...</ResendPort>
</ResendChannel>
<TimeToLive>...</TimeToLive>
<DataStreamChannelList>
    <DataStreamChannel>
        <ChannelName>...</ChannelName>
        <LocalInterface>...</LocalInterface>
        <IPAddress>...</IPAddress>
        <Port>...</Port>
    </DataStreamChannel>
    <DataStreamChannel>
```

```
        <ChannelName>...</ChannelName>
        <LocalInterface>...</LocalInterface>
        <IPAddress>...</IPAddress>
        <Port>...</Port>
    </DataStreamChannel>
</DataStreamChannelList>
</Publisher>
```

Table 11. XML Elements

Element	Description
Publisher	Root element for the configuration file.
Logger	Contains settings for logging activities.
LogLevel	The level of logging. Valid values are: • Error – log only errors. • Warning – log warnings in addition to errors. • Info – log informational messages in addition to messages logged at the warning level. • Debug – log debugging messages in addition to messages logged at the info level.
LogFile	Name and location of the log file. The file name can be relative or a full path.
NumMessageBuffers	Number of message buffers. One message buffer is required for each message that is being simultaneously built. This setting can have a value from 1 to 65535, although the machine must have enough memory to hold the number of buffers specified.
NumPacketBuffers	Maximum number of packets to cache to satisfy requests by a subscriber to resend a packet. The number of packets is cached per data stream channel. The setting can have a value from 1 to 4294967296, although the machine must have enough memory to hold the number of packets specified. The number of buffers is allocated on initialization of the publisher.
MessageFlushInterval	Interval, in seconds, during which a partially filled message buffer must be idle before being sent on the network. This setting can have a value from 1 to 65535.
LatencyCheckInterval	Number of seconds after which to perform a latency check on a message. This setting can have a value from 1 to 65535.
AdminChannel	Information about the administration channel. This channel accepts requests for version information, statistics, and shutdown.

Element	Description
LocalInterface	Local interface that the publisher uses to monitor administrative requests.
AdminPort	Port number used by the UAF agent to communicate with the publisher. The publisher listens for incoming administration requests on this port.
MaxConnections	Determines the number of concurrent connections to the AdminChannel. Value must be in the range of 1 to 65535. Default value is 10
ResendChannel	Information about the resend channel. This channel listens for connections from subscribers, who open connections to publishers and issue requests to resend packets.
ResendPort	Port number used by subscribers to request resends of dropped network packets, and to time network latency between publisher and subscriber.
TimeToLive	The limit on the number of routing devices a message may pass through before expiring.
DataStreamChannelList	A list of data stream channel definitions. There can be up to 255 data stream channels.
DataStreamChannel	Contains information for one data stream channel. Each message sent by the publisher is sent in a network packet buffer over one of the defined channels. The publisher attempts to balance sending over all the channels while the system is under load.
ChannelName	Descriptive name for the channel, which is used to identify the channel when logging.
LocalInterface	IP address of a network interface on the local machine, which should be used for sending data.
IPAddress	UDP multicast address for sending the messages over the network.
Port	Port over which messages are sent using UDP multicasting.

RDS Template File

The RAP data stream (RDS) template file defines the structure of RAP message types in Event Stream Processor.

For more information on RAP Messages and Schemas, see *Customizing the RAP Messages and Schema* Chapter in the *RAP - The Trading Edition R4.1 Developers Guide*.

Syntax

```
<?xml version="1.0" encoding="UTF-8"?>

<Template xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="../template.xsd">
    <MessageDefnList>
        <MessageDefn>
            <MessageDesc>...</MessageDesc>
            <MessageType>...</MessageType>
            <DestTableName>...</DestTableName>
            <FieldDefnList>
                <FieldDefn>
                    <FieldName>...</FieldName>
                    <StringField/>
                    <DestColumnName>...</DestColumnName>
                    <Lookup>
                        <LookupTableName>...T</LookupTableName>
                        <LookupColumnName>...</LookupColumnName>
                        <LookupColumnReturn>...</LookupColumnReturn>
                    </Lookup>
                </FieldDefn>
            </FieldDefnList>
        </MessageDefn>
    </MessageDefnList>
</Template>
```

Table 12. XML Elements

Element	Description
Template	Root element for the template.
MessageDefnList	A list of one or more message definitions.
MessageDefn	Information that defines a single message type.
MessageDesc	Description of the type of market data message, for example, Stock Quote. This element is used only for descriptive purposes, and can contain any string.
MessageType	Unique number representing the type of market data message. This number must uniquely identify the message type across all message definitions within all templates. The value can contain any integer from 1 – 65535.
DestTableName	Name of the database table into which the message should be stored. There is one database table per message type. The value can contain any string.
FieldDefnList	A list of one or more field definitions.

Element	Description
FieldDefn	Information that defines a single field.
FieldName	Name of the field. The value can contain any string.
IntegerField	Indicates that the field is some type of integer datatype.
IntegerDataType	Datatype of an integer field. Valid values: `uint8`, `uint16`, `uint32`, `uint64`, `sint8`, `sint16`, `sint32`, or `sint64`.
DecimalField	Indicates that the field is a decimal value. The field definition can contain only one of: IntegerField, DecimalField, StringField, DateField, TimeField, DateTimeField, DateTime2Field, or Time2Field.
Precision	Precision of a decimal field. Maximum precision allowed is 38 digits.
Scale	Scale of a decimal field (the number of digits after the decimal point). Maximum scale can be no larger than the precision (38).
StringField	Indicates that the field is a string datatype.
DateField	Indicates that the field is a date datatype. The field definition can contain only one of: IntegerField, DecimalField, StringField, DateField, TimeField, DateTimeField, DateTime2Field, or Time2Field.
TimeField	Indicates that the field is a time datatype.
Time2Field	Indicates that the field is a bigtime datatype (granularity to 6 decimal places). The field definition can contain only one of: IntegerField, DecimalField, StringField, DateField, TimeField, DateTimeField, DateTime2Field, or Time2Field.
DateTimeField	Indicates that the field is a datetime datatype.
DateTime2Field	Indicates that the field is a bigdatetime (granularity to 6 decimal places). The field definition can contain only one of: IntegerField, DecimalField, StringField, DateField, TimeField, DateTimeField, DateTime2Field, or Time2Field.
DestColumnName	Name of the column into which the field data should be stored. There is one column per field. The value of this element can contain any string.
Lookup	(Optional) Indicates that the data in the field should be used as a lookup for another table. If this element does not appear in a field definition, then no lookup is required.
LookupTableName	Name of the table to use to look up a value.

Element	Description
LookupColumnName	Name of the column to use to look up a value.
LookupColumnReturn	Name of the column from which to return data when doing a lookup.

Example: Configuring the RAP Adapter

Set the configuration, publisher, and RDS template files to configure the RAP adapter for communication between RAP and Event Stream Processor.

1. Set the `$RAPOUT_HOME` environment variable to `$ESP_HOME/adapters/rap_out` directory.
2. Navigate to the `$RAPOUT_HOME` directory.
3. Create a project defining the streams you want to publish to RAP, and save it to a file named `model.ccl` in the `$ESP_HOME/bin` directory.
4. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

   ```
   cd %ESP_HOME%\cluster\nodes\node1
     %ESP_HOME%\bin\esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   %ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

   UNIX:

   a. Start the example cluster.

   ```
   cd $ESP_HOME/cluster/nodes/node1
     $ESP_HOME/bin/esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   $ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   $ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

5. In the `$RAPOUT_HOME/config` directory, modify the `espfeedhandler.xml` configuration file to specify which Event Stream Processor streams are providing data to RAP. For example, the file below configures the adapter to publish a single stream called Trades:

```
<?xml version="1.0" encoding="UTF-8"?>
<ESPFeedHandler>
        <Logger>
                <LogLevel>warning</LogLevel>
                <LogFile>ESPFeedHandler.log</LogFile>
        </Logger>
        <MainCommandControlServer>
       <MainCCHost>127.0.0.1</MainCCHost>
       <MainCCPort>19011</MainCCPort>
       <Workspace>w1</Workspace>
       <Project>p1</Project>
        </MainCommandControlServer>
        <StandbyCommandControlServer>
                <StandbyCCHost/>
                <StandbyCCPort/>
        </StandbyCommandControlServer>
        <UseEncryption/>
        <ESPAuthentication>
                <User></User>
                <Password></Password>
        </ESPAuthentication>
        <Subscription>
                <ProjectionSQL></ProjectionSQL>
                <SubscriptionStream>Trades</SubscriptionStream>
                <RAPMessageType>69</RAPMessageType>
        </Subscription>
</ESPFeedHandler>
```

6. In `$RAPOUT_HOME/config`, modify the existing publisher file to specify the multicast address used by the RAP subscriber. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Publisher>
        <Logger>
                <LogLevel>debug</LogLevel>
                <LogFile>Publisher.log</LogFile>
        </Logger>
    <NumMessageBuffers>1</NumMessageBuffers>
    <NumPacketBuffers>10000</NumPacketBuffers>
        <MessageFlushInterval>1</MessageFlushInterval>
        <LatencyCheckInterval>30</LatencyCheckInterval>
        <AdminChannel>
        <LocalInterface>testmachine</LocalInterface>
        <AdminPort>5002</AdminPort>
        </AdminChannel>
        <ResendChannel>
        <ResendPort>5103</ResendPort>
        </ResendChannel>
```

```
        <TimeToLive>1</TimeToLive>

        <DataStreamChannelList>
        <DataStreamChannel>
            <ChannelName>test2</ChannelName>
            <LocalInterface>127.0.0.1</LocalInterface>
            <IPAddress>224.0.2.0</IPAddress>
            <Port>5050</Port>
        </DataStreamChannel>
        <DataStreamChannel>
            <ChannelName>test1</ChannelName>
            <LocalInterface>127.0.0.1</LocalInterface>
            <IPAddress>224.0.2.0</IPAddress>
            <Port>5800</Port>
        </DataStreamChannel>
        </DataStreamChannelList>
</Publisher>
```

7. In `$RAPOUT_HOME/templates`, create an RDS template for each stream you want to publish to RAP.

```
<?xml version="1.0" encoding="UTF-8"?>

<Template xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="../template.xsd">
    <MessageDefnList>
        <MessageDefn>
            <MessageDesc>Split Event</MessageDesc>
            <MessageType>70</MessageType>
            <DestTableName>rapout2</DestTableName>
            <FieldDefnList>
                <FieldDefn>
                    <FieldName>integer</FieldName>
                    <IntegerField>
                        <IntegerDataType>sint32</IntegerDataType>
                    </IntegerField>
                    <DestColumnName>int16</DestColumnName>
                </FieldDefn>
        <FieldDefn>
            <FieldName>string</FieldName>
            <StringField/>
            <DestColumnName>string</DestColumnName>
        </FieldDefn>
                <FieldDefn>
                    <FieldName>int32</FieldName>
                    <IntegerField>
                        <IntegerDataType>sint32</IntegerDataType>
                    </IntegerField>
                    <DestColumnName>int32test</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>int64</FieldName>
                    <IntegerField>
                        <IntegerDataType>sint64</IntegerDataType>
                    </IntegerField>
```

```
                    <DestColumnName>int64test</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>double</FieldName>
                        <DecimalField>
                            <Precision>18</Precision>
                            <Scale>4</Scale>
                        </DecimalField>
                    <DestColumnName>doubletest</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>money</FieldName>
                    <DecimalField>
                        <Precision>18</Precision>
                        <Scale>4</Scale>
                    </DecimalField>
                    <DestColumnName>money_test</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>money(1)</FieldName>
                    <DecimalField>
                        <Precision>18</Precision>
                        <Scale>1</Scale>
                    </DecimalField>
                    <DestColumnName>money1</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>money(2)</FieldName>
                    <DecimalField>
                        <Precision>18</Precision>
                        <Scale>2</Scale>
                    </DecimalField>
                    <DestColumnName>money2</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>money(3)</FieldName>
                    <DecimalField>
                        <Precision>18</Precision>
                        <Scale>3</Scale>
                    </DecimalField>
                    <DestColumnName>money3</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>money(4)</FieldName>
                    <DecimalField>
                        <Precision>18</Precision>
                        <Scale>4</Scale>
                    </DecimalField>
                    <DestColumnName>money4</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>money(5)</FieldName>
                    <DecimalField>
                        <Precision>18</Precision>
                        <Scale>5</Scale>
                    </DecimalField>
```

```
                    <DestColumnName>money5</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>money(6)</FieldName>
                    <DecimalField>
                        <Precision>18</Precision>
                        <Scale>6</Scale>
                    </DecimalField>
                    <DestColumnName>money6</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>money(7)</FieldName>
                    <DecimalField>
                        <Precision>18</Precision>
                        <Scale>7</Scale>
                    </DecimalField>
                    <DestColumnName>money7</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>money(8)</FieldName>
                    <DecimalField>
                        <Precision>18</Precision>
                        <Scale>8</Scale>
                    </DecimalField>
                    <DestColumnName>money8</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>money(9)</FieldName>
                    <DecimalField>
                        <Precision>18</Precision>
                        <Scale>9</Scale>
                    </DecimalField>
                    <DestColumnName>money9</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>money(10)</FieldName>
                    <DecimalField>
                        <Precision>18</Precision>
                        <Scale>10</Scale>
                    </DecimalField>
                    <DestColumnName>money10</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>money(11)</FieldName>
                    <DecimalField>
                        <Precision>18</Precision>
                        <Scale>11</Scale>
                    </DecimalField>
                    <DestColumnName>money11</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>money(12)</FieldName>
                    <DecimalField>
                        <Precision>18</Precision>
                        <Scale>12</Scale>
                    </DecimalField>
```

```
                    <DestColumnName>money12</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>money(13)</FieldName>
                    <DecimalField>
                        <Precision>18</Precision>
                        <Scale>13</Scale>
                    </DecimalField>
                    <DestColumnName>money13</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>money(14)</FieldName>
                    <DecimalField>
                        <Precision>18</Precision>
                        <Scale>14</Scale>
                    </DecimalField>
                    <DestColumnName>money14</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>money(15)</FieldName>
                    <DecimalField>
                        <Precision>18</Precision>
                        <Scale>15</Scale>
                    </DecimalField>
                    <DestColumnName>money15</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>interval</FieldName>
                    <IntegerField>
                        <IntegerDataType>sint64</IntegerDataType>
                    </IntegerField>
                    <DestColumnName>interval</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>bigdatetime</FieldName>
                    <DateTime2Field/>
                    <DestColumnName>bigdatetime</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>date</FieldName>
                    <DateTimeField/>
                    <DestColumnName>date_test</DestColumnName>
                </FieldDefn>
                <FieldDefn>
                    <FieldName>timestamp</FieldName>
                    <DateTimeField/>
                  <DestColumnName>timestamp_test</DestColumnName>
                </FieldDefn>
    </FieldDefnList>
        </MessageDefn>
    </MessageDefnList>
</Template>
```

Ensure that the template file is copied to the RAP subscriber template directory

Enabling Kerberos Authentication for the RAP Output Adapter

Enable Kerberos authentication for the RAP Output Adapter by setting the necessary environment variables and specifying the **UseKerberos** element.

1. Set the following environment variables:
   a) Set the ESP_SERVICE_NAME environment variable to set the service principal name.
   b) Set the ESP_GSSAPI_LIB environment variable to point to the shared library provided by the Kerberos install. The library contains the GSSAPI function implementations.

      Note: If using a Kerberos library that depends on additional libraries, set the PATH environment variable for Windows or the LD_LIBRARY_PATH environment variable for Solaris and Linux.

   c) Set the KRB5CCNAME environment variable to point to the ticket cache.
   d) Set the KRB5_CONFIG environment variable to point to the configuration file used by the Kerberos library.
2. Specify the **UseKerberos** element and set the value to "true" in the `espfeedhandler.xml` file.

Operation

Start and stop the RAP adapter from the command line.

Starting the RAP Adapter

Once you have configured the adapter, start it using the **start.sh** script.

Prerequisites

- Start the RAP databases (there are message tables in the database), RAP subscribers, the Server, and that the project you want the adapter to connect to.
- Install `libodbc.so` if it is not present. A symbolic link with the file name `libodbc.so.1` should be made to `libodbc.so` version 1.0.0 and this file should be put in `$ESP_HOME/adapters/rap_out/lib.`
- To use the `start.sh` script, copy the platform specific `libpublisher.so` to the `$ESP_HOME/adapters/rap_out/lib` directory.

Task

1. Start the RAP databases (RAPCache and RAPStore) by selecting `start` in the RAP OpsConsole.
2. Start the RAP subscribers by selecting `start` in the RAP OpsConsole.

3. From a command prompt, execute the **start.sh** script.

   The start.sh script executes:

```
esp_rap_out_adapter -f $RAPOUT_HOME/config/espfeedhandler.xml -t
$RAPOUT_HOME/templates -p $RAPOUT_HOME/config
```

See also

- *Start Command* on page 544

Stopping the RAP Adapter

Once you have configured the adapter, stop it using the **esp_rap_out_adapter** command.

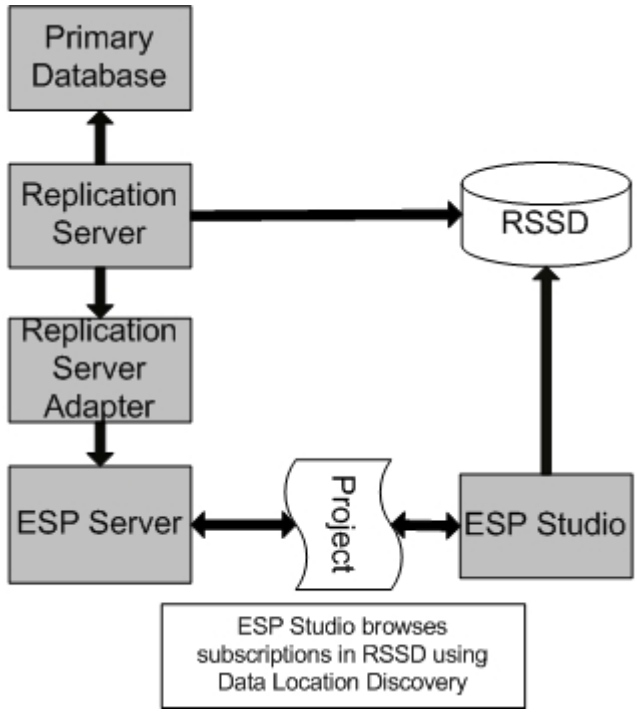
1. Shut down the adapter:
   - If you are running the adapter in the foreground, go to the window in which you started the adapter and press Ctrl-C.
   - If you are running the adapter in the background, enter `ps -eaf | grep esp_rap_out_adapter` to get the process ID of the adapter, then enter `kill -l5 processID`.
2. Shut down the RAP subscribers by selecting `stop` in the RAP OpsConsole.
3. Shut down the RAP databases by:
   - Selecting `stop` in the RAP OpsConsole.
   - If you are running the adapter in the background, enter `ps -eaf | grep dataserver` (for RAPCache) or `ps -eaf | grep IQSRV15` (for RAPStore) to get the process ID of the databases, then enter `kill -l5 processID`.

See also

- *Stop Command* on page 545

Replication Server Adapter

This external adapter receives data from SAP Replication Server and feeds that data into ESP. Like the RepAgent within ASE and the Replication Agents for other supported databases (such as MS SQL Server, Oracle, and UDB), the Replication Server Adapter does the necessary conversion to the datatypes of the target system, in this case ESP.



See also

- *Adapter Support for Schema Discovery* on page 1005

Configuring the Adapter on the Replication Server Workstation

Set up the Replication Server adapter, schema, and source location on the Replication Server workstation.

These configuration instructions are compatible with source data coming from the SAP Adaptive Server Enterprise database, as well as other databases.

1. Set up the replication system according to the SAP Replication Server documentation.
2. On a machine that can connect to the Replication Server, create a temporary directory and go to that directory.
3. Copy the following scripts from the `$ESP_HOME/adapters/repserver/config` directory on the machine where you installed Event Stream Processor to the temporary directory you just created.
4. Apply these scripts to the Replication Server.
   a) `isql -Usa -SSAMPLE_RS -P -i srsa_funcstrings.sql`
   b) `isql -Usa -SSAMPLE_RS -P -i srsa_errors.sql`

c) `isql -Usa -SSAMPLE_RS -P -i srsa_rs_errors.sql`

5. Using the **dsedit** utility, add an entry to the interfaces (`sql.ini`) file with the name of theEvent Stream Processor workstation and the port used for the Replication Server adapter connection. This entry specifies the **Replication Server adapter data server name** and **TDS Port** in the Replication Server Adapter configuration process. The port must match the adapter's **tdsListenerPort** configuration parameter. See *Chapter 6: Using dsedit* in the *SAP Adaptive Server Enterprise 15.7 Utility Guide* for more information on modifying the interface or `sql.ini` files. For example, if the adapter and the Event Streaming Processor are on a workstation named `my_workstation` and the connection is to be made on port 5100, use:

   ```
   [RSadapter]
   query=TCP,my_workstation,5100
   ```

6. Define the user name and password used for this connection within the Replication Server. This user name must NOT be the same as that of the administrator user (adminUser) of the Replication Server adapter.
   a) Connect to the Replication Server using **isql**.
   b) Define theuser name and password: they must mastch the adapter's **repConnUser** and **repConnPasswd** configuration parameters.

   ```
   create user rsuser
   set password rspassword
   go
   ```

7. Create the connection from the Replication Server to the adapter. This must match the adapter's **repSubscriptionServer** configuration parameter. Log in to the Replication Server to create the Replication Server adapter connection. For example,

   ```
   create connection to RSadapter.RSadapter
   set error class to srsa_error_class
   set function string class srsa_function_class
   set username rsuser
   set password rspassword
   set batch to 'off'
   with dsi_suspended
   go
   alter connection to RSadapter.RSadapter
   set replication server error class to srsa_rs_error_class
   go
   ```

   Turn off minimal columns,

   ```
   alter connection to RSadapter.RSadapter
   set replicate_minimal_columns to 'off'
   go
   ```

   Note: Do not use minimal columns in the **repdef**.

   To enable batching,

   ```
   alter connection to RSadapter.RSadapter
    set batch to 'on'
   go
   alter connection to RSadapter.RSadapter
   ```

```
 set dsi_cmd_separator to ';'
go
```

8. Create the replication definitions. A replication definition specifies the schema and the source location for a given table or stored procedure.
   There is a replication definition for a source table named "TEST" (`create table TEST (ID int, FNAME char(15)`), which is defined on a sourced database located on an Adaptive Server Enterprise server named ASEHOST.

```
create replication definition TESTrep
with primary at ASEHOST.sourcedb
with all tables named 'TEST'
(ID int, FNAME char(15))
primary key (ID)
go
```

9. Mark the Adaptive Server Enterprise source "TEST" table for replication. Log in to the Adaptive Server Enterprise server, locate the source table "TEST", and execute:

```
use <database name>
go
```

   followed by command:

```
sp_setreptable 'TEST', true
go
```

10. Define the subscriptions. Each subscription defines a target for the information coming through the Replication Server. In the following example, the target is the Replication Server adapter connection for the Replication Server adapter located on the Event Streaming Processor workstation.

```
create subscription TESTsub
for TESTrep
with replicate at RSadapter.RSadapter
without materialization
go
```

Note: You must follow the *Setting the JAVA_HOME Environment Variable* procedure before using the Replication Server Adapter with the ESP Studio. Doing it now is recommended.

You may now log off from the Replication Server.

Setting the JAVA_HOME Environment Variable

Set the JAVA_HOME environment variable to point to the Java directory.

Prerequisites

- Install Java Runtime Environment version 1.7.0_1 or higher. To see if you have a suitable version of Java installed, go to *http://www.java.com/en/download/installed.jsp*.
- To download and install Java, go to *http://jdl.sun.com/webapps/getjava/BrowserRedirect?locale=en"&"host=www.java.com:80*.

Task
Set the JAVA_HOME environment variable to the directory path where Java Runtime Environment 1.7.0_1 or higher is installed.

Next
Verify that the ESP_HOME environment variable is set correctly.

Configuring the Adapter on an Event Stream Processor Workstation
Set up a project using the Replication Server adapter on an Event Stream Processor workstation.

Prerequisites
Complete the Replication Server adapter configuration on the Replication Server workstation.

Task

1. Start the Studio:

Windows	Click **Start > All Programs > Sybase > Event Stream Processor > Studio**
UNIX	Enter `$ESP_HOME/ESP/studio/esp_studio`

2. Define a new project using the Studio Visual editor. In the Studio, select **File > New > Project**.
3. Configure the Replication Server Input Adapter.
   a) Go to the Input Adapters Palette and click on the Replication Server Input Adapter; then go to the Canvas and click again.
   b) Click the Edit Properties icon.
   c) From the Configure Adapter Properties window that opens, configure the adapter parameters used to connect to the Replication Server System Database (RSSD) to obtain metadata on tables and stored procedures:

Note: The RSSD contains the Replication Server system tables which provide information about the databases involved and the data transfer rules. It is hosted and managed by the Adaptive Server. Or, it could be an Embedded Replication Server System Database (ERSSD) hosted by an SQL Anywhere database. Provide property values in the same manner for both RSSDs and ERSSDs, except where indicated. (If you don't know whether you are working with an RSSD or an ERSSD, look in your Replication Server's configuration file to see whether **RSSD_embedded** is set to yes or no.)

Property Label	Description
ESP Server User ID	Property ID: **espUser** Type: `string` (Required)User name for connecting to the ESP Server.
ESP Server Password	Property ID: **espPassword** Type: `string` (Optional)ESP Server password for the ESP Server User ID. Do not use if using RSA authentication.
Replication Server Connection User	Propery ID: **repConnUser** Type: `string` (Required) Specify the same user name you specified when *Configuring the Adapter on the Replication Server Workstation*.
Replication Server Connection Password	Property ID: **repConnPasswd** Type: `string` (Required)Specify the same user password you specified when *Configuring the Adapter on the Replication Server Workstation*.
RSSD Host	Property ID: **rssdHost** Type: `string` (Optional for adapter operation; required only for schema discovery) Specify the name of the server on which the RSSD resides.
RSSD Port	Property ID: **rssdPort** Type: `uint` (Optional for adapter operation; required only for schema discovery) Specify which port on the RSSD Host to use when accessing the RSSD.

Property Label	Description
RSSD Database Name	Property ID: **rssdDatabaseName** Type: `string` (Optional for adapter operation; required only for schema discovery) Specify the name of the RSSD database created by the Replication Server to store replication information. **Note:** For an ERSSD, the RSSD Database Name is to be left blank.
RSSD User Name	Property ID: **rssdUser** Type: `string` (Optional for adapter operation; required only for schema discovery) Specify the user name used to connect to the RSSD server during schema discovery. This user must have permissions to run the RSSD Stored Procedures.
RSSD Password	Property ID: **rssdPasswd** Type: `password` (Optional for adapter operation; required only for schema discovery) Specify the password for the RSSD user. While this is optional for running, it is required for schema discovery unless your RSSD User Name account does not require a password.
RSadapter Data Server Name	Property ID: **repSubscriptionServer** Type: `string` (Required)Specify the shared data server and database name that defines the Replication Server connection pointing to the Replication Server adapter within Event Stream Processor. This server name is also used to define an entry in the Replication Server's interfaces `sql.ini` file. This value should be used for both the "data server" and "database name" portions of the Replication Server connection definition.

Property Label	Description
Project URI	Property ID: **projectUri** Type: `string` (Required)URI to connect to a project in a cluster environment. The required URI format is: esp://hostname:port/workspace_name/project_name
Authentication Mechanism Type	Property ID: **authType** Type: `string` (Required)Specify the authentication mechanism to use: valid values are `kerberos`, `rsa`, or `user_password`. User/Password authentication can employ an LDAP, native OS, or pre-configured user name and password. If you are uncertain what type of authentication you are using, refer to the cluster configuration file.
RSA Key Store	Property ID: **rsaKeyStore** Type: `string` (Optional)RSA Key Store file name and location.
RSA Key Store Password	Property ID: **rsaKeyStorePassword** Typer: `password` (Optional)RSA Key Store password.
Kerberos KDC	Property ID: **kerberosKdc** Type: `string` (Optional)Host name of Kerberos key distribution center. Required if using Kerberos authentication.
Kerberos Realm	Property ID: **kerberosRealm** Type: `string` (Optional)Kerberos realm setting. Required if using Kerberos authentication.

Property Label	Description
Kerberos Service	Property ID: **kerberosService** Type: `string` (Optional)Kerberos principal name that identifies an Event Stream Processor cluster. Required if using Kerberos authentication.
Kerberos Ticket Cache	Property ID: **kerberosTicketCache** Type: `string` (Optional)Location of Kerberos ticket cache file. Required if using Kerberos authentication.

4. Select the **Advanced** tab and configure the adapter parameters for runtime processing, internal communications, and establishing connections through SSL:

Property Label	Description
Stored Proc Stream Operation	Property ID: **storedProcStreamOp** Type: `choice` (Advanced)Specify the ESP stream operation to perform when replicating the Stored Procedure. Valid values are `insert` and `upsert`. The default is insert.
TDS Port	Property ID: **tdsListenerPort** Type: `uint` (Advanced)Specify the port used by the adapter. This is the port to which the Replication Server connection definition must connect. This port is defined within the interfaces (`sql.ini`) file on the Replication Server workstation. It defines connectivity between the Replication Server and the adapter.
Adapter Admin User	Property ID: **adminUser** Type: `string` (Advanced)Specify a user name for the adapter to use for communications between the first instance of the adapter (the "main listener") and subsequent instances of the adapter. This name can be anything except it must not match the user defined within the Replication Server connection definition.

Property Label	Description
Adapter Admin Password	Property ID: **adminPasswd** Type: `password` (Advanced)Specify a password associated with the user name the adapter uses for communications between the first instance of the adapter (the "main listener") and subsequent instances of the adapter.
Transactional Stream Operations	Property ID: **isTransactional** Type: `boolean` (Advanced)Specify whether or not multiple events are transmitted to Event Stream Processor as a single transaction.
Async Stream Operations	Property ID: **isAsync** Type: `boolean` (Advanced) Specify whether or not to use asynchronous stream operations.
Batched Stream Operations	Property ID: **isBatched** Type: `boolean` (Advanced)Specify whether or not the Replication Server adapter sends data to the Event Stream Processor in batches.
Batch Size	Property ID: **batchSize** Type: `uint` (Advanced) If the **Batched Stream Operations** parameter is set to true, specify the number of rows in the batch.
Publish When Batch is Full	Property ID: **publishWhenBatchFull** Type: `boolean` (Advanced) If set to true, the adapter writes data to the stream when the batch reaches **Batch size**. If set to false, the adapter waits for a commit to write data to the stream.

Property Label	Description
Error on Missing Stream Column	Property ID: **errorOnMissingStreamColumn** Type: `boolean` (Advanced)Set to true to send an error back to the Replication Server if a column in the **repdef** is not defined within the stream. Setting this parameter to false (to avoid frequent disconnections from the Replication Server) is recommended.
RSSD Table Name	Property ID: **tableName** Type: `tables` (Advanced)Specify the name of the table in the RSSD. Required if the RSSD table name is different from the corresponding stream name.
Field Mapping	Property ID: **permutation** Type: `permutation` (Advanced)Mapping between Event Stream Processor and external fields. Format is the ESP column name equals the database column name to which you are mapping. Multiple mappings are separated by a colon. For example, <esp_columnname>=<database_columnname>:<esp_columnname>=<database_columnname>. No default value.
Property Set	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Property Label	Description
TDS SSL Port	Property ID: **tdsSslListenerPort** Type: `uint` (Advanced)The SSL port the adapter will use to communicate securely with Replication Server. This port number should match the SSL port specified in the interfaces (`sql.ini`) file on the Replication Server workstation. The regular TDS port parameter is still required, as that port is used by the adapter for internal communication. Therefore, when enabling SSL communication, the adapter uses 2 ports.
Keystore	Property ID: **keystore** Type: `string` (Advanced)The path to the keystore that contains the certificate used in the SSL handshake with Replication Server. Specify a path relative to the working directory of the application (that is, the <base-directory> property under <ApplicationTypes> in the `<node_name>.xml` configuration file), or make the path absolute.
Key store Password	Property ID: **keyStorePasswd** Type: `string` (Advanced)The password for the keystore that contains the certificate used in the SSL handshake with Replication Server. Specific password requirements such as minimum number of characters depend on the tool you use to generate the keystore.

5. Select **OK** to save the project configuration.
6. Discover the source tables or stored procedures for the adapter. The schema discovery process establishes a connection to the RSSD and reveals defined subscriptions that target the Replication Server adapter.
   a) Click **Schema Discovery** in the Adapter shape.
   A list of tables or stored procedures that have Replication Server adapter subscriptions associated with them is returned. The `rs_lastcommit` table is also returned.
   b) Select one of the discovered tables and click **Next**.
   c) In the Create Element dialog, choose **Create new input window**.
   d) Click **Finish**.

e) Define primary keys. Each table and stored procedure stream must contain at least one primary key. Making this key match the primary key defined in the replication definition is recommended. Locate the appropriate primary key column and click on the toggle key to the left of it. The toggle key changes to an image of a key.

   Schema discovery adds a special column named 'ra_pkey' to the stream definition for a stored procedure. The 'ra_pkey' column must be set as the primary key and 'Autogen' set to true for stored procedures.

f) Select **File > Save** to save the changes.

7. Test data movement to the Event Stream Processor stream. Use the Studio for these steps, unless otherwise noted:

a) Open the project within the SAP Sybase Event Stream Processor Studio.

b) Go to the SAP Sybase ESP Run-Test perspective.

c) Run this project in a pre-started cluster. The cluster URI, workspace name, and project name must match the projectURI which is defined in adapter parameters. A successful start-up appears as:

   Stream TEST is ready for Replication Server connections.

d) Log in to the Replication Server using **isql** and resume the Replication Server adapter connection:

```
resume connection to RSadapter.RSadapter
go
```

e) Insert sample data into the source table.

f) Verify that the replicated data reaches the Replication Server adapter stream using the Stream View tab in Studio.

8. Add rs_lastcommit to the project.

a) Select the rs_lastcommit table returned as part of the schema discovery process and click **Next**.
   The system displays the Create Element dialog.

b) If you have not already created an input stream or window, choose **Create a new input window**, otherwise, choose **Create a new named schema**.

c) Click **Finish**.

d) Click on the "origin" column icon to set that column as the primary key.

Note: The rs_lastcommit table is non-persistent by default. It is held in memory and cleared when the stream is shut down. This results in a full replay of all remaining items within the Replication Server when the stream is restarted. Making rs_lastcommit persistent to minimize the replay of transactions following a stream restart is recommended.

---

9. Configure rs_lastcommit to use the persistent log store.

a) In the **Palette**, expand **Shared Components**.

b) Click the **Log Store** component from **Shared Components** then go to the canvas and click again.

c) Edit the store property in rs_lastcommit so that it selects the log store.

Configuring the Adapter for Communication through Secure Socket Layers

You can configure the Replication Server adapter to communicate with Replication Server over a secure connection using secure socket layers (SSL).

1. Provide a keystore with the key or certificate to be used during the SSL handshake. You can use a self-signed certificate or obtain one from a certification authority (CA).

   The JRE provides a keytool executable you can use to create a keystore containing a self-signed certificate. To use this tool, execute:

   ```
   keytool -genkey -keyalg RSA -alias <keystore_alias> -keystore
        <keystore_path>\keystore.jks -storepass <keystore password>
   -validity 360 -keysize
         2048
   ```

   The values and requirements for the command parameters depend on the tool you use to generate the keystore; there are no ESP-specific requirements for these values. In the current example, the keystore location can use an absolute or relative path. The validity period is user-defined and expressed in number of days.

   Ensure that the first and last name field of the key or certificate matches the name of the adapter connection definition in the `RepServer sql.ini` file. For example, if the adapter connection definition is **ESP_RSAdapter**, then the first and last name field for the key or certificate should be **ESP_RSAdapter**.

   All other values, except for the key password, are optional.

2. Export the certificate from the keystore in RFC format into a file named `trusted.txt`. If you are using the JRE keytool executable, execute:

   ```
   keytool -exportcert -alias <keystore alias> -keystore <path to
   keystore.jks> -storepass
         <keystore password> -file <path to trusted.txt> -rfc
   ```

3. Place the `trusted.txt` file inside the SQL directory of your Replication Server installation. Replication Server looks in this file for trusted certificates during the SSL handshake.

4. In the `RepServer sql.ini` interface file, modify the adapter connection entry by adding "`,ssl`" to the end. For example:

   ```
   [espadapter]
   master=TCP,<hostname>,<port>,ssl
    query=TCP,<hostname>,<port>,ssl
   ```

5. When configuring the adapter:

   a) Set the **keystore** parameter to point to the keystore file created in step 1. Specify a path relative to the working directory of the application (that is, the <base-directory>

property under <ApplicationTypes> in the `<node_name>.xml` configuration file), or make the path absolute.

b) Set the keystore password parameter to the password you set for the keystore you created in step 1.

c) Specify the port in the TDS SSL port parameter to match the port in the `RepServer sql.ini` file adapter connection entry.

Providing a greater-than-zero value to the TDS SSL port parameter enables SSL communication between the adapter and Replication Server. The regular TDS port parameter is still required, as that port is used by the adapter for internal communication. Therefore, when enabling SSL communication, the adapter uses 2 ports.

When you have configured SSL communication between the adapter and Replication Server, you can use SSL communication against ESP projects running in a cluster with SSL turned on by specifying the project URI adapter property with "esps://". This will achieve end-to-end SSL communication from Replication Server to the adapter to the ESP server.

Supported Datatypes

Map the datatypes provided by Replication Server to the datatypes used by Event Stream Processor.

Replication Server supports other databases (such as Oracle, Microsoft SQL Server, and IBM DB2 UDB) and the datatypes that they use. It provides data to ESP using the datatypes shown in the mapping table. These datatypes are also used by Adaptive Server Enterprise.

Unsupported datatypes may need to be mapped. For example, when using Oracle datatypes, there are two options for handling Oracle BIGDATETIME and TIMESTAMP values.

- Configure the Replication Agent for Oracle (RAO) to automatically convert Oracle BIGDATETIME and TIMESTAMP values into an SAP Adaptive Server Enterprise format:

  ```
  ra_config pdb_convert_bigdatetime, true
  ```

  In the Replication Server `repdef`, define the Oracle BIGDATETIME and TIMESTAMP columns using the corresponding SAP datatype (bigdatetime, timestamp)

  ```
  create replication definition oraBIGDATETIMErep
  with primary at rao.aeoracle
  with all tables named 'TESTBIGDATETIME'
  (ID INTEGER, ORABIGDATETIME bigdatetime, ORATIMESTAMP timestamp)
  primary key (ID)
  ```

- Configure the RAO to not automatically convert Oracle BIGDATETIME and TIMESTAMP values into an SAP ASE format:

  ```
  ra_config pdb_convert_bigdatetime, false
  ```

  In the RS `repdef`, define the Oracle BIGDATETIME and TIMESTAMP columns using the RAO defined data types (rs_oracle_bigdatetime, rs_oracle_timestamp9) and map them to the corresponding SAP datatype (bigdatetime, timestamp)

```
create replication definition oraBIGDATETIMErep
with primary rao.aeoracle
with all tables names 'TESTBIGDATETIME'
(ID INTEGER, ORABIGDATETIME rs_oracle_bigdatetime to bigdatetime,
ORATIMESTAMP rs_oracle_timestamp9 map to timestamp)
primary key (ID)
```

Replication Server also handles the different character sets used by supported databases. It provides data to the adapter in Unicode. ESP has Unicode (UTF-8) support: it is turned off by default and may be turned on for individual projects. See the *Project Configurations* topic in the *Studio Users Guide* for details.

ESP Datatype	Replication Server Datatypes
`integer`	`smallint, tinyint, int, bit`
`timestamp`	`datetime, time`
`date`	`date, smalldatetime`
`long`	`bigint, unsigned bigint, unsigned int, unsigned smallint`
`string`	`binary, char, unichar, nchar, nvarchar, varbinary, univarchar, varchar, timestamp`
`float`	`numeric, float, real`
`money`	`money, smallmoney`
`bigdatetime`	`bigdatetime` (ASE15.7 or later)
`boolean`	`bit`
`binary`	`varbinary`

Note: When creating replication definitions, please review the Supported Datatypes section and map any unsupported datatypes. The Replication Server Adapter does not handle the full range of `unsigned bigint` values; it throws an error for any value outside the range of a `bigint`.

Performance Tips

Modify the system configuration to improve performance.

- Ensure parallel DSI configuration parameters are being used.
- Configure the Replication Server to use batching (terminator must be a semicolon).
- Configure the Replication Server Adapter to use batching.

- Configure the Replication Server packets to be larger than the default. A value of 4096 is recommended.
- Take advantage of minimal columns.
- Set all log levels to INFO or lower in the `log4j.properties` file.
- Tune the source ASE database.
- Run on a 64-bit machine with the java JDK.

Logging

The Replication Server adapter uses the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing the logging configuration is part of the Replication Server adapter distribution.

You can modify the logging levels of the `log4j.properties` configuration file which is located in the `%ESP_HOME%\adapters\<adapter-name>\config` directory. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
```

```
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Tracing

You can enable or enhance tracing settings to assist in debugging issues. You can enable tracing in Replication Server itself, and you can increase the level of detail provided in both ESP project logs and JRE logs.

Increasing the Debug Logging Level

Increase project-level log details to assist with debugging and troubleshooting issues with your projects.

1. In the **CCR Project Configuration** editor, select the **Advanced** tab.
2. Click Project Deployment followed by **Project Options** to navigate to the Debug Level option.

3. Set the log level to a value between 0 and 7, depending on the amout of detail you require. 0 provides the least detail and 7 provides the most. Typically, a value of 4 provides the optimal level of detail debugging projectsIncrease the Debug Level to at least 4, warning conditions.

   Increasing this option to a value of 7 allows the greatest amount of detail to be logged as each of the following level increments log additional detail.

Name	Level	Description
LOG_EMERG	0	system is unusable
LOG_ALERT	1	action must be taken immediately
LOG_CRIT	2	critical conditions
LOG_ERR	3	error conditions
LOG_WARNING	4	warning conditions
LOG_NOTICE	5	normal but significant condition
LOG_INFO	6	informational
LOG_DEBUG	7	debug-level messages

Note: You will lose your changes to the logging level if you restart the project without also changing the logging level on the Server. After you have made the change, stop and remove the project from the Server, then redeploy the project to activate the new logging level.

Enabling Replication Server Tracing

Enable Replication Server Tracing from the command line using the ISQL client.

From within the ISQL client, execute the following command against the server.

```
TRACE 'ON', 'DSI', DSI_BUF_DUMP'</codeblock>
```

Replication Server Tracing is now enabled.

Increasing JDK Logging Levels

You can increase the level of logging detail to aid in debugging. Log levels for JDK are specified in the `log4j.properties` file under `$ESP_HOME/adapter/repserver/config`.

Note: When modifying these settings, remember that TRACE contains the most information. Also, changing one or more settings is allowed; it is not required that you change all three.

Set one or more of the following to either DEBUG or, for more detail, TRACE.

- log4.appender.R.Threshold

- log4j.appender.stdout.Threshold
- log4j.appender.R.File

Troubleshooting

Learn how to troubleshoot various issues including verifying component connectivity, resetting and restarting the system, and more.

Gathering Version Information

Identify complete version numbers for both your installation of Replication Server and the Replication Server adapter to help identify known issues and assist with troubleshooting.

1. Log into the Replication Server Adapter directly using ISQL and execute `select@@version`. The following is returned: `SAP Sybase Event Stream Processor Adapter for Replication Server 5.1.0.` This is the adapter's version information.
2. The Replication Server version is written in the Replication Server Log file. By default, this log is located in the install directory.

Verifying Connectivity Between Components

If you are experiencing communication issues between Replication Server and the adapter, verify that the adapter is running and that you have configured the correct connection information for your Replication Server data source.

Verifying that the Adapter is Running

Use ISQL to verify that the adapter is running.

1. From the Replication Server workstation, use ISQL to issue the following command: `isql -SESPadapter -Uanyuser -Panypassword`.

   Note: The -S option is the argument for ISQL to specify the name of the server to connect to. In this case, you are connecting to the Replication Server Adapter. If, for example, the adapter and the Event Stream Processor are on a workstation named *my_workstation* and the connection is to be made on port 5100, you should have an entry defined similar to the following: `[ESPadapter] query=TCP,localhost,5100.`
2. Check the ESP Server Console for Replication Server connection messages. If the adapter displays an error message upon startup, follow the troubleshooting steps provided by the message.

Verify that Replication Server is Communicating With the Adapter

If you are experiencing communication issues between Replication Server and the adapter, verify that you are using the correct port, host, and username.

To check for a connection failure:

1. Resume the connection and verify the connection status. For example, resume the connection to `ESPadapter.ESPadapter` and verify that the connection status is either `Awaiting Command` or `Awaiting Message`.
2. If the connection fails to resume, check the Replication Server log for additional information. For example `2011/01/25 14:31:41. Still trying to connect to server 'ESPadapter' as user 'partner'` indicates a connection failure. In such cases:
   a) Verify the port number defined within the `interfaces/sql.ini`file for the ESP Adapter for Replication Server connection (ESPadapter) matches the TDS PORT value within the ESP DataLocatioin advanced properties.
   b) Verify the hostname used within the `interfaces/sql.ini` file for the ESP Adapter for Replication Server connection (ESPadapter) is available to the Replication Server workstation and is the hostname for the ESP server.
   c) Verify the username defined within the ESP Adapter for Replication Server connection (`ESPadapter.ESPadapter`) does not match the Adapter Admin User value defined within the ESP DataLocation advanced properties.

Problems Resetting and Restarting the System

Learn about issues that may occur when resetting and restarting the system.

Problems with transactions replaying upon a restart of an ESP project or an "Unknown streamName" error within the ESP Server Console.	• Indicates a replay of transactions for a stream that is no longer available. • A non-persistent rs_lastcommit always replays transactions. • A persistent rs_lastcommit minimizes transaction replay. Persist the rs_lastcommit using a log store and verify that the Transactional Stream Operations is set to True.
Problems with transactions replaying upon a restart of an ESP project or an error within the ESP Server Console.	• Purge the Replication Server Adapter queue (queue number) prior to a project restart by logging into the Replication Server and executing `sysadmin hibernate_on - go - sysadmin sqm_purge_queue, 107,0 - go - sysadmin hibernate_off - go.` **Note:** The queue number (107) should correspond with the queue number of your ESP Replication Server Adapter connection on the Replication Server. Execute an "admin who"to determine the queue number.

- Modifying the ESP Adapter for Replication Server parameters requires a restart of projects, thus all non-persistent streams are cleared.
- Re-materialize source data, if desired, by dropping and recreating subscriptions.
- Suspend the Replication Server connection to the Replication Server Adapter prior to stopping the project. This forces an update to the persistent rs_lastcommit, which improves startup time for Replication Server connecting to the ESP Adapter for Replication Server.

Common Errors and Resolutions

Learn how to troubleshoot common errors.

Problem	Solution
Replication Server Adapter's Data Discovery fails due to failing to access a row.	A stored procedure replication definition is defined using old Replication Server syntax `create function replication definition`. This format is not supported within the Replication Server Adapter. Drop the existing subscription and replication definition and redefine the replication definition using the new Replication Server syntax `create applied function replication definition`.
Replication Server connection fails to connect to the Replication Server Adapter.	• Ensure the port defined within `interfaces/sql.ini` matches the TDS Port property within the ESPDataLocation advanced properties. • Ensure that the servername parameter within the `interfacessql.ini` file, the Replication Server Adapter Replication Server connection name (servername and database values), and the Replication Server Adapter Data Server Name parameter value for the Replication Server Adapter configuration all match. • Ensure that the Replication Server connection username does not match the Adapter Admin User property value within the Replication Server Adapter configuration.

A project or stream fails at startup with the following error: "Platform call returned error see console for details."	A primary key has not been defined within the stream. Select the appropriate column within the stream to define as a key.
No data is flowing to the Replication Server adapter stream following a full configuration of the Replication Server replication definition, subscription, and Replication Server Adapter streams.	Verify the table in the source database is marked for replication using "sp_setreptable". For example, on an ASE source table, use the following to verify the table has been marked successfully. If the table is marked for replication, the following message returns: `sp_setreptable 'TESTTABLE' - the following message returns: The replication status for 'TESTTABLE' is currently true, owner_off. (return status = 0).`
Only one row is inserted into a stored procedure stream.	Verify the **Stored Proc Stream Operation** parameter for the Replication Server Adapter has been defined as *insert*and the **autogen** parameter for the ra_pkey column has been set to *true* for the stored procedure stream.
Long delays occur before a long-running transaction is applied to the Replication Server Adapter stream.	When using ESP batching, long-running transactions may see delays while the Replication Server Adapter waits for the Replication Server to send the final commit. This is expected behaviour.
Only the first SQL statement in a batch is processed.	The **ds_command_separator** must be defined as ";". Otherwise, all statements beyond the first SQL are intepreted as invalid and thrown out.
Dematerialization shuts down the Replication Server connection to the Replication Server Adapter.	Dematerialization is not supported. To clear the dematerialization queue: identify the dematerialization queue using an `admin who` on the Replication Server and execute a `drop_queue` similar to `sysadmin drop_queue, 107, -2147483468 go.`
No errors are reported upon a failed SQL command within a batch process.	This is expected behaviour as a result of the ESP API. For performance reasons, the ESP API and RS Adapter do not wait for a confirmation on each transaction.

Some unsigned bigint values fail to insert into the Replication Server Adapter stream, reporting a parsing error for the value.	The maximum UNSIGNED BIGINT value supported by ESP is smaller than the maximum value supported by the data source. Modify the datatype for the UNSIGNED BIGINT column within the Replication Server Adapter stream and change it from an INT64 to a STRING.
Replication Server connection to the Replication Server Adapter fails and errors 84083972/13045 are reported by the Replication Server log.	Check the `interfaces/sql.ini` file on the Replication Server workstation and verify the entry for the Replication Server Adapter connection is defined properly and points to the ESP workstation.Verify that port corresponds with the TDS Port defined within the ESP Replication Server Adapter project.
Data is resent although a persistent rs_lastcommit has been defined.	Verify that the persistent rs_lastcommit within the project has been defined with only one key and that the column is selected as the key in the "origin" columnn.

Unsupported Replication Server Features

Learn about features that are not supported in this version.

These features are not supported:

- Guaranteed transactional consistency
- Request function replication
- The old replication server style function replication
- Update of primary keys
- If there are two different RepDefs that define the same `owner.objectname` the columns from both will show up in one discovered table.
- Dematerialization of subscriptions
  - Drop subscriptions without using purge. If used with purge, the Replication Server will initiate dematerialization, which is not supported by the Replication Server Adapter.
- Bulk materialization
- Bulk insert
- Replicating DDL must be turned off
- dsi_bulk_copy must be set to off
- dsi_compile_enable must be set to off
- dsi_commit_control must be set to on
- dsi_quoted_identifier must be set to off
- Dynamic SQL set to off at server and rep def level
- No spaces in table, column, or stored procedure names

- No reserved SQL keywords allowed
- No delimited identifiers
- Truncate table
- All DML

These features are supported, but Data Discovery does not detect them:

- Function strings
- Database replication
- Publications

Reuters Marketfeed Adapter

The SAP Reuters Marketfeed adapter is a software interface between SAP Sybase Event Stream Processor and the Reuters Market Data System (RMDS). It uses the Reuters Marketfeed message format.

You can configure the adapter as an input or output adapter. The input adapter subscribes to one or more Reuters Instrument Codes (RICs) on the RMDS to provide input to Event Stream Processor. The output adapter publishes output from Event Stream Processor to the RMDS. This enables Event Stream Processor to use the speed and reliability of Reuters' infrastructure to deliver data.

The Reuters Marketfeed Input adapter supports schema discovery. Run two adapter instances if you require both input and output capabilities.

The adapter runs on Solaris and Linux operating systems but you can use it with Event Stream Processor software running on Solaris, Linux, or Windows.

See also

- *Adapter Support for Schema Discovery* on page 1005

Requirements

The Reuters Marketfeed input and output adapters have several requirements.

An input adapter requires:

- A RMDS market data connection that uses the Marketfeed protocol
- A working subscription for data on one or more financial instruments

An output adapter requires:

- A working connection with support for sending data to RMDS using the Marketfeed protocol

General Configuration

Enable user access for each user account that runs the Reuters Marketfeed adapter, and configure an input connection from Reuters and an output connection to Reuters.

Enabling User Access

Enable user access for each user account that uses the Reuters Marketfeed adapter.

1. Ensure the user account has permission to execute the installed software.
2. Add an environment variable, $ESP_REUTERS_HOME, set to the root of the adapter hierarchy, to the user's runtime environment.
3. (Optional) Add the environment variable to your shell profile.
4. Event Stream Processor supports RSA, LDAP, and Kerberos authentication. If your installation uses one of these authentication methods, ensure the user account is set up to work with that method of authentication.

Files for External Libraries

On SuSE11 and RH6.0 operating systems, the Reuters Marketfeed adapter has dependencies on certain external libraries.

Note: Athough the ESP intallation is 64bit, this adapter requires the 32bit OpenSSL version 0.9.8 (or higher) and 32bit glibc (version 2.5 or higher).

1. Execute the following command to ensure that the 32bit version of OpenSSL is installed in your system:

   ```
   % rpm -qa | grep openssl | grep 32bit
     libopenssl0_9_8-32bit-0.9.8j-0.38.1
   ```

2. After installation, verify that `libssl.so.6` and `libcrypto.so.6` are present. If they are not, execute the following from the root account to create symbolic links to them:

   ```
   # cd /usr/lib
   # ln -s /usr/lib/libcrypto.so.0.9.8 libcrypto.so.6
   # ln -s /usr/lib/libssl.so.0.9.8 libssl.so.6
   ```

3. Test that the 32bit shared libraries can be found and loaded and that the adapter is ready to run:

   ```
   % ldd esp_rmds | grep "not found" % esp_rmds -v
   ```

   The adapter is ready when there are no "not found" messages and you can get the version string.
4. If the ldd command reports that any libraries are not found, set your LD_LIBRARY_PATH to the directory where these libraries reside.

Configuring an Input Connection from Reuters

Modify the sample configuration file for your site's RMDS connection.

Prerequisites

- Create (or choose) a directory in which to store your site-specific configuration files.
- Create an environment variable (MY_CONFIG) and set it to the full path name of that directory.

Task

During the installation process, a sample configuration file (`rfasub.cfg`) was placed in the `$ESP_REUTERS_HOME/config directory`. This file, shown below, follows the Reuters format for configuration files.

```
# Change this if necessary.
# the port number of the P2PS (default 8101)
\Connections\Connection_SSLED\PortNumber           = 8101

# Change this if necessary.
# the user name to connect with (should be the DACS name if DACS is
enabled)
\Connections\Connection_SSLED\UserName             = "triarch"

# Change this if necessary.
# a list of P2PS host names
\Connections\Connection_SSLED\ServerList           = "localhost"

# Refer to RFA documentation for more advanced changes to the
remaining entries
\Connections\Connection_SSLED\connectionType       = "SSLED"

\Adapters\SASS3_Adapter\requestQueueReadThreshold  = 1
\Adapters\SASS3_Adapter\mainLoopTimerInterval      = 200

\Adapters\SSLED_Adapter\masterFidFile              = "config/
appendix_a"
\Adapters\SSLED_Adapter\enumTypeFile               = "config/
enumtype.def"
\Adapters\SSLED_Adapter\downloadDataDict           = false

# Change the fileLoggerFilename appropriately for your setup
\Logger\AppLogger\windowsLoggerEnabled             = false
\Logger\AppLogger\fileLoggerEnabled                = true
\Logger\AppLogger\fileLoggerFilename              = "rfasub.{p}.log"

\Control\Entitlements\dacs_SbeEnabled              = false
\Control\Entitlements\dacs_CbeEnabled              = false

\Logger\ComponentLoggers\Connections\messageFile   = "config/
messages/RFA7_Connections.mc"
\Logger\ComponentLoggers\Adapter\messageFile       = "config/
```

```
messages/RFA7_Adapter.mc"
\Logger\ComponentLoggers\SessionCore\messageFile   = "config/
messages/RFA7_SessionLayer.mc"
\Logger\ComponentLoggers\SSLED_Adapter\messageFile = "config/
messages/RFA7_SSLED_Adapter.mc"


\Sessions\Session1\connectionList                   =
"Connection_SSLED"
```

1. Obtain this information from your system administrator:
   - Name of the server that receives Marketfeed data from RMDS
   - Port number on the machine to which your system connects
   - User name defined for your connection to Reuters
   - Name of each Reuters service to which you are subscribed
2. Make a copy of the sample configuration file in your `$MY_CONFIG directory`:
   ```
   cp $ESP_REUTERS_HOME/config/rfasub.cfg $MY_CONFIG
   ```
3. Use a text editor to open the configuration.
4. In the `\Connections\Connection_SSLED\PortNumber` line, replace the default port number (8101) with the number used by your Reuters connection, if different.
5. In the `\Connections\Connection_SSLED\UserName` line, replace triarch with the user name for your Reuters subscription. Keep the surrounding quotation marks. In the `\Connections\Connection_SSLED\ServerList` line, replace `localhost` with the name of the server that receives Marketfeed data from RMDS. Keep the surrounding quotation marks.

   If your system has more than one server receiving data from RMDS, include all of their names in a comma-separated list, in priority order.
6. (Optional) In the `\Logger\AppLogger\fileLoggerFilename` line, you can change the name of the log file.

   The default file name specified here, `rfasub.{p}.log`, includes the string {p} which the Reuters library replaces with the UNIX process ID when it creates the log file.
7. Save the modified file.

   The other parameters in the configuration file also affect the functioning of the Reuters Marketfeed adapter, and you may want to modify them as well.

Configuring an Output Connection to Reuters

Modify the sample configuration file for your site's RMDS connection.

Prerequisites

- Create (or choose) a directory in which to store your site-specific configuration files.
- Create an environment variable (MY_CONFIG) and set it to the full path name of that directory.

Task

During the installation process, a sample configuration file (`rfapub.cfg`) was placed in the `$ESP_REUTERS_HOME/config` directory. This file, shown below, follows the Reuters format for configuration files.

```
# Change this if necessary.
# This needs to match port number for the route as defined in the
Source Distributor.
\Connections\Connection_SSLED_MP\ipcServerName      = "8105"

# Refer to RFA documentation for more advanced changes to the
remaining entries.
\Connections\Connection_SSLED_MP\connectionType     = "SSLED_MP"
\Connections\Connection_SSLED_MP\entitlementData    = false
\Sessions\Session1\connectionList                   =
"Connection_SSLED_MP"

# Change the fileLoggerFilename appropriately for your setup
\Logger\AppLogger\windowsLoggerEnabled              = false
\Logger\AppLogger\fileLoggerEnabled                 = true
\Logger\AppLogger\fileLoggerFilename                = "./rfapub.
{p}.log"

\Control\Entitlements\dacs_SbeEnabled               = false
\Control\Entitlements\dacs_CbeEnabled               = false

\Logger\ComponentLoggers\Connections\messageFile      = "./config/
messages/RFA7_Connections.mc"
\Logger\ComponentLoggers\Adapter\messageFile          = "./config/
messages/RFA7_Adapter.mc"
\Logger\ComponentLoggers\SessionCore\messageFile      = "./config/
messages/RFA7_SessionLayer.mc"
\Logger\ComponentLoggers\SSLED_Adapter\messageFile    = "./config/
messages/RFA7_SSLED_Adapter.mc"
\Logger\ComponentLoggers\SSLED_MP_Adapter\messageFile = "./config/
messages/RFA7_SSLED_MP_Adapter.mc"
```

1. Obtain this information from your system administrator:
   - Port number at which the src_dist or RMDS infrastructure server listens for updates from the Reuters Marketfeed adapter
   - Name of the server that receives updates from Event Stream Processor
2. Make a copy of the sample configuration file in your `$MY_CONFIG` directory.

   ```
   cp $ESP_REUTERS_HOME/config/rfapub.cfg $MY_CONFIG
   ```
3. Use a text editor to open the configuration.
4. In the `\Connections\Connection_SSLED_MP\ipcServerName` line, replace the default port number (8105) with the port number at which your src_dist listens for updates from the Reuters Marketfeed adapter, if different.
5. (Optional) In the `\Logger\AppLogger\fileLoggerFilename` line, change the name of the log file. The default file name specified here, `./rfapub.{p}.log`,

includes the string {p} which the Reuters library replaces with the UNIX process ID when it creates the log file.

6. Save the modified file.

Enabling Kerberos Authentication for the Reuters Marketfeed Adapter

Enable Kerberos authentication for the Reuters Marketfeed input and output adapters by setting the necessary environment variables and specifying the -G option.

Prerequisites

Ensure you have a 32-bit version of Kerberos installed in the machine as the Reuters Marketfeed adapter does not work with 64-bit versions.

Task

1. Set the following environment variables:
   a) Set the ESP_SERVICE_NAME environment variable to set the service principal name.
   b) Set the ESP_GSSAPI_LIB environment variable to point to the shared library provided by the Kerberos install. The library contains the GSSAPI function implementations.

      Note: If using a Kerberos library that depends on additional libraries, set the PATH environment variable for Windows or the LD_LIBRARY_PATH environment variable for Solaris and Linux.

   c) Set the KRB5CCNAME environment variable to point to the ticket cache.
   d) Set the KRB5_CONFIG environment variable to point to the configuration file used by the Kerberos library.
2. Specify the -G option.

Input Adapter Configuration

Configure an input adapter to push data from the Reuters Market Data Service (RMDS) to Event Stream Processor.

Before configuring an input adapter, decide what data you need and how you want to set up your system.

You need to know the following about the Event Stream Processor instance from which you receive data.

- Possible security options in a cluster environment, and the workspace and project name.
- What type of authentication mechanism (Kerberos, RSA, LDAP, or Native OS (user name/password) does it use?

Data Decisions

Decide how the incoming Reuters data fits into the project.

Also decide whether you require Level 1 or Level 2 data. For Level 1 data, use the Reuters Marketfeed adapter, and for Level 2 data, use the Reuters OMM adapter instead.

Decision	Description
Venues	Decide which venues are of interest (for example, NYSE, NASDAQ, Toronto, and so on).
RICs and FIDs	Determine what market data you need. Specifically, which Reuters Instrument Codes (RICs) you want the adapter to provide to Event Stream Processor, and which Reuters Field IDs (FIDs) for these instruments you want to use.
Streams	The Reuters adapter can furnish data to one or more streams on Event Stream Processor. To use the Reuters Market Data provided by the adapter, decide which existing data streams to map to the adapter's data feed, or define one or more new streams.

Administrative Decisions

You have several administrative decisions to make in regards to the project.

Decision	Description
Session Name	An arbitrary string used to link the project and the adapter map file. Use it consistently.
Directories for logging and stream output	The adapter writes its own log messages and can generate a separate set of Reuters log messages. In the configuration, specify if and where these log files should be written.
SAP user account	Specify a valid Event Stream Processor user account for the adapter to use.

Input Adapter Map File

The map file configures the interface between the Reuters Marketfeed adapter and Event Stream Processor. It specifies which source streams receive data from RMDS via the adapter, and it maps specific RMDS Field Identifiers (FIDs) to specific columns in that source stream.

The input adapter map file must accomplish three major tasks:

- Match incoming data elements to columns in one or more streams defined in the Event Stream Processor configuration file.
- Match the RIC provided with each update from the adapter with a row in the Event Stream Processor configuration file.

- Ensure that each update from the adapter can be converted into a record that provides a unique key for each stream being populated, as defined by the stream's column definitions.

Data Structures

Data structures have three important structural aspects: data columns, datatypes, and key values.

- Each data stream includes one or more data columns.
- Each column has a datatype.
- In most streams, each row has a unique key value. The source stream definition designates one or more columns as "key" columns.

Incoming RMDS Data

When the adapter subscribes to RMDS for a certain RIC, RMDS first sends an initial image containing all available market data for that RIC.

After that, RMDS sends an update when any values for a subscribed RIC change. Each update consists of the identifying RIC, with the Field Identifier (FID), and the new value for each change. Each FID defined for RMDS has a datatype.

Market Data Field Mapping

Map each column in the target Event Stream Processor stream to a Reuters FID or a pseudo-field.

Find the appropriate FID for each column in the stream. The datatype of the Event Stream Processor column must be compatible with the datatype of the Reuters FID that feeds it.

Here are possible matches between FID datatypes and Event Stream Processor datatypes:

Event Stream Processor Datatype	Reuters Datatype
`integer` or `long`	`integer`
`string`	`alphanumeric`
`integer`	`enumerated`
`timestamp` or `date`	`time, date`
`money` or `float`	`price`
`long` or `integer`	`time_seconds`
Not supported	`binary`

Reuters Instrument Code Mapping

The identifier of each incoming RMDS update is the Reuters Instrument Code (RIC).

Map the RIC to a column of datatype string in the stream. If the stream you want to map to does not have a suitable column, either add a column to the stream, or map to a different stream.

Matching the Stream's Key

The adapter map file must configure the adapter so that every update sent to the Event Stream Processor stream includes a field or combination of fields conforming to the unique key defined for that stream. To make this more flexible, the adapter configuration mechanism supports "pseudofields".

The market data updates that the adapter receives from RMDS are mapped to columns in the Event Stream Processor stream using the dataField or dateTimeField element in the map file. RMDS also provides non-market data information, for example, each update includes a RIC. Additionally, you can configure the adapter to add a sequence number to each update.

To make these data items available to the mapping process, the map file mechanism supports the following elements, called pseudofields:

Field	Description
`itemName`	Type: `string` (Required) The RIC.
`serviceName`	Type: `string` (Optional) Name of the service from which RMDS received the market data from this RIC.
`itemState`	Type: `integer` (Optional) The item state.
`sequenceNumber`	Type: `long` (Optional) A unique number, assigned sequentially by the adapter to each incoming event (whether or not it causes an update).
`FIDListField`	Type: `string` (Optional) Shows the FID name and value for each updated value.
`updateNumber`	Type: `long` (Optional) A unique number, assigned sequentially by the adapter to each incoming update.

Getting Stream Information from the Project

Gather the necessary information about the Reuters stream.

The first step in configuring the input adapter is to determine the source streams on Event Stream Processor which will receive the RMDS market data. If the Event Stream Processor project does not already include one or more streams for this purpose, define a new stream (or streams) for use with the Reuters adapter.

After you have chosen (or defined) the streams that will receive data from the Reuters Marketfeed adapter, collect information about that stream from your project file. The Event Stream Processor project file contains one or more stream definitions. Each stream definition specifies a data stream that is instantiated when Event Stream Processor is started. The stream definition comprises:

- A unique ID for the stream
- A database store and output file for the stream data
- A list of the columns used as the unique key value for each row in the data stream

Once you have decided which streams will carry the RMDS data provided by the Reuters adapter, get information from the stream definition in the project file. There is no standard for project file names. Two Event Stream Processor installations may have completely different stream definitions, but the definition of any stream includes the same basic set of components.

These instructions refer to the example project to show what components of the stream configuration you must identify to configure the Reuters Marketfeed adapter.

1. Open the project to which the adapter provides data. The example shown here is the `$ESP_REUTERS_HOME/examples/example.ccl` file supplied with the Reuters Marketfeed adapter distribution.
2. Find the name of the source stream. The opening SourceStream tag specifies the name of the stream as the value of the id attribute. The first source stream in this example is named “stream1.”

   The stream used for subscription by the Reuters Marketfeed adapter must always be a source stream.
3. Determine the key fields. Examine each of the column entries between the opening and closing SourceStream tags to see if the key attribute is set to true. In this example, “stream1”, has one key field, “symbol.”
4. Carefully note the number and order of the column entries in the source stream definition.

   In the input adapter map file, list the same set of data in the same order.

Creating the Input Adapter Map File

Create an adapter map file to configure the interface between the Reuters Marketfeed input adapter and Event Stream Processor.

This procedure maps updates from RMDS to the source stream defined in the `example.ccl` file. This file is in the `$ESP_REUTERS_HOME/examples` directory along with an example map file.

1. Open a new map file using an editor.
2. Enter the following as the first line of the file to specify that the adapter map file conforms to XML version 1.0.:

   ```
   xml version="1.0" encoding="UTF-8"
   ```

3. Specify that this is an adapter map file and that includes a separate file:

   ```
   <!DOCTYPE adapter [
   <!ENTITY rmdsFields SYSTEM "rmds.sm.mf.xml">
   ]>
   ```

4. Add the opening and closing adapter tags. In the opening adapter tag, specify the name of the adapter. For example:

   ```
   <adapter name="mySubscribeAdapter1">
   </adapter>
   ```

5. After the opening adapter tag, add the publication element. Specify the name to be used in log messages for this adapter and any other attributes required to prescribe how the adapter should deliver data to Event Stream Processor.

   For example,

   ```
   <publication name="RMDS Adapter - low latency" retryInterval="5" /
   >
   ```

   This example also includes a retryInterval attribute with a value that tells the adapter to wait five seconds before retrying if it fails to connect to Event Stream Processor.

6. After the publication element, add the opening and closing streamMaps tags to contain the streamMap elements that do the actual mapping between RMDS FIDs and columns of an Event Stream Processor stream. Each streamMap maps to one and only one Event Stream Processor stream.

   ```
   <streamMaps>
   </streamMaps>
   ```

   Since the streamMaps section can contain more than one streamMap, one instance of the adapter can provide RMDS data to more than one Event Stream Processor stream.

7. Enter a streamMap element for each Event Stream Processor stream to which you wish to send RMDS data. For each streamMap,
   a) Enter the opening streamMap tag specifying the name of the Event Stream Processor stream to which the RMDS data is sent as the value of the name attribute.
   b) Enter the closing streamMap tag.

c) Between the streamMap tags, add one mapping element for each column defined in the target stream's definition. You can do this in the map file itself or in a separate file that is included in the map file as an entity.

```
<streamMap name="stream1" flags="NO_SHINE">
&rmdsFields;
</streamMap>
```

8. After the streamMaps section, add the rfa element, including:
   - A config attribute that specifies the absolute path and file name of the Reuters configuration file
   - A sessionName attribute that specifies a session name corresponding to the one used in the Reuters configuration file

   ```
   <rfa config="$ESP_REUTERS_HOME/config/rfasub.cfg"
   sessionName="Session1" />
   ```

   The rfa element may also include attributes to modify the adapter's treatment of blanks (by default it converts them to zeros). You can specify the value for the blank attribute or specify values for each datatype directly using the blankInt32, blankInt64, blankMoney, blankString, blankDate, and blankTimestamp attributes. Specify a value that does not conflict with any of the values you expect in your data. If you are using both input and output adapters, specify the same value for each attribute to both adapters.
9. Between the rfa element and the closing adapter tag, add the opening and closing itemLists tags. When entering the opening itemLists tag:
   - Specify the Reuters service from which the adapter is receiving RMDS data as the value of the service attribute.
   - Specify the name of the Event Stream Processor stream that is receiving the RMDS data as the value of the stream attribute.

   ```
   <itemLists service="IDN_RDF" stream="stream1">
   </itemLists>
   ```

   The itemLists tags will contain one or more pairs of opening and closing itemList tags.
10. Between the itemLists tags, add opening and closing itemList tags for each separate list of RICs to which the adapter subscribes.
11. Between the itemList tags, add an item element for each RIC to add to the list. When entering the item element:
    a) Specify an RIC to which the adapter subscribes as the value of the name attribute.
    b) (Optional) Specify the name of the queue you wish to use as the value of the rfaQueue attribute. Specifying an rfaQueue spawns a separate thread to do the processing.
    c) (Optional) Specify the name of the service to use.

    For example:

    ```
    <itemList>
    <item name="AAPL.O" rfaQueue="queue1" />
    <item name="CSCO.O" />
    </itemList>
    ```

Running the Input Adapter

Run the Reuters Marketfeed input adapter once you have configured it.

Prerequisites

Configure an adapter.

Task

1. Ensure that **esp_server** is running and that the project has been loaded and started.
2. Start the adapter using the **esp_rmds** command.
   a) If the Event Stream Processor is running with Native OS (user name/password) authentication, start the adapter with this command:

      ```
      esp_rmds -a in -f mapfile -p cluster_host:cluster_port/
      workspace/project \
      -c username:password
      ```

      using the appropriate mapfile, cluster_host, cluster_port, workspace, and project names for the project to which the adapter will connect.
   b) If the Event Stream Processor is running with some other form of authentication, refer to *Command Usage* to obtain the additional arguments necessary for the command to start the adapter.

   The exact usage of the command depends on how you started your Event Stream Processor. You must invoke the adapter with compatible options. The command string shown above does not invoke encryption: you can specify it.
3. The adapter starts the subscription by first connecting to Event Stream Processor and then connecting to RMDS. Both connections must be operational for any data to flow.

   If you plan to direct the adapter's log output to stderr, as shown here, you may want to redirect stdout and stderr to a log file (for example, append `>& myrmdslog &` to the command line shown above).

Testing the Adapter

If the adapter is not working as expected, you can perform a quick sanity check by executing the **esp_rmds** command and verifying whether the adapter is sending Reuters market data to Event Stream Processor.

- Execute **esp_rmds**:

  ```
  esp_rmds -v
  ```

  This command returns the version information. Ensure that the Event Stream Processor to which you are connecting is compatible with your version of the adapter.
- There are three quick ways to verify that the Reuters Marketfeed adapter is sending Reuters Market Data to Event Stream Processor:

- Use the SAP Sybase Event Stream Processor Studio or the **esp_subscribe** command to check the output of the stream configured to receive Reuters data.
- Use the **tail** command on the redirected adapter log file (specified in the adapter map file) or the Reuters subscriber log (specified in the configuration file `rfasub.cfg`) for activity.
- Run the **esp_rmds** command with the -d7 option to produce verbose output.

Multiple RICs

When configuring an input adapter, you will usually want to specify multiple RICs.

There are several ways to do this:

- Specify each individual RIC by entering the name directly into the map file or use an XML ENTITY include file.
- Specify a chain RIC from Reuters.
- Create a dynamic watch list, which employs Event Stream Processor to specify the list of RICs.
- Use a combination of the options above.

Individual RICs

Enter an item element declaration for each RIC you want in the itemList section of the map file.

Here is an example of this:

```
<itemLists service="SSL_PUB" stream="stream1">
<itemList>
<item name="CSCO.O"/>
<item name="K.N"/>
<item name="KBN.N"/>
<item name="KBR.N"/>
<item name="ACAM.ARC"/>
<item name="IBM.ARC"/>
</itemList>
</itemLists>
```

It can become difficult to create and maintain your list of RICs this way if it is very large or changes frequently, for example, if you are attempting to track all of the stocks traded on the NYSE. All RICs for the same stream must use the same FID set. Since FIDs often vary by venue, use a different itemList and streamMap for each venue.

Chain RIC

When you specify the name of a chain RIC, Reuters translates it to a list of individual RICs. Chain RICs usually contain all of the RICs from a single market or for a single index instrument, such as the S&P 500 or the Russell 2000.

For example, to specify the chain RICs for the Dow Jones Index and the SIAC entities, add a chain- map section:

```
<streamMap name="chainMap" chain="1" >
<itemName /> <dataField name="REF_COUNT" />
```

```
<dataField name="NEXT_LR" /> <dataField name="PREF_LINK" />
<dataField name="LINK_1" /> <dataField name="LINK_2" />
<dataField name="LINK_3" /> <dataField name="LINK_4" />
<dataField name="LINK_5" /> <dataField name="LINK_6" />
<dataField name="LINK_7" /> <dataField name="LINK_8" />
<dataField name="LINK_9" /> <dataField name="LINK_10" />
<dataField name="LINK_11" /> <dataField name="LINK_12" />
<dataField name="LINK_13" /> <dataField name="LINK_14" />
<dataField name="LONGNEXTLR" /> <dataField name="LONGPREVLR" />
<dataField name="LONGLINK1" /> <dataField name="LONGLINK2" />
<dataField name="LONGLINK3" /> <dataField name="LONGLINK4" />
<dataField name="LONGLINK5" /> <dataField name="LONGLINK6" />
<dataField name="LONGLINK7" /> <dataField name="LONGLINK8" />
<dataField name="LONGLINK9" /> <dataField name="LONGLINK10" />
<dataField name="LONGLINK11" /> <dataField name="LONGLINK12" />
<dataField name="LONGLINK13" /> <dataField name="LONGLINK14" />
</streamMap>
```

and enter their names in the itemList section.

```
<itemList stream="stream1" service="IDN_RDF" >
<item name="0#.DJI" /> <!-- The Dow Jones Index -->
<item name="0#SIAC" /> <!-- The entities of SIAC -->
</itemList>
```

For more details about chains, look at the example in `chain.example.map.xml` in the `$ESP_REUTERS_HOME/examples` directory. For more information about Reuters chain RICs, see the *Reuters Venue Guide* for your chosen venue, which is available from Reuters.

Creating a Dynamic Watch List

Creating a dynamic watch list is a bit more complex than creating individual or chain RICs, but is also more flexible. Chain RICs are limited to those defined by Reuters, but with this method you can specify a customized list of RICs.

Prerequisites

Define a source stream (named MyInfoStream) to receive the data, and manually edit the list of RICs to include.

Task

Creating a dynamic watch list is also dynamic: when inserts or deletes occur on the stream configured using these steps, RMDS subscriptions to the appropriate RICs are started or stopped.

1. Define a stream on Event Stream Processor (for example, MyListStream), which publishes to the adapter the list of RICs to which you want to subscribe. This stream requires these columns:

Column	Description
symbol	Specifies an RIC symbol ticker (for example, CSCO.O) to which the adapter should subscribe.
service	Specifies the RMDS service on which to subscribe to obtain data for that RIC.
stream	Specifies the name of the stream (for example, MyInfoStream) on which the adapter publishes data for this RIC.

The stream can also include an optional fourth column, rfaQueue.

2. Define a second stream on Event Stream Processor (for example, MyInfoStream) that receives the data requested by the first stream.
3. Edit the map file to include the subscription.

```
<subscriptions>
<subscription name="subscription1" flags="BASE" >
<stream name="MyListStream" >
<name column="3" /> <!-- symbol -->
<field column="1" name="service"/>
<field column="2" name="stream"/>
</stream>
</subscription>
</subscriptions>
```

4. Specify the set of RICs you want and send them to the first stream you created (for example, MyListStream) to subscribe to them.
   a) Create a file with the same six columns that the stream expects in comma-separated values (CSV) format. The columns are: stream from which you are receiving data, opcode, service, symbol, and stream to which you are sending data.

      For example, open a new file (`RIClist.csv`) using an editor and put in these lines.

      ```
      MyListStream,p,,IDN_RDF,MyInfoStream,CSCO.O
      MyListStream,p,,IDN_RDF,MyInfoStream,K.N
      MyListStream,p,,IDN_RDF,MyInfoStream,KBN.N
      MyListStream,p,,IDN_RDF,MyInfoStream,KBN.R
      MyListStream,p,,IDN_RDF,MyInfoStream,ACAM.ARC
      MyListStream,p,,IDN_RDF,MyInfoStream,IBM.ARC
      ```

   b) Send the data from the file to Event Stream Processor using the **esp_convert** and **esp_upload** commands. The following example assumes that you have installed all SAP command line tools in the default directories and added those directories to your PATH variable. If you have not, prepend the appropriate path to each command shown in this example.

      For example, to send the file created in the previous step to Event Stream Processor running a project named p1 in workspace ws1 on port 19011 of your local server, enter:

```
cat RIClist.csv | esp_convert -c user:password -d "," -p
localhost:19011/ws1/p1 | esp_upload -c user:password -p
localhost:19011/ws1/p1
```

c) Start the adapter:

```
esp_rmds -f mapfile -d7 -c user:password -p localhost:19011/
ws1/p1 >& logfile &
```

If the adapter and Event Stream Processor are on different machines, enter the name of the remote host in place of localhost after the -p in the previous command.

Output Adapter Configuration

Configure an output adapter to push data from Event Stream Processor to RMDS, using RMDS as a message infrastructure.

Before configuring an output adapter, decide which data to provide and how you want to set up your system.

You need to know the following about the Event Stream Processor instance from which you receive data.

- Possible security options in a cluster environment, and the workspace and project name.
- What type of authentication mechanism (Kerberos, RSA, LDAP, or Native OS (user name/password) does it use?

Data Decisions

Identify which columns from which streams in Event Stream Processor to publish data from.

The Reuters Marketfeed adapter can rearrange the columns from a stream in any order. Its output can also include constants, and the published output can include values from more than one stream.

Consider these items when planning the output of the Reuters Marketfeed Output adapter:

- For each stream for which to publish data, you must specify a unique key in the output adapter map file. Since this adapter sends data to RMDS, the unique identifier should be an RIC.
- Each data column you want to publish from any stream must map to a unique FID.
- Data from one column can be repeated in the published output, giving you a way to publish a DateTime value as separate Date and Time values.
- If the stream you are working with receives data about the same FID from more than one service, you can configure the adapter to differentiate these data items by service and transmit each service's data separately.
- The first time the Reuters Marketfeed adapter publishes to RMDS, it publishes values for all the columns for which it is configured. After that initial image, the adapter only publishes updates for individual columns as these updates occur.

Administrative Decisions

You have several administrative decisions to make in regards to the project.

Decision	Description
Session Name	An arbitrary string used to link the project and the adapter map file. Use it consistently.
Directories for logging and stream output	The adapter writes its own log messages and can generate a separate set of Reuters log messages. In the configuration, specify if and where these log files should be written.
SAP user account	Specify a valid Event Stream Processor user account for the adapter to use.

Reuters Information

You need several pieces of information from Reuters to enable the Reuters Marketfeed adapter to publish to the RMDS.

- The name of the Reuters service on which the adapter transmits data
- Up-to-date lists of valid Reuters Instrument Codes (RICs) and Field Identifiers (FID) used by RMDS
- The Product Permission Code assigned by Reuters

The adapter does not work with the Reuters Data Access Control System (DACS), so the Product Permission Code is needed to allow access to the information you are transmitting on the RMDS.

A list of FIDs, `$ESP_REUTERS_HOME/config/appendix_a`, has been supplied as part of the Reuters adapter distribution. You can obtain the latest list and other information from your Reuters technical contact.

The datatype of the Event Stream Processor column must be compatible with the Reuters FID datatype that feeds it. This table shows possible matches between Event Stream Processor and FID datatypes:

Event Stream Processor Datatype	Reuters Datatype
`integer`, `long`	`integer` or `price`
`money`, `float`	`price`
`string`	`alphanumeric`
`date`, `timestamp`	`date` or `time`

Getting Stream Information from the Project

Gather the necessary information from the project.

The first step in configuring the output adapter is determining which data elements from which streams on the Event Stream Processor are to be published. After you have chosen (or defined) a project containing the items for publication over RMDS via the Reuters adapter, collect information from the streams to obtain the data to send to RMDS.

Each stream definition specifies a data stream that is instantiated when Event Stream Processor is started. The stream definition:

- Specifies a unique ID for the stream
- Identifies the columns used as the unique key value for each row in the data stream

Once you have decided which streams will provide the information to be sent to RMDS by the Reuters adapter, get information from the stream definition in the project file. There is no standard for project file names. Two Event Stream Processor installations may have completely different stream definitions, but the definition of any stream includes the same basic set of components.

1. Open the project to which the adapter provides data. The Reuters Marketfeed adapter distribution includes an example project, `$ESP_REUTERS_HOME/examples/example.ccl`.
2. From the definition of each stream defined in the project:
   a) Obtain the name of the stream from the id attribute in the opening tag of that stream.
   b) Verify that the key attribute is set to true for the column containing the RIC and note the column. In this example, both “stream1” and “orderbookStream” have the RIC in the column named “symbol,” which is identified as a key field.
   c) Decide what data, if any, you want the adapter to send to RMDS.
3. Carefully note which streams contain data to send to RMDS, and where in the stream definition it is located.

   In the output adapter map file, reference each of the columns you want to publish.

Creating the Output Adapter Map File

Create an adapter map file to configure the interface between the output adapter and Event Stream Processor.

The examples shown below map updates from RMDS to the source stream defined in the `example.project.xml` file.

1. Open a new map file using an editor.
2. Enter the following as the first line of the file to specify that the adapter map file conforms to XML version 1.0.:

```
xml version="1.0" encoding="UTF-8"
```

3. Add the opening and closing adapter tags by entering this as the first line:.

```
<adapter>
</adapter>
```

4. Define the configuration of the adapter's interface to RMDS by adding the rfa tag, with these attributes:

Attribute	Description
config	Specify the full path name of the Reuters configuration file.
fidFile	Specify the full path name of the Reuters-supplied file that lists all of the valid FIDs.
enumFile	Specify the full path name of the Reuters-supplied file that lists each enumerated type along with the range of values it can take.
serviceName	Specify the service name provided by Reuters for the adapter to send data to RMDS.
sessionName	Specify the sessionName value found in the Reuters configuration file, `rfasub.cfg`.

   For example, using the files that were shipped with the adapter distribution:

```
<rfa config="$ESP_REUTERS_HOME/config/rfapub.cfg"
fidFile="$ESP_REUTERS_HOME/config/appendix_a"
enumFile="$ESP_REUTERS_HOME/config/enumtype.def"
serviceName="IDN_RDF" sessionName="Session1" />
```

5. Add the subscriptions begin and end tags between the rfa element and the closing adapter tag.

```
<subscriptions>
</subscriptions>
```

   The adapter subscribes to Event Stream Processor to get the data to publish to the RMDS.

6. Between the opening and closing subscriptions tags, add opening and closing subscription tags to define a subscription. Include these attributes in the opening subscription tag:

Attribute	Description
name	Specify a unique name for this subscription.
flags	Set this parameter to “BASE” to obtain a complete set of initial values. This may be undesirable in situations such as recovery if there are a lot of unchanging values because getting those values adds latency to the other values. In these cases, set this parameter to “NO_BASE”.

```
<subscription name="subscription1" flags="BASE" >
</subscription>
```

Each subscription defined in the output adapter map file must reference at least one Event Stream Processor stream.

7. Add the stream definition to the subscription.
   a) Immediately before the closing subscription tag, insert the opening and closing stream tags. In the opening stream tag, include the name attribute set to the name of the stream.
   b) To use a "constant" rather than a column to specify your Reuters permission code, insert the constant tag immediately before the closing stream tag, including these attributes:

Attribute	Description
`name`	Specify the Reuters FID "PROD_PERM."
`value`	Specify the permission code issued by Reuters that certifies your permission to publish to RMDS.

   c) Immediately following the opening stream tag, insert the name tag, with the attribute column set to the column before the column with the symbol or RIC in the project. For example, if the symbol or RIC is in the first column in the project, set the value of column to 0.
   d) Immediately following the opening name tag, insert the stale tag, with the attribute column set to one less than the position of the value in the project.
   e) Between the stale and the constant tags, add a field tag for each data column in the stream that you want to send to RMDS. Include these attributes:

Attribute	Description
`column`	Set this parameter to either the name of the column or the numeric position (one less than the position of the value in the project).
`name`	Specify the Reuters FID for this data.

   For fields of datatype `float`, you may also include the precision attribute, set to the number of digits you want after the decimal place in the value sent to RMDS. For example:

```
<stream name="stream1" >
<name column="0"/>
<stale column="3" />
<field column="4" name="BID" precision="5" />
<field column="5" name="ASK" precision="0" />
<field column="6" name="TRDPRC_1"/>
<field column="7" name="ACVOL_1"/>
<constant name="PROD_PERM" value="1"/>
</stream>
```

Running the Output Adapter

Run the adapter once you have configured it.

Prerequisites

Configure an adapter.

Task

1. Ensure that **esp_server** is running and that the project has been loaded and started.
2. Start the adapter using the **esp_rmds** command.
   a) If the Event Stream Processor is running with Native OS (user name/password) authentication or encryption, start the adapter:

      ```
      esp_rmds -a out -f mapfile -p cluster_host:cluster_port/
      workspace/project \
      -c username:password
      ```

      using the appropriate mapfile, cluster_host, cluster_port, workspace, and project names for the project to which the adapter will connect.
   b) If the Event Stream Processor is running with encryption or some other form of authentication, refer to *Command Usage* to obtain the additional arguments necessary for the command to start the adapter.

   The exact usage of the command depends on how you started your Event Stream Processor. You must invoke the adapter with compatible options. The command string shown does not invoke encryption: you can specify it.
3. The adapter starts the subscription by first connecting to Event Stream Processor and then connecting to RMDS. Both connections must be operational for any data to flow.

   If you plan to direct the adapter's log output to stderr, as shown here, you may want to redirect stderr to a log file (for example, append `>& myrmdslog &` to the command line shown above).

Testing the Adapter

If the adapter is not working as expected, you can perform a quick sanity check by executing the **esp_rmds** command and verifying whether the adapter is sending Reuters market data to Event Stream Processor.

- Execute **esp_rmds**:

  ```
  esp_rmds -v
  ```
- This command returns the adapter release number and the revision number of the source tree separated by an underscore character. Ensure that your version of the adapter is compatible with your version of Event Stream Processor.
- There are several ways to verify whether the Reuters Marketfeed adapter is publishing to RMDS:

- Use the **tail** command on the adapter log file to which console output was redirected or any of the Reuters publisher log files (specified in `rfapub.cfg`) to look for activity.
- Use the **esp_subscribe** command to look at the outbound stream and verify that values are changing.
- Use RMDS tools to subscribe to RICs provided by the output adapter.
- Use an input adapter to subscribe to the output adapter.

Creating a Subordinate Map File

Create a subordinate map file to hold part of the map file configuration.

It can be advantageous to put part of your input or output adapter map file in a separate file. For example, you might want to keep a subscription configuration in a map file, but break out the list of RICs you want the adapter to subscribe to.

1. Go to the directory that contains the map file.
2. Create a new file with the extension `.xml`.

   You need not add a declaration of the XML version.
3. Insert the selected content from the map file into the new file.

   The content you add depends on which part of the map file you have decided to store separately.
4. (Optional) Add a comment to the new file.
5. Save the file.

Modifying the Main Map File

Modify the main map file to reference the subordinate file.

1. Make sure the first line of the main map file is:
   ```
   <?xml version="1.0"?>
   ```
2. Between the XML version declaration and the opening adapter tag, add these lines:
   ```
   <!DOCTYPE adapter SYSTEM "adapter.dtd" [
   ]>
   ```
3. For each subordinate map file:
   a) Between the two lines just added, add:
      ```
      <!ENTITY SUBREF SYSTEM "SUBFILE">
      ```
      where SUBREF is a string to reference the subordinate file and SUBFILE is the path and file name of the subordinate file itself. Enclose the path and file name in quotation marks.
   b) Remove the content that you put in the subordinate map file.
   c) Insert a string like the following to include the content from the subordinate map file:
      ```
      &SUBREF;
      ```

where SUBREF is the string you specified to reference the subordinate file.

Example

Configure the input adapter in the map file (`subInclude.map.xml`) to reference two subordinate files (`RIClist1.sm.mf.xml`, and `RIClist2.sm.mf.xml`).

The map file `subInclude.map.xml` configures the input adapter to reference two subordinate files, each containing a list of RICs for the adapter to subscribe to.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE adapter [
<!ENTITY RIClist1 SYSTEM "RIClist1.sm.mf.xml">
<!ENTITY RIClist2 SYSTEM "RIClist2.sm.mf.xml">
<!ENTITY rmdsFields SYSTEM "rmds.sm.mf.xml">
]>
<adapter>
<publication name="RMDS Adapter" retryInterval="5"
sendAsTransactions="0" flushInterval="1000"
intraSubscribeDelay="100"/>
<streamMaps>
<streamMap name="stream1">
&rmdsFields;
</streamMap>
</streamMaps>
<rfa config="$ESP_REUTERS_HOME/config/rmdsmf.cfg"
sessionName="Inbound" />
<itemLists>
&RIClist1;
&RIClist2;
</itemLists>
</adapter>
```

The first file, `RIClist1.sm.mf.xml`, contains:

```
<!-- This fragment is meant to be included in an itemLists section.--
>
<!-- These are FX RICs -->
<itemList service="IDN_RDF" stream="stream1">
<item name="GRMN.O"/>
<item name="INTC.O"/>
<item name="KLAC.O"/>
<item name="XLNX.O"/>
<item name="YHOO.O"/>
</itemList>
```

The second file, `RIClist2.sm.mf.xml`, contains:

```
<!-- This fragment is meant to be included in an itemLists section.--
>
<!-- These are FX RICs -->
<itemList service="IDN_RDF" stream="stream1">
<item name="AUD="/>
<item name="CAD="/>
<item name="DKKTN="/>
<item name="GBPSW="/>
```

```
<item name="GBPTN="/>
<item name="JPYSN="/>
<item name="JPYSW="/>
<item name="JPYTN="/>
<item name="HKD="/>
<item name="SGDSW="/>
<item name="ZAR="/>
<item name="ZARSN="/>
</itemList>
```

Performance Tuning

There are several attributes you can use to fine-tune performance in an input adapter.

Attribute	Description
`flushInterval`	Specify an interval of time in microseconds (for example, 5000 microseconds = 5 milliseconds) to wait while accumulating data. At the end of this interval, any accumulated events are sent to Event Stream Processor. Send events less often to allow more events to be placed into a message, resulting in a communications overhead savings. Use a nonzero flushInterval to make even accumulation time-based.
`maxRecordsPerBlock`	Specify the maximum number of accumulated events that the adapter should send to Event Stream Processor at a time. When the number of accumulated events is larger than this value, the envelope or transaction is broken into fragments that are less than or equal to the specified value. For example, if accumulated event counts of more than 1024 (which would immediately fill the Event Stream Processor Gateway's inbound queue) are expected, set maxRecordsPerBlock to a value like 500 to prevent the inbound queue from filling.
`pendingLimit`	Specify a threshold for the number of events that must accumulate before they are sent to Event Stream Processor. Set this parameter to zero to publish each event immediately when it happens (providing the lowest latency), at the expense of high network overhead (a TCP/IP packet for each update). If you set this parameter to a larger value, the adapter waits until number of events have accumulated, packs them efficiently in TCP/IP packets, and sends them to Event Stream Processor. This saves communication work but increases latency on both the adapter and Event Stream Processor.

Attribute	Description
`sendAsTransactions`	This parameter controls whether events are sent as an envelope or a transaction. You can specify this parameter on a per-stream basis. Set this parameter to true for Event Stream Processor to treat a group of events as a single transaction. Transactions typically cause application-level workload savings, since Event Stream Processor collapses multiple events to the same value (as determined by identical key columns) in a transaction to a single event. If a transaction contains a delete, additional savings are achieved since updates prior to the delete can be discarded. If you set this parameter to false and you are not in low-latency mode (pendingLimit and flushInterval both set to zero), then use the maxRecordsPerBlock to control the size of the envelope. You still gain the communications overhead savings mentioned above, but not the transactional savings. This is the preferred configuration for applications that require every event to be sent separately, such as a market data compliance application. As a general rule, for quote-based applications, where only the most recent update matters, use transactions to be most efficient. For trades, however, every event must be processed seperately to compute a total volume, use envelopes instead.

When you use both flushInterval and pendingLimit, no event waits longer than the time indicated in the flushInterval before being sent, and as long as the number of events specified in pendingLimit arrive, they are sent immediately. The adapter waits flushInterval and, if any events have accumulated, it sends them. If the number of pendingLimit events, or more, accumulate while the adapter is sending the earlier events, the new events are sent immediately (without waiting for the flushInterval). If fewer than the number of pendingLimit events accumulate while the adapter is sending events, it waits for the flushInterval to elapse.

You can also use the rfaQueue attribute at the itemLists, itemList, or item element level. When specified, the rfaQueue attribute causes the element to be subscribed from Reuters on a named rfaQueue. Each rfaQueue is processed by its own thread within the Reuters adapter. Spreading requests across multiple threads can reduce latency and improve overall adapter throughput at the cost of greater CPU usage.

Since all events (images and updates) for the same RIC come from Reuters on the same queue, the integrity of the order of arrival is maintained for any individual RIC. If you do not specify an rfaQueue for any of the elements, a single default queue (named "default") is used for all RICs.

Command Usage

The Reuters Marketfeed adapter converts data from the Reuters Market Data System (RMDS) to the Event Stream Processor and vice versa.

Synopsis

```
esp_rmds -f mapFile -p host:port/workspace/project [ OPTION ...]
```

Description

esp_rmds can operate as either an input or an output adapter. An input adapter passes data from RMDS in to the Event Stream Processor. An output adapter passes data from the Event Stream Processor out to RMDS. A single adapter instance cannot operate both ways. To have an input adapter and an output adapter, you must run two separate adapter instances.

The metadata describing the connection has several parts, including a map file, a configuration file, and possibly a configuration stream resident on a running instance of the Event Stream Processor.

Only limited Level 2 data is available via RMDS Marketfeed. For full order book depth, use the Reuters OMM adapter (**esp_rmdsomm**).

The Marketfeed adapter process runs as a daemon, getting its configuration from a map file. It handles SIGHUP; so you can enter `kill -s SIGHUP pid` on Linux or `kill -s HUP pid` on Solaris (where pid is the process ID of the **esp_rmds** daemon, which you can obtain using the **ps** command) to gracefully shut down the adapter. Using the KILL signal rather than the HUP signal may prevent a complete clean up of system resources.

There are three directories containing additional information underneath the directory where the adapter is installed: `doc`, `examples`, and `config`. The doc directory contains Reuters README files that describe various configuration options. The `examples` directory contains several example map files that demonstrate many features. The `config` directory contains example RMDS configuration files. Minimally, you must modify the RMDS `config` file with your site's specific information. Typically, you must also modify the map file to match the Event Stream Processor.

Required Arguments

- **-f mapFile –** specify the map file containing the metadata required to map the market data to/from RMDS.
- **-p host:port/workspace/project –** specify the URI to connect to the server (cluster manager). For example, `-p localhost:19011/default/prj1` specifies a project called prj1 in the default workspace of an ESP cluster server using port 19011 on the machine at which you entered the command.

Options

- **-a in|out –** specify whether the RMDS Marketfeed adapter instance is passing data in to the Event Stream Processor or receiving data passed out from it. Valid values are `in` and `out`. Since the default value is `in`, this option is typically omitted when subscribing to market data.

  For backward compatibility, "subscribe" (in) and "publish" (out) are still allowed, but these options have been deprecated.
- **-c user:password –** if you are using an authentication method that requires credentials (such as Kerberos, PAM, or RSA), this option passes those authentication credentials to Event Stream Processor. If Event Stream Processor successfully authenticates with these credentials, the connection is maintained, otherwise Event Stream Processor immediately closes the connection.
- **-d debugLevel –** set the debug level. The valid range is 0 - 7, with 0 being minimal and 7 being verbose. By default, the debug level is 4.
- **-e –** negotiate encrypted OpenSSL sockets for all communication with the Event Stream Processor, which must be started in encrypted mode when using this option.
- **-F configFile –** specify the RMDS configuration file, overriding the configuration file specified in the map file.
- **-g gatewayHost –** explicitly the Event Stream Processor gateway host.
- **-G –** use Kerberos authentication. This option is required when the Event Stream Processor is started with the **-V gssapi** option.
- **-h –** print a short help message describing the syntax of this command.
- **-k privateRSAKeyFile –** perform authentication using the RSA private key file mechanism instead of password authentication. The privateRSAKeyFile must specify the absolute path file name of the private RSA key file. With this option enabled, the user name must be specified with the -c option, but the password is not required. In addition, Event Stream Processor must be started with the -k option.
- **-l 0|1|2|3 –** specify the location to which log messages are sent. Use 0 for no log messages, 1 to send to stderr only (the default), 2 to send to syslog only, and 3 to send to both stderr and syslog.
- **-r subscribeRetryInterval –** specify how many seconds to wait (default is 300) between attempts to resubscribe to a RIC. (If a subscription to a RIC is marked CLOSED or CLOSEDRECOVER, you must resubscribe to that RIC for data to flow.) To disable resubscription attempts, specify 0 as the value. Periodically resubscribing can compensate for a temporary condition where the source is not ready for subscribers. Each unsuccessful resubscribe attempt generates a failure event which may result in a status update marking the item stale.
- **-s streamName –** specify the stream to be used when running in discovery mode. This option is used by the connector start mechanism and specifies the single stream for which mapped columns have been discovered.

- **-v –** print the version of the RMDS Marketfeed Adapter and exit.
- **-w retrySeconds –** specify the number of seconds to wait between retries when connecting to the Event Stream Processor. The default is 5. Specify 0 to try only once.
- **-x optName –** specify various extra settings; use `-x help` to see a list of possible values.
- **-z publishCount –** specify the number of values to pass to the Event Stream Processor before terminating. By default this is 0, which means never terminate.
- **-Z subscribeCount –** specify the number of values to pass to RMDS before terminating. By default this is 0, which means never terminate.

Examples

To start an input adapter, using the map file `subexample.map.xml`, running a project named project1 in a workspace named ws1 on port 19011 of the localhost machine, enter:

```
esp_rmds -c user:pw -p localhost:19011/ws1/project1 -a in -f
subexample.map.xml
```

Environment Variables

The Reuters Marketfeed adapters use environment variables to specify behavior.

Environment Variable	Used By	Description
ESP_ACCUMULATOR_DELAY	Input	(Expert) Delay connection to the Event Stream Processor (seconds).
ESP_DISABLE_REPORT_ENCOD-ING_NULL	Output	Stop warning about blank to null conversions (bool) [false].
ESP_FLUSH_INTERVAL	Input	Override the publication flushInterval (micro-seconds).
ESP_INTRASUBSCRIBE_DELAY	Input	Override the map attribute (milliseconds).
ESP_LOG_CONFIG_EVENTS	Both	Set log level (1–7) for config event processing [-1].
ESP_MARKETFEED_DUMP	Output	Set the log level (0–7) at which to dump raw Reuters messages to the log.
ESP_MAX_RE-CORDS_PER_BLOCK	Input	Override the publication maxRecordsPerBlock (count).
ESP_PENDING_LIMIT	Input	Override the publication pendingLimit.
ESP_RETRY_INTERVAL	Both	Override the publication retryInterval.
ESP_REUTERS_HOME	Both	Specify the installation directory.

Environment Variable	Used By	Description
ESP_RMDS_DISPATCH	Both	(Expert) Dispatch RFA every N milliseconds [10,000].
ESP_RMDS_EVENT_TRACE	Both	(Expert) Enables RFA event tracing every N event (int).
ESP_RMDS_PUBLISH_BUFSIZE	Output	Override the buffer size.
ESP_RMDS_PUBLISH_DEBUG_LEVEL	Output	Set to 7 to see values.
ESP_RMDS_PUBLISH_DEBUG_SYMBOLS	Output	Contains a space-delimited list of symbols that are used when default behavior is overridden. If this environment variable is not set, all symbols are used.
ESP_RMDS_SUBSCRIBE_DEBUG_LEVEL	Input	Set to 7 to see values.
ESP_RMDS_SUBSCRIBE_DEBUG_SYMBOLS	Input	Contains a space-delimited list of symbols for above. If this environment variable is not set, all symbols are used.
ESP_RMDS_SUBSCRIBE_SYMBOL_FORMAT	Input	Specify symbol list format: 0 for multiline; or 1 for single line.
ESP_SEND_AS_TRANSACTIONS	Input	Override the map attribute.
ESP_SHOW_FIELD_INFO	Input	Show FID, column, spColumn, and stream name [false].
ESP_SHOW_SP_EVENT_DATA	Output	Set log level (1–7) for events from the Event Stream Processor [-1].

Input Adapter Map File XML Syntax

The syntax of the map file for a Reuters Marketfeed Input adapter.

```
adapter                    (required, limit one)
  |----publication                   (required, limit one)
  |----streamMaps                    (required, limit one)
  |      '----streamMap              (required)
  |             |----itemName        (required, limit one)
  |             |----serviceName     (optional)
  |             |----sequenceNumber  (optional)
  |             |----itemStale       (optional)
  |             |----dataField       (required)
  |             |----updateNumber    (required)
```

```
  |             |----dateTimeField   (optional)
  |             |----FIDListField    (optional)
  |             '----nullField       (optional)
  |----recordTypeMap                 (optional)
  |       '----recordType            (optional)
  |----rfa                           (required, limit one)
  '----itemLists                     (required, limit one)
         '----itemList               (required)
                '----item            (optional)
```

adapter

The **adapter** element is the root element of the map file.

Summary

```
adapter                    (required, limit one)
  |----publication                   (required, limit one)
  |----streamMaps                    (required, limit one)
  |       '----streamMap             (required)
  |             |----itemName        (required, limit one)
  |             |----serviceName     (optional)
  |             |----sequenceNumber  (optional)
  |             |----itemStale       (optional)
  |             |----dataField       (required)
  |             |----updateNumber    (required)
  |             |----dateTimeField   (optional)
  |             |----FIDListField    (optional)
  |             '----nullField       (optional)
  |----recordTypeMap                 (optional)
  |       '----recordType            (optional)
  |----rfa                           (required, limit one)
  '----itemLists                     (required, limit one)
         '----itemList               (required)
                '----item            (optional)
```

Nest all the configuration sections between the **adapter** start and end tags.

Parent

None

Children

The following child elements are defined for the adapter element. All of these elements must be in the order specified.

Name	Requirement
publication	Exactly one required
streamMaps	Exactly one required
recordTypeMap	Optional
rfa	Exactly one required

Name	Requirement
itemLists	Exactly one required

Attributes

Name	Description	Requirement
name	A string that uniquely identifies this adapter (included in log entries)	Optional

Notes
None

Example
See the examples for the individual elements contained within the **adapter** definition.

dataField
In the **streamMap** definition, the **dataField** element maps one column from a source stream to a Reuters Field ID (FID).

Summary

```
adapter                    (required, limit one)
  |----publication                   (required, limit one)
  |----streamMaps                    (required, limit one)
  |       '----streamMap             (required)
  |              |----itemName       (required, limit one)
  |              |----serviceName    (optional)
  |              |----sequenceNumber (optional)
  |              |----itemStale      (optional)
  |              |----dataField      (required)
  |              |----updateNumber   (required)
  |              |----dateTimeField  (optional)
  |              |----FIDListField   (optional)
  |              '----nullField      (optional)
  |----recordTypeMap                 (optional)
  |       '----recordType            (optional)
  |----rfa                           (required, limit one)
  '----itemLists                     (required, limit one)
         '----itemList               (required)
                '----item            (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
name	The Reuters FID that identifies the data item that appears in this column of the source stream	Required
key	Either true or false, depending on whether this column is part of the source stream's unique key	See Notes

Notes

Each element in the **streamMap** section of the input adapter map file must represent a column in the row definition of the target source stream. (The order of the streamMap elements must mirror the order of the columns in the RowDef.) If the column in the RowDef is a data item (Bid, Ask, and so on), the corresponding **streamMap** entry must be a **dataField** element for which the name attribute identifies a specific FID. Any time RMDS publishes an update tagged with that FID, the adapter sends it to Event Stream Processor source stream as a value in the corresponding row.

Use the **key** attribute to set the value to true. If this column is not part of the stream's key, you can omit the key attribute.

Example

```
<streamMap name="stream1">
        <itemName key="true"/>
        <FIDListField />
        <!-- serviceName / -->
        <sequenceNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <dataField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

The example shown above maps columns 5–8 of stream1 to the Reuters FIDs BID, ASK, TRDPRC_1, and ACVOL_1.

dateTimeField

In the **streamMap** definition, the **dateTimeField** element maps a Reuters date or time FID (or one of each) to a date column, a timestamp column, or both, in a stream.

Summary

```
adapter                    (required, limit one)
  |----publication                   (required, limit one)
  |----streamMaps                    (required, limit one)
  |      '----streamMap              (required)
```

```
  |             |----itemName         (required, limit one)
  |             |----serviceName      (optional)
  |             |----sequenceNumber   (optional)
  |             |----itemStale        (optional)
  |             |----dataField        (required)
  |             |----updateNumber     (required)
  |             |----dateTimeField    (optional)
  |             |----FIDListField     (optional)
  |             '----nullField        (optional)
  |----recordTypeMap                  (optional)
  |      '----recordType              (optional)
  |----rfa                            (required, limit one)
  '----itemLists                      (required, limit one)
         '----itemList                (required)
                '----item             (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
dateName	The FID of the date value provided by RMDS	See Notes
timeName	The FID of the time value provided by RMDS	See Notes

Notes
The most commonly used datatype for date/time information in Event Stream Processor data streams is dateTime, which combines both date and time. In most cases, however, the updates provided by RMDS and brought in to the Event Stream Processor by the Reuters Marketfeed adapter use separate FIDs for date and time.

To address this discrepancy, the map file provides the dateTimeField element, which provides separate attributes for date and time, allowing you to map two FIDs (one for date, one for time) to the same column in the stream definition.

If **dateTime** is used, it must be used alone. The **dateName** and **timeName** can be used either separately or together. One of these three attributes must be used.

The value for each FID must match one listed in the FID list referenced in the Reuters-side configuration file (the FID list provided with the adapter is named `appendix_a`). This file is referenced in the configuration file `rfasub.cfg`.

Example

```
<streamMap name="stream1">
        <itemName key="true"/>
```

```
        <FIDListField />
        <!-- serviceName / -->
        <sequenceNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <dataField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

This example maps the TIMACT and ACTIV_DATE FIDs together to the ninth column of the Event Stream Processor source stream named stream1.

FIDListField

In the **streamMap** definition, the **FIDListField** element maps all of the Reuters FIDs with their values for an event to the Event Stream Processor source stream.

Summary

```
adapter                    (required, limit one)
  |----publication                    (required, limit one)
  |----streamMaps                     (required, limit one)
  |      '----streamMap               (required)
  |             |----itemName         (required, limit one)
  |             |----serviceName      (optional)
  |             |----sequenceNumber   (optional)
  |             |----itemStale        (optional)
  |             |----dataField        (required)
  |             |----updateNumber     (required)
  |             |----dateTimeField    (optional)
  |             |----FIDListField     (optional)
  |             '----nullField        (optional)
  |----recordTypeMap                  (optional)
  |      '----recordType              (optional)
  |----rfa                            (required, limit one)
  '----itemLists                      (required, limit one)
         '----itemList                (required)
                '----item             (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
name	A string that will appear in any adapter-related log entries	Optional

Notes
None

Example

```
<streamMap name="stream1">
        <itemName key="true"/>
        <FIDListField />
        <!-- serviceName / -->
        <sequenceNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <dataField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

In this example, the second column of the source stream is identified as the one that carries the FIDList string of any update from the adapter.

item

The **item** element is used to identify a RIC to which the Reuters Marketfeed adapter subscribes.

Summary

```
adapter                    (required, limit one)
  |----publication                   (required, limit one)
  |----streamMaps                    (required, limit one)
  |       '----streamMap             (required)
  |              |----itemName       (required, limit one)
  |              |----serviceName    (optional)
  |              |----sequenceNumber (optional)
  |              |----itemStale      (optional)
  |              |----dataField      (required)
  |              |----updateNumber   (required)
  |              |----dateTimeField  (optional)
  |              |----FIDListField   (optional)
  |              '----nullField      (optional)
  |----recordTypeMap                 (optional)
  |       '----recordType            (optional)
  |----rfa                           (required, limit one)
  '----itemLists                     (required, limit one)
         '----itemList               (required)
                '----item            (optional)
```

Parent
itemList

Children
None

Attributes

Name	Description	Requirement
name	An RIC to which the adapter subscribes	Required
rfaQueue	A name for the rfaQueue, which, if provided, replaces the default rfaQueue name and causes separate thread to be used for this queue	Optional
service	The name of a Reuters Service that provides incoming data through RMDS	Optional if already specified in the parent itemList or itemLists element, otherwise required
stream	The source stream on which updates for this RIC are brought to the Event Stream Processor	Optional if already specified in the parent itemList or itemLists element, otherwise required

Notes
The value for the name attribute must match one listed in the `appendix_a` file referenced in the Reuters-side configuration file (`rfasub.cfg` is the name of the file provided with the adapter).

If you specify a stream name here, updates for this RIC are brought in to the Event Stream Processor on that stream. If you do not specify a stream here, the stream specified at the itemList level is used.

The stream you specify must match one of the streamMaps defined elsewhere in the map file by the value of the streamMap's name attribute.

Example

```
<itemLists service="SSL_PUB" stream="stream1">
        <itemList service="IDN_RDF" >
                <item name="EUR=" />
                <item name="EURJPY=" stream="stream6" />
        </itemList>
</itemLists>
```

These two **item** elements subscribe the adapter to the RICs EUR and EURJPY. The EUR updates are sent to the stream1, which is set in the itemLists element. The EURJPY updates are sent to stream6, since the **item** level stream attribute overrides the **itemLists** level attribute.

itemList

The **itemList** element contains one or more instances of the **item** element.

Summary

```
adapter                    (required, limit one)
  |----publication                    (required, limit one)
  |----streamMaps                     (required, limit one)
  |      '----streamMap               (required)
  |             |----itemName         (required, limit one)
  |             |----serviceName      (optional)
  |             |----sequenceNumber   (optional)
  |             |----itemStale        (optional)
  |             |----dataField        (required)
  |             |----updateNumber     (required)
  |             |----dateTimeField    (optional)
  |             |----FIDListField     (optional)
  |             '----nullField        (optional)
  |----recordTypeMap                  (optional)
  |      '----recordType              (optional)
  |----rfa                            (required, limit one)
  '----itemLists                      (required, limit one)
         '----itemList                (required)
                '----item             (optional)
```

Parent

itemLists

Children

Name	Requirement
item	Zero or more required

Attributes

Name	Description	Requirement
rfaQueue	A name for the rfaQueue which, if provided, replaces the default rfaQueue name and causes a separate thread to be used for this queue	Optional
service	The name of a Reuters Service that provides incoming data through RMDS	Optional if already specified in the parent itemLists element or in all child item elements, otherwise required

Name	Description	Requirement
stream	The name of an Event Stream Processor source stream receives updates on the RICs specified in this list of items	Optional if already specified in the parent itemLists element or in all child item elements, otherwise required

Notes

Configure the adapter to push updates for every item in this section to the specified stream. However, you can override the stream specification at the item level.

The adapter supports more than one itemList element under itemLists; this allows you to configure one instance of the adapter to direct updates from two or more groups of RICs to different Event Stream Processor source streams.

The stream you specify must match, by the value of the name attribute, one of the streamMaps defined elsewhere in the map file.

Use the rfaQueue attribute to control scalability.

Example

```
<itemLists service="SSL_PUB" stream="stream1">
        <itemList service="IDN_RDF" >
                <item name="EUR=" />
                <item name="EURJPY=" stream="stream6" />
        </itemList>
</itemLists>
```

This itemList element sets the service attribute to IDN_RDF, overriding the SSL_PUB service attribute defined in the parent itemLists element.

itemLists

The **itemLists** element contains one or more instances of the **itemList** element.

Summary

```
adapter                    (required, limit one)
  |----publication                    (required, limit one)
  |----streamMaps                     (required, limit one)
  |      '----streamMap               (required)
  |             |----itemName         (required, limit one)
  |             |----serviceName      (optional)
  |             |----sequenceNumber   (optional)
  |             |----itemStale        (optional)
  |             |----dataField        (required)
  |             |----updateNumber     (required)
  |             |----dateTimeField    (optional)
  |             |----FIDListField     (optional)
  |             '----nullField        (optional)
  |----recordTypeMap                  (optional)
```

```
  |       '----recordType              (optional)
  |----rfa                             (required, limit one)
  '----itemLists                       (required, limit one)
         '----itemList                 (required)
                '----item              (optional)
```

Parent
adapter

Children

Name	Requirement
itemList	One required, two or more supported

Attributes

Name	Description	Requirement
name	A string that appears in any adapter-related log entries	Optional
rfaQueue	A name for the rfaQueue which, if provided, replaces the default rfaQueue name and causes a separate thread to be used for this queue	Optional
service	The name of a Reuters service that provides incoming data through RMDS	Optional if specified in the child itemLists or item elements so that all child item elements either specify or inherit it, otherwise required
stream	The name of an Event Stream Processor source stream that receives updates on the RICs specified in the item lists in this section (a default that can be overridden at the item level)	Optional if specified in the child itemLists and/or item elements so that all child item elements either specify or inherit it, otherwise required

Notes
Each **itemList** instance in this section is a list of one or more of the RICs to which the adapter subscribes.

Example

```
<itemLists service="SSL_PUB" stream="stream1">
        <itemList service="IDN_RDF" >
                <item name="EUR=" />
                <item name="EURJPY=" stream="stream6" />
        </itemList>
</itemLists>
```

This itemLists element sets the service attribute to SSL_PUB and the stream attribute to stream1. These attributes are either inherited or overridden at the **itemList** and/or **item** level.

itemName

In the **streamMap** definition, the **itemName** element identifies the row in the Event Stream Processor source stream that carries the RIC from the RMDS update.

Summary

```
adapter                    (required, limit one)
  |----publication                    (required, limit one)
  |----streamMaps                     (required, limit one)
  |       '----streamMap              (required)
  |              |----itemName        (required, limit one)
  |              |----serviceName     (optional)
  |              |----sequenceNumber  (optional)
  |              |----itemStale       (optional)
  |              |----dataField       (required)
  |              |----updateNumber    (required)
  |              |----dateTimeField   (optional)
  |              |----FIDListField    (optional)
  |              '----nullField       (optional)
  |----recordTypeMap                  (optional)
  |       '----recordType             (optional)
  |----rfa                            (required, limit one)
  '----itemLists                      (required, limit one)
         '----itemList                (required)
                '----item             (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
key	True or false, depending on whether or not this column is part of the source stream's unique key	See first note

Notes
You do not need to use the **key** attribute. It is present for backward compatibility.

Insert the **itemName** element in the **streamMap** to correspond with the column in the RowDef that carries the RIC or symbol. If this column is part of the source stream's key, set the key attribute to true.

This element is one of the "pseudofields" that specify data items that are not part of the data feed coming directly from RMDS.

Example

```
<streamMap name="stream1">
        <itemName key="true"/>
        <FIDListField />
        <!-- serviceName / -->
        <sequenceNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <dataField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

The first column of the stream is identified as the one that carries the RIC value of any update from the adapter. It is also identified as part of the stream's key.

itemStale

In the **streamMap** definition, the **itemStale** element identifies a column in the Event Stream Processor source stream that carries a flag indicating whether incoming RMDS data has gone stale.

Summary

```
adapter                    (required, limit one)
  |----publication                   (required, limit one)
  |----streamMaps                    (required, limit one)
  |       '----streamMap             (required)
  |              |----itemName       (required, limit one)
  |              |----serviceName    (optional)
  |              |----sequenceNumber (optional)
  |              |----itemStale      (optional)
  |              |----dataField      (required)
  |              |----updateNumber   (required)
  |              |----dateTimeField  (optional)
  |              |----FIDListField   (optional)
  |              '----nullField      (optional)
  |----recordTypeMap                 (optional)
  |       '----recordType            (optional)
  |----rfa                           (required, limit one)
  '----itemLists                     (required, limit one)
         '----itemList               (required)
                '----item            (optional)
```

Parent

streamMap

Children
None

Attributes

Name	Description	Requirement
name	A string that appears in any adapter-related log entries	Required

Notes
Use this element in the streamMap if one of the columns in the source stream is a "stale" flag.

RMDS itself does not supply a stale flag with regular market data, although it may pass along such a flag if it is provided by another service you are subscribing to via RMDS. If this element is used in the streamMap, the adapter sends out an update value of 1 if it receives a stale flag from RMDS, or stops receiving any data from RMDS.

Example

```
<streamMap name="stream1">
        <itemName key="true"/>
        <FIDListField />
        <!-- serviceName / -->
        <sequenceNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <dataField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

The fourth column of the source stream is identified as the one that is updated if the adapter receives a stale notification, or stops receiving data from RMDS.

nullField

In a **streamMap**, the **nullField** element acts as a placeholder that always delivers a NULL value to the stream. This lets you add extra fields to a source stream to get the configuration you want.

Summary

```
adapter                    (required, limit one)
  |----publication                   (required, limit one)
  |----streamMaps                    (required, limit one)
  |      '----streamMap              (required)
  |             |----itemName        (required, limit one)
  |             |----serviceName     (optional)
  |             |----sequenceNumber  (optional)
  |             |----itemStale       (optional)
```

```
  |             |----dataField       (required)
  |             |----updateNumber    (required)
  |             |----dateTimeField   (optional)
  |             |----FIDListField    (optional)
  |             '----nullField       (optional)
  |----recordTypeMap                 (optional)
  |       '----recordType            (optional)
  |----rfa                           (required, limit one)
  '----itemLists                     (required, limit one)
         '----itemList               (required)
                '----item            (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
name	A string that appears in any adapter-related log entries	Optional

Notes
When experimenting with a project, you can use a **nullField** to temporarily stop feeding data into one column of the stream. In this case, you can simply keep the name of the **dataField** that you are temporarily replacing, as in the following example.

Example

```
<streamMap name="stream1">
        <itemName key="true"/>
        <FIDListField />
        <!-- serviceName / -->
        <sequenceNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <nullField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

The seventh column of the source stream is identified as a placeholder receives a null value in each update from the adapter. It includes the name of the **dataField** that it replaces for debugging purposes.

publication

The **publication** element specifies basic operating parameters for this instance of the adapter.

Summary

```
adapter                    (required, limit one)
  |----publication                    (required, limit one)
  |----streamMaps                     (required, limit one)
  |      '----streamMap               (required)
  |             |----itemName         (required, limit one)
  |             |----serviceName      (optional)
  |             |----sequenceNumber   (optional)
  |             |----itemStale        (optional)
  |             |----dataField        (required)
  |             |----updateNumber     (required)
  |             |----dateTimeField    (optional)
  |             |----FIDListField     (optional)
  |             '----nullField        (optional)
  |----recordTypeMap                  (optional)
  |      '----recordType              (optional)
  |----rfa                            (required, limit one)
  '----itemLists                      (required, limit one)
         '----itemList                (required)
                '----item             (optional)
```

Parent

adapter

Children

None

Attributes

Name	Description	Requirement
flushInterval	Specify the number of microseconds the adapter allows events to accumulate before sending them to the Event Stream Processor. A non zero flushInterval makes event accumulation time-based.	Optional
intraSubscribeDelay	Specify the number of milliseconds the adapter pauses between subscription requests.	Optional

Name	Description	Requirement
maxRecordsPer-Block	Specify the maximum number of accumulated events that the adapter should send to the Event Stream Processor at a time. This reduces the size of each transaction or envelope fragment when there is a large number of accumulated events. For example, if 140 events have accumulated and maxRecordsPerBlock is set to 50, the adapter will send the envelope or transaction as three fragments.	Optional
name	Specify a string that identifies the adapter instance in log file entries.	Required
pendingLimit	Specify the number of events that may accumulate before the adapter sends them in to the Event Stream Processor. Using a pendingLimit makes event accumulation count-based.	Optional
retryInterval	Specify the number of seconds for which the adapter attempts to connect to RMDS before shutting down.	Required
sendAsTransactions	Set to true to treat a group of updates as a single transaction or false to treat them as separate transactions within an envelope.	Optional

Notes

You can optimize the adapter's performance using the **pendingLimit** and **flushInterval** attributes, along with the **maxRecordsPerBlock** and **sendAsTransactions** attributes from the Pub/Sub interface that the adapter uses to communicate with the Event Stream Processor.

Some venues send initial images as multi part messages, which may produce large data sets. The intraSubscribeDelay attribute paces these subscriptions and prevents the adapter from being overwhelmed by initial images. The default value is zero, which is suitable for short RIC lists. When intraSubscribeDelay is set to a nonzero value, the adapter pauses between subscription requests for milliseconds. The suggested value is ten (10).

Example

```
<publication name="RMDS Adapter - low latency" retryInterval="5"
        flushInterval="0" pendingLimit="0" sendAsTransactions="0" />
```

recordType

The **recordType** element maps a stream to a predefined set of FIDs.

Summary

```
adapter                    (required, limit one)
  |----publication                   (required, limit one)
```

```
  |----streamMaps                    (required, limit one)
  |       '----streamMap             (required)
  |              |----itemName       (required, limit one)
  |              |----serviceName    (optional)
  |              |----sequenceNumber (optional)
  |              |----itemStale      (optional)
  |              |----dataField      (required)
  |              |----updateNumber   (required)
  |              |----dateTimeField  (optional)
  |              |----FIDListField   (optional)
  |              '----nullField      (optional)
  |----recordTypeMap                 (optional)
  |       '----recordType            (optional)
  |----rfa                           (required, limit one)
  '----itemLists                     (required, limit one)
         '----itemList               (required)
                '----item            (optional)
```

Parent
recordTypeMap

Children
None

Attributes

Name	Description	Requirement
number	The ID of a recordType defined in Reuters configuration	Required
stream	The name of a stream to which this record will be mapped	Required

Notes
The pre-defined record specified by **recordType** must match all the columns in the stream's definition. Otherwise, these columns must be explicitly mapped in a **streamMap** configuration.

Example

```
<recordTypeMap>
        <recordType number="123" stream="eqInput"/>
</recordTypeMap>
```

This example maps a set of FIDs pre-defined as record "123" to the source stream eqInput.

recordTypeMap

The **recordTypeMap** element contains one or more instances of **recordType**.

Summary

```
adapter                    (required, limit one)
  |----publication                   (required, limit one)
```

```
  |----streamMaps                    (required, limit one)
  |       '----streamMap             (required)
  |              |----itemName       (required, limit one)
  |              |----serviceName    (optional)
  |              |----sequenceNumber (optional)
  |              |----itemStale      (optional)
  |              |----dataField      (required)
  |              |----updateNumber   (required)
  |              |----dateTimeField  (optional)
  |              |----FIDListField   (optional)
  |              '----nullField      (optional)
  |----recordTypeMap                 (optional)
  |       '----recordType            (optional)
  |----rfa                           (required, limit one)
  '----itemLists                     (required, limit one)
         '----itemList               (required)
                '----item            (optional)
```

Parent
adapter

Children

Name	Requirement
recordType	Zero or more supported

Attributes
None

Notes
A stream must have either a recordTypeMap or a streamMap; it cannot have both.

The pre-defined record must match all the columns in the stream's definition to use the implicit mapping provided by recordTypeMap. Otherwise, these columns must be explicitly mapped in a streamMap configuration.

Example

```
<recordTypeMap>
        <recordType number="123" stream="eqInput"/>
</recordTypeMap>
```

This example maps a set of FIDs pre-defined as record "123" to the source stream eqInput.

rfa

The **rfa** element links the input adapter map file to the Reuters-side configuration file.

Summary

```
adapter                    (required, limit one)
  |----publication                    (required, limit one)
  |----streamMaps                     (required, limit one)
  |       '----streamMap              (required)
  |              |----itemName        (required, limit one)
  |              |----serviceName     (optional)
  |              |----sequenceNumber  (optional)
  |              |----itemStale       (optional)
  |              |----dataField       (required)
  |              |----updateNumber    (required)
  |              |----dateTimeField   (optional)
  |              |----FIDListField    (optional)
  |              '----nullField       (optional)
  |----recordTypeMap                  (optional)
  |       '----recordType             (optional)
  |----rfa                            (required, limit one)
  '----itemLists                      (required, limit one)
         '----itemList                (required)
                '----item             (optional)
```

Parent

adapter

Children

None

Attributes

Name	Description	Requirement
config	The absolute path and file name of the Reuters-side configuration file for subscription (the sample file supplied with the adapter is `$ESP_REUTERS_HOME/config/rfasub.cfg`.	Required
configDatabaseName	Must be set to RFA.	Required
enumFile	The full path name of the Reuters-supplied file that lists each enumerated type along with the range of values it can take.	Required
fidFile	The full path name of the Reuters-supplied file that lists all of the valid FIDs.	Required

Name	Description	Requirement
sessionName	A reference to a session name defined in the Reuters-side configuration file for subscription.	Required
blank	Specify a marker to use for blanks	Optional
blankInt32	Specify a marker to use for blank `Int32` fields	Optional
blankInt64	Specify a marker to use for blank `Int64` fields	Optional
blankMoney	Specify a marker to use for blank `Money` fields	Optional
blankString	Specify a marker to use for blank `String` fields	Optional
blankDate	Specify a marker to use for blank `Date` fields	Optional
blankTimestamp	Specify a marker to use for blank `Timestamp` fields	Optional

Notes
None

Example

```
<rfa config="$ESP_REUTERS_HOME/config/rfasub.cfg"
        sessionName="Session1" />
```

This example points the Reuters Marketfeed adapter to the Reuters-side configuration in the file `rfasub.cfg`. The list line in this configuration file is:

```
\Sessions\Session1\connectionList =
"Connection_SSLED"
```

This line defines a session name that is referenced by other lines in the configuration file. When the map file references a session name in the **sessionName** attribute, it links the adapter to the Reuters-side configuration parameters identified by that name.

sequenceNumber

In the **streamMap** definition, the **sequenceNumber** element maps a column in the source stream that is populated by a unique number generated by the adapter, not provided as part of the data from RMDS.

Summary

```
adapter                    (required, limit one)
  |----publication                   (required, limit one)
  |----streamMaps                    (required, limit one)
  |      '----streamMap              (required)
  |             |----itemName        (required, limit one)
  |             |----serviceName     (optional)
  |             |----sequenceNumber  (optional)
  |             |----itemStale       (optional)
```

```
  |             |----dataField       (required)
  |             |----updateNumber    (required)
  |             |----dateTimeField   (optional)
  |             |----FIDListField    (optional)
  |             '----nullField       (optional)
  |----recordTypeMap                 (optional)
  |       '----recordType            (optional)
  |----rfa                           (required, limit one)
  '----itemLists                     (required, limit one)
         '----itemList               (required)
                '----item            (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
key	True or false, depending on whether this column is part of the source stream's unique key	See Notes
name	A string that appears in log entries	Optional

Notes
The adapter maintains a separate counter for each RIC to which it is subscribed. Each time it receives an update for a RIC, it increments the counter for that RIC. This number is the one sent to the source stream column mapped by the sequenceNumber element.

Many source stream definitions include a column specification similar to:

```
<Column datatype="int32" name="Id"/>
```

This line specifies a unique ID for the source stream. The **sequenceNumber** pseudo field is a good match for this column in the input adapter map file

You must use the key attribute to set the value to true. If this column is not part of the stream's key, you can omit this.

Example

```
<streamMap name="stream1">
        <itemName key="true"/>
        <FIDListField />
        <!-- serviceName / -->
        <sequenceNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
```

```
        <dataField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

The third column of the source stream is mapped to the sequence number provided by the adapter. This column is also identified as part of the source stream's unique key.

serviceName

In the **streamMap** definition, the **serviceName** element maps a column in the source stream to the service identifier that the adapter provides as part of the envelope for each update.

Summary

```
adapter                    (required, limit one)
  |----publication                    (required, limit one)
  |----streamMaps                     (required, limit one)
  |       '----streamMap              (required)
  |              |----itemName        (required, limit one)
  |              |----serviceName     (optional)
  |              |----sequenceNumber  (optional)
  |              |----itemStale       (optional)
  |              |----dataField       (required)
  |              |----updateNumber    (required)
  |              |----dateTimeField   (optional)
  |              |----FIDListField    (optional)
  |              '----nullField       (optional)
  |----recordTypeMap                  (optional)
  |       '----recordType             (optional)
  |----rfa                            (required, limit one)
  '----itemLists                      (required, limit one)
         '----itemList                (required)
                '----item             (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
key	True or false, depending on whether this column is part of the source stream's unique key	See Notes

Notes
You must use the key attribute to set the value to true. If this column is not part of the stream's key, you can ommit this.

Example

```
<streamMap name="stream1">
        <itemName key="true"/>
        <FIDListField />
        <!-- serviceName / -->
        <sequenceNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <dataField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

In this example, no column of the source stream is mapped to the service name provided by the adapter because it is commented out.

streamMap

The **streamMap** element contains the mappings between the columns of an Event Stream Processor source stream and the RMDS FIDs being subscribed to by the adapter.

Summary

```
adapter                    (required, limit one)
  |----publication                    (required, limit one)
  |----streamMaps                     (required, limit one)
  |       '----streamMap              (required)
  |              |----itemName        (required, limit one)
  |              |----serviceName     (optional)
  |              |----sequenceNumber  (optional)
  |              |----itemStale       (optional)
  |              |----dataField       (required)
  |              |----updateNumber    (required)
  |              |----dateTimeField   (optional)
  |              |----FIDListField    (optional)
  |              '----nullField       (optional)
  |----recordTypeMap                  (optional)
  |       '----recordType             (optional)
  |----rfa                            (required, limit one)
  '----itemLists                      (required, limit one)
         '----itemList                (required)
                '----item             (optional)
```

Parent

streamMaps

Children

The following child elements are defined for **streamMap**. These child elements can occur in any order, but for a specific **streamMap**, the order of the child elements must mirror the order

of the columns of the source stream (as defined in the project). This is how the adapter is configured to deliver RMDS updates to the appropriate rows in the source stream.

Name	Requirement
dataField	One required, two or more supported
dateTimeField	Zero or more supported
itemName	One required, two or more supported
itemStale	Zero or one supported
sequenceNumber	Zero or one supported
serviceName	Zero or one supported

Attributes

Name	Description	Requirement
name	References the source stream to which the RMDS updates are mapped. Must match the name of a source stream defined in the Event Stream Processor project.	Required
opcode	Defines the operation the adapter performs when sending updates to the source stream. Possible values are insert and upsert. The insert operation adds new updates to the end of the source stream. The upsert operation replaces an existing source stream entry if its key matches the entry's key; if not, the update is added.	Optional (default value is upsert)

Notes
None

Example

```
<streamMaps>
        <streamMap name="stream1">
                <itemName key="true"/>
                <FIDListField />
                <!-- serviceName / -->
                <sequenceNumber />
                <itemStale/>
                <dataField name="BID"/>
                <dataField name="ASK"/>
                <dataField name="TRDPRC_1"/>
                <dataField name="ACVOL_1"/>
                <dateTimeField timeName="TIMACT"
dateName="ACTIV_DATE"/>
        </streamMap>
</streamMaps>
```

This example maps a set of the adapter's updates to an Event Stream Processor source stream named stream1. All updates going to this source stream are added using the upsert mode. The RICs for which updates are sent to this source stream are specified in an itemList elsewhere in the map file that also references stream1.

streamMaps

The **streamMaps** element contains one or more instances of the **streamMap** element.

Summary

```
adapter                    (required, limit one)
  |----publication                    (required, limit one)
  |----streamMaps                     (required, limit one)
  |       '----streamMap              (required)
  |              |----itemName        (required, limit one)
  |              |----serviceName     (optional)
  |              |----sequenceNumber  (optional)
  |              |----itemStale       (optional)
  |              |----dataField       (required)
  |              |----updateNumber    (required)
  |              |----dateTimeField   (optional)
  |              |----FIDListField    (optional)
  |              '----nullField       (optional)
  |----recordTypeMap                  (optional)
  |       '----recordType             (optional)
  |----rfa                            (required, limit one)
  '----itemLists                      (required, limit one)
         '----itemList                (required)
                '----item             (optional)
```

Parent

adapter

Children

Name	Requirement
streamMap	One required, two or more supported

Attributes

None

Notes

Each **streamMap** instance in this section maps incoming FIDs from the Reuters adapter to columns in an Event Stream Processor source stream.

Example

```
<streamMaps>
        <streamMap name="stream1">
                <itemName key="true"/>
```

```
                <FIDListField />
                <!-- serviceName / -->
                <sequenceNumber />
                <itemStale/>
                <dataField name="BID"/>
                <dataField name="ASK"/>
                <dataField name="TRDPRC_1"/>
                <dataField name="ACVOL_1"/>
                <dateTimeField timeName="TIMACT"
dateName="ACTIV_DATE"/>
        </streamMap>
</streamMaps>
```

updateNumber

In the **streamMap** definition, the **updateNumber** element maps a column in the Event Stream Processor source stream that is populated by a unique number generated by the adapter, not provided as part of the data from RMDS.

Summary

```
adapter                    (required, limit one)
  |----publication                    (required, limit one)
  |----streamMaps                     (required, limit one)
  |       '----streamMap              (required)
  |             |----itemName         (required, limit one)
  |             |----serviceName      (optional)
  |             |----sequenceNumber   (optional)
  |             |----itemStale        (optional)
  |             |----dataField        (required)
  |             |----updateNumber     (required)
  |             |----dateTimeField    (optional)
  |             |----FIDListField     (optional)
  |             '----nullField        (optional)
  |----recordTypeMap                  (optional)
  |       '----recordType             (optional)
  |----rfa                            (required, limit one)
  '----itemLists                      (required, limit one)
         '----itemList                (required)
                '----item             (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
key	True or false, depending on whether this column is part of the source stream's unique key	See Notes
name	A string that appears in log entries	Optional

Notes

The adapter maintains a separate counter for each RIC to which it is subscribed. Each time it receives an update for a RIC, it increments the counter for that RIC. This number is the one sent to the source stream column mapped by the **updateNumber** element.

Many source stream definitions include a column specification similar to:

```
<Column datatype="integer" name="Id"/>
```

This line specifies a unique ID for the source stream. The **updateNumber** pseudo field is a good match for this column in the input adapter map file.

You must use the key attribute to set the value to true. If this column is not part of the stream's key, you can omit this attribute.

Example

```
<streamMap name="stream1">
        <itemName key="true"/>
        <FIDListField />
        <!-- serviceName / -->
        <updateNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <dataField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

The third column of the source stream is mapped to the update number provided by the adapter. This column is also identified as part of the source stream's unique key.

Output Adapter Map File XML Syntax

The syntax of the map file for a Reuters Marketfeed output adapter.

The following listing shows the structure of an output adapter map file. Each line of this summary lists one element of the map file structure. See the topics for each element for details.

```
adapter                                 (required, limit one)
  |----rfa                              (required, limit one)
  '----subscriptions                    (required, limit one)
         '----subscription              (required)
```

```
                '----stream                (required)
                       |----name           (required, limit one)
                       |----service        (optional)
                       |      '----enum    (required)
                       |----stale          (optional)
                       |----field          (required)
                       '----constant       (optional)
```

adapter

The **adapter** element is the root element of the output map file.

Summary

```
adapter                                    (required, limit one)
  |----rfa                                 (required, limit one)
  '----subscriptions                       (required, limit one)
         '----subscription                 (required)
                '----stream                (required)
                       |----name           (required, limit one)
                       |----service        (optional)
                       |      '----enum    (required)
                       |----stale          (optional)
                       |----field          (required)
                       '----constant       (optional)
```

Nest all configuration elements between the start and end **adapter** tags.

Parent

None

Children

The following child elements are defined for **adapter**. All of these elements must be present in the specified order.

Name	Requirement
rfa	Exactly one required
subscriptions	Exactly one required

Attributes

None

Notes

None

Example

See the examples for the child elements.

constant

The **constant** element defines a data item with a constant value that is published to RMDS by the adapter.

Summary

```
adapter                                  (required, limit one)
  |----rfa                               (required, limit one)
  '----subscriptions                     (required, limit one)
         '----subscription               (required)
                '----stream              (required)
                       |----name         (required, limit one)
                       |----service      (optional)
                       |       '----enum (required)
                       |----stale        (optional)
                       |----field        (required)
                       '----constant     (optional)
```

Parent

stream

Children

None

Attributes

Name	Description	Requirement
name	The name associated with this data item in the image published by the adapter	Required
value	The value of this constant (always the same whenever this data item is published to RMDS)	Required

Notes

At start-up, the adapter publishes a complete image, containing all data items defined in the map file, to RMDS. After that, the adapter publishes updated values for data items when they change, unless the Event Stream Processor goes stale and then recovers. This means that the value for **constant** is published only when a complete image is published.

Example

```
<stream name="stream1" >
        <name column="0"/>
        <stale column="3" name="ACVOL_1"/>
        <field column="1" name="DSPLY_NAME" />
        <field column="4" name="BID" precision="47" />
        <field column="5" name="ASK" precision="0" />
        <field column="6" name="TRDPRC_1"/>
        <field column="7" name="ACVOL_1"/>
```

```
        <constant name="PROD_PERM" value="1"/>
</stream>
```

This example defines a constant called PROD_PERM, with the constant value 1, to be published with data values from the stream1 under the publication name subscription1.

enum

The **enum** element maps the value of the Event Stream Processor stream's service column to a unique string that is prepended to the name element of an update published to RMDS by the adapter.

Summary

```
adapter                                    (required, limit one)
  |----rfa                                 (required, limit one)
  '----subscriptions                       (required, limit one)
         '----subscription                 (required)
                '----stream                (required)
                       |----name           (required, limit one)
                       |----service        (optional)
                       |      '----enum    (required)
                       |----stale          (optional)
                       |----field          (required)
                       '----constant       (optional)
```

If the Event Stream Processor stream from which you are publishing handles data items for the same symbol from different sources (the "Ask" price for IBM from NASDAQ and from S&P, for example), you can use the service and enum attributes in the output adapter map file to configure the adapter to differentiate between updates of the same value for the same symbol from different sources.

Parent

service

Children

None

Attributes

Name	Description	Requirement
value	A possible value for the data stream column specified by the **service** element	Required
prefix	The string prepended to the value of the **name** element when it publishes updates received from the Event Stream Processor with the **service** value that matches **prefix**	Required

Notes

The **service** element in the output adapter map file must contain one **enum** element for each possible value in the source column.

Example

```
<service column="2" delim="_">
        <enum value="RDF" prefix="R"/>
        <enum value="ISFS" prefix="I"/>
</service>
```

Within a service definition, each **enum** element specifies a particular service. Based on this value, the published RICs are renamed to indicate the provider of the data. Assume that RIC.X is the RIC found in the name column. If the value in column 2 is RDF, the RIC becomes "R_RIC.X". If the value in column 2 is ISFS, the RIC becomes "I_RIC.X". If neither is true, no value is published.

field

In a **stream** definition in an output adapter map file, **field** specifies a column in a stream to publish.

Summary

```
adapter                                  (required, limit one)
  |----rfa                               (required, limit one)
  '----subscriptions                     (required, limit one)
         '----subscription               (required)
                '----stream              (required)
                       |----name         (required, limit one)
                       |----service      (optional)
                       |      '----enum  (required)
                       |----stale        (optional)
                       |----field        (required)
                       '----constant     (optional)
```

Parent

stream

Children

None

Attributes

Name	Description	Requirement
column	A number that represents the position of the source column in the stream being published from (the first column in the stream has the number 0)	Required

Name	Description	Requirement
name	The FID that identifies this data value when published to RMDS	Required
precision	An integer that specifies the total number of digits after the decimal point in the published value (for example, 1.23 has a precision of 2)	Optional

Notes

Modify the value of the name attribute to indicate the source of the data item if you have defined the **parmname** and **enum** elements in this stream definition.

Include the precision attribute only for columns of datatype `double`.

Example

```
<stream name="stream1" >
        <name column="0"/>
        <stale column="3" name="ACVOL_1"/>
        <field column="1" name="DSPLY_NAME" />
        <field column="4" name="BID" precision="47" />
        <field column="5" name="ASK" precision="0" />
        <field column="6" name="TRDPRC_1"/>
        <field column="7" name="ACVOL_1"/>
        <constant name="PROD_PERM" value="1"/>
</stream>
```

The adapter is configured to publish updates from the fourth, fifth, sixth and seventh columns of the Event Stream Processor stream named stream1 as data items named BID, ASK, TRDPRC_1 and ACVOL_1, respectively.

name

In a **stream** definition in an output adapter map file, **name** specifies the column in the source stream that provides the value to use to identify each update.

Summary

```
adapter                                  (required, limit one)
  |----rfa                               (required, limit one)
  '----subscriptions                     (required, limit one)
         '----subscription               (required)
                '----stream              (required)
                       |----name         (required, limit one)
                       |----service      (optional)
                       |       '----enum (required)
                       |----stale        (optional)
                       |----field        (required)
                       '----constant     (optional)
```

Parent

stream

Children
None

Attributes

Name	Description	Requirement
column	A number that represents the position of the column in the stream that carries the stream's unique identifier (the first column in the stream is number 0)	Either column or name
name	The name of the column in the stream that carries the stream's unique identifier	Either column or name

Notes
The output adapter uses RMDS as a simple message bus; the published updates need not conform to Reuters protocols. This means that the column specified by this element does not have to be a Reuters RIC, but it must follow Reuters RIC syntax.

If the source stream's unique key is a composition of two or more columns, you can use the name element in combination with one or more instances of the service element to configure the adapter to publish updates with completely unique names.

Example

```
<stream name="stream1" >
        <name column="0"/>
        <stale column="3" name="ACVOL_1"/>
        <field column="1" name="DSPLY_NAME" />
        <field column="4" name="BID" precision="47" />
        <field column="5" name="ASK" precision="0" />
        <field column="6" name="TRDPRC_1"/>
        <field column="7" name="ACVOL_1"/>
        <constant name="PROD_PERM" value="1"/>
</stream>
```

This example identifies the first column of stream1 as its unique identifier or "key" column.

rfa

The **rfa** element provides information for configuring the Reuters side of the adapter, including an explicit reference to the Reuters-side configuration file.

Summary

```
adapter                                    (required, limit one)
  |----rfa                                 (required, limit one)
  '----subscriptions                       (required, limit one)
         '----subscription                 (required)
                '----stream                (required)
                       |----name           (required, limit one)
```

```
                        |----service       (optional)
                        |      '----enum   (required)
                        |----stale         (optional)
                        |----field         (required)
                        '----constant      (optional)
```

Parent
adapter

Children
None

Attributes

Name	Description	Requirement
serviceName	Defines a service name that is included in the header of every update sent out by the Reuters Marketfeed adapter	Optional
config	The absolute path and file name of the Reuters-side configuration file for publication (the sample file supplied with the adapter is `$ESP_REUTERS_HOME/config/rfapub.cfg`)	Required
sessionName	A reference to a session named defined in the Reuters-side configuration file for publication	Required
configDatabaseName	A reference to the Reuters database name	Optional

Notes
None

Example

```
<rfa serviceName="IDN_RDF"
        config="$ESP_REUTERS_HOME/config/rfapub.cfg"
        sessionName="Session1" configDatabaseName="RFA" />
```

This example points the Reuters Marketfeed adapter to the Reuters-side configuration in the file `rfapub.cfg`. The first four uncommented lines in this configuration file are:

```
\Connections\Connection_SSLED_MP\ipcServerName = "8105"
\Connections\Connection_SSLED_MP\connectionType = "SSLED_MP"
\Connections\Connection_SSLED_MP\entitlementData = false
\Sessions\Session1\connectionList = "Connection_SSLED_MP"
```

The last of these lines implicitly defines a session name that is defined as the **sessionName** in the map file. The other three lines from `rfapub.cfg` key on this session name. This is how

the value for **sessionName** ties this **publication** section of the map file to a configuration set in the `.cfg` file.

When the adapter publishes using this configuration, each update is identified with the **serviceName** "IDN_RDF".

service

In a **stream** definition in an output adapter map file, **service** identifies a column in the source stream that is another component of the stream's unique key.

Summary

```
adapter                                  (required, limit one)
  |----rfa                               (required, limit one)
  '----subscriptions                     (required, limit one)
         '----subscription               (required)
                '----stream              (required)
                       |----name         (required, limit one)
                       |----service      (optional)
                       |      '----enum  (required)
                       |----stale        (optional)
                       |----field        (required)
                       '----constant     (optional)
```

Parent

stream

Children

None

Attributes

Name	Description	Requirement
column	A number that represents the position of the column with the secondary key value (the first column in the stream has the number 0)	Required
delim	Specifies a character to use as the separator between a name and a prefix	Optional

Notes

The **service** element in the output adapter map file must contain one **enum** element for each possible value in the source column.

Example

```
<service column="2" delim="_">
        <enum value="RDF" prefix="R"/>
        <enum value="ISFS" prefix="I"/>
</service>
```

This section configures the adapter to test the value of the second column of every update from the Event Stream Processor stream (the value of the **name** attribute of the **stream** element).

If the value is RDF, the adapter adds the prefix "R" followed by the specified delim value to the name of the published update (the value of the **name** attribute of the **publication** element).

If the value is ISFS, the adapter adds the prefix "I" to the **name** of the published update.

stale

In a **stream** definition in an output adapter map file, the **stale** element identifies a column in the source stream for which the value changes from 0 to 1 if the stream goes stale.

Summary

```
adapter                                    (required, limit one)
  |----rfa                                 (required, limit one)
  '----subscriptions                       (required, limit one)
         '----subscription                 (required)
                '----stream                (required)
                       |----name           (required, limit one)
                       |----service        (optional)
                       |      '----enum    (required)
                       |----stale          (optional)
                       |----field          (required)
                       '----constant       (optional)
```

A stream is considered to have gone stale if, for example, one of the stream's data sources is no longer being updated.

Parent
stream

Children
None

Attributes

Name	Description	Requirement
column	A number representing the position of the column with the secondary key value (the first column in the stream has the number 0)	Required
name	A string that identifies the stale column so that it may be mapped to a FID (published)	Optional

Notes
None

Example

```
<stream name="stream1" >
        <name column="0"/>
        <stale column="3" name="ACVOL_1"/>
        <field column="1" name="DSPLY_NAME" />
        <field column="4" name="BID" precision="47" />
        <field column="5" name="ASK" precision="0" />
        <field column="6" name="TRDPRC_1"/>
        <field column="7" name="ACVOL_1"/>
        <constant name="PROD_PERM" value="1"/>
</stream>
```

This example identifies the third column of stream1 as its stale column. If the stale column is specified, the column value is published and the RIC is marked stale.

stream

In a subscription section in an output adapter map file, identifies the stream from which the adapter obtains the data it publishes to RMDS.

Summary

```
adapter                                  (required, limit one)
  |----rfa                               (required, limit one)
  '----subscriptions                     (required, limit one)
         '----subscription               (required)
                '----stream              (required)
                       |----name         (required, limit one)
                       |----service      (optional)
                       |      '----enum  (required)
                       |----stale        (optional)
                       |----field        (required)
                       '----constant     (optional)
```

Parent

subscription

Children

Name	Requirement
Name	One
Service	Optional
Stale	Optional
Field	One or more
Constant	Optional

Attributes

Name	Description	Requirement
exitOnStreamExit	This is a boolean attribute. When true,RMDS terminates if the stream exits, the Event Stream Processor exits, or the connection is lost.	Optional
finalizer	This string specifies an action to take if the specified number of heartbeat milliseconds elapse without an event being published to the Event Stream Processor.	Optional
heartbeat	This integer specifies the number of milliseconds to wait without an event being published to the Event Stream Processor before executing the finalizer action.	Optional
name	The name of the stream from which the adapter receives the data it publishes on RMDS	Required
templateNumber	A Reuters template set up in the RMDS configuration	Optional

Notes

You must define the value of the **name** attribute in the Event Stream Processor project.

Any stream in the Event Stream Processor project can map to only one **stream** section in the map file.

The **templateNumber** must be a unique identifier of the stream for which it is defined

Example

```
<stream name="stream1">
        <name column="0"/>
        <field column="4" name="TRDPRC_1"/>
        <field column="9" name="BID" precision="5"/>
</stream>
```

This example configures the Event Stream Processor to publish data from a stream named stream1.

subscription

The **subscription** element contains one or more instances of the **stream** element, enabling you to configure the adapter to receive data from one or more streams.

Summary

```
adapter                                    (required, limit one)
  |----rfa                                 (required, limit one)
  '----subscriptions                       (required, limit one)
         '----subscription                 (required)
                '----stream                (required)
                       |----name           (required, limit one)
```

```
                        |----service       (optional)
                        |      '----enum   (required)
                        |----stale         (optional)
                        |----field         (required)
                        '----constant      (optional)
```

The output adapter map file can contain two or more **subscription** sections. At runtime, the publishing mechanism for each **subscription** section is instantiated on a separate thread, which provides scalability.

Parent
subscriptions

Children

Name	Requirement
stream	One or more

Attributes

Name	Description	Requirement
name	A name for this subscription that appears in updates published on RMDS and in log file entries	Required

Notes
None

Example

```
<subscriptions>
    <subscription name="subscription1" >
        <stream name="stream1" >
            <name column="0"/>
            <field column="4" name="BID"/>
            <field column="5" name="ASK"/>
            <field column="6" name="TRDPRC_1"/>
            <field column="7" name="ACVOL_1"/>
            <constant name="PROD_PERM" value="1"/>
        </stream>
    </subscription>
</subscriptions>
```

This example configures the adapter to publish some columns from stream1 using the name subscription1.

subscriptions

The **subscriptions** element contains one or more **subscription** elements.

Summary

```
adapter                                    (required, limit one)
  |----rfa                                 (required, limit one)
  '----subscriptions                       (required, limit one)
         '----subscription                 (required)
                '----stream                (required)
                       |----name           (required, limit one)
                       |----service        (optional)
                       |      '----enum    (required)
                       |----stale          (optional)
                       |----field          (required)
                       '----constant       (optional)
```

Parent

adapter

Children

Name	Requirement
Subscription	One or more

Attributes

None

Notes

Each **subscription** instance in this section defines one set of data that the adapter publishes to RMDS.

Example

See the example for an individual **subscription** instance.

Logging Facilities

The Reuters Marketfeed adapter supports two different logging mechanisms.

In addition to its own logging mechanism, the Reuters Marketfeed adapter can utilize Reuters-side logging. You can use both of these mechanisms to check the adapter's performance and diagnose problems.

You can configure these logs to be written to stderr, syslog, or both.

Adapter Logging

The Reuters Marketfeed adapter supports the same options for logging as the Event Stream Processor.

The **-d** option sets the debug level (0=emergency messages only, 7=all messages).

The **-l** option tells the adapter to write log messages to stderr, syslog, both, or neither. If you use the **-l** option to direct adapter log messages to stderr, you may also want to redirect stderr to a file.

The name attribute of the **publication** element in the input adapter map file specifies a descriptive text string that is logged to help identify how the adapter was configured. For example, lines 3–6 of `subexample.xml` specify the **publication** element for a subscribing instance of the Reuters Marketfeed adapter, as follows:

```
<publication
    name="RMDS Adapter exp"
    retryInterval="5"
/>
```

As the adapter connects with and interacts with Event Stream Processor, this configuration causes the adapter to write log messages similar to:

```
(0.123) @1 INFO: Configuring publication with name RMDS Adapter exp
```

The first two fields are the timestamp (in seconds since start-up) and the thread number, respectively. The base time for the timestamp, along with other information, is written to the log file on start-up as shown in the following example. To convert the timestamp to a date and time, simply add the number of seconds to the base time.

```
(63359098041.768) @1 NOTICE:Base time is 10/08/08-17:27:21
(0.001) @1 NOTICE:insta-a sub -c cimtest:-- -d 7
-f /home/sybase/support/1.0.3/ReutersAdapter/quotes.map.xml
-l 1 -p tigris:12192 -P 1
(0.001) @1 NOTICE:pid=28649
(0.001) @1 DEBUG:Using ESP_RMDS_SUBSCRIBE_DEBUG_LEVEL=7ll/
i86pc_64_spro/bin/rmds version:
1.0.3a-alpha_r18674M
```

Page Data and Partial Page Updates

Some Reuters data comes as pages which use Marketfeed partial format. Each page consists of multiple lines; initially sent as a snapshot. Page data is supported without any special configuration. The following extract from an adapter log file shows the delivery of the initial page image (which is displayed).

```
(27.729) @6 INFO:Publishing VOD.mGBPd 21 of 21 on stream1 as UPSERT
_ITEM_NAME_ STRING: VOD.mGBPd
_SERVICE_NAME_ STRING: IDN_RDF
_SEQUENCE_NUMBER_ INTEGER: 1
_ITEM_STALE_ INTEGER: 0
ROW80_1 STRING: VOD.mGBPd SI Quote Publication
ROW80_2 STRING:
```

```
ROW80_3 STRING: DATE:03/07/2008 Time:11:09
ROW80_4 STRING:
ROW80_5 STRING: Time Venue SI Bid Size Bid Price Ask Price Ask Size
Status
ROW80_6 STRING: ==== ===== == ======== ========= ========= ========
======
ROW80_7 STRING: 110937 GSILGB2XXXX GSIL 1 150.9000 150.9500 1 OPEN
ROW80_8 STRING: 070021 SBILGB2LXXX CITI OPEN
ROW80_9 STRING: 110909 CSFBGB2LXXX CSFB 329 150.7000 151.1500 329
OPEN
ROW80_10 STRING: 110942 DEUTGB22ZEQ DBBL 528 150.6500 151.2000 527
OPEN
ROW80_11 STRING: 110946 ABNAGB22XXX ABNV 483306 150.9000 150.9500
483306 OPEN
ROW80_12 STRING: 110936 UBSWGB2LEQU UBSI 1 149.7682 152.1325 1 OPEN
ROW80_13 STRING: 110828 SBUKGB21XXX CITI 20600 150.9000 151.0000
20600 OPEN
ROW80_14 STRING: 110937 SLIIGB2LXXX LEHM 3750 150.9000 150.9500 15
OPEN
ROW80_15 STRING:
ROW80_16 STRING:
ROW80_17 STRING:
(27.730) @6 DEBUG:Immediate flush for low latency; opcode=p
```

Each line of the page has its own FID to facilitate line-oriented deltas to the page. The adapter parses the partial page updates from Reuters and produces strings like the ones shown in the following extract from an adapter log file.

```
(49.934) @6 DEBUG:Processing update for VOD.mGBPd from service
IDN_RDF
(49.934) @6 INFO:Publishing VOD.mGBPd 4 of 21 on stream1 as UPSERT
_ITEM_NAME_ STRING: VOD.mGBPd
_SEQUENCE_NUMBER_ INTEGER: 2
ROW80_3 STRING: off:78 size:2 value:10
ROW80_11 STRING: off:2 size:3 value:101
(49.934) @6 DEBUG:Immediate flush for low latency; opcode=p
(50.315) @6 DEBUG:Processing update for VOD.mGBPd from service
IDN_RDF
(50.315) @6 INFO:Publishing VOD.mGBPd 3 of 21 on stream1 as UPSERT
_ITEM_NAME_ STRING: VOD.mGBPd
_SEQUENCE_NUMBER_ INTEGER: 3
ROW80_11 STRING: off:5 size:1 value:7
(50.315) @6 DEBUG:Immediate flush for low latency; opcode=p
```

The first update in the example is to write the 2-character string 10 at an offset of 78 characters in the line of the page which contains the data from the ROW80_3 FID. The second update in the example is to write the 3-character string 101 at an offset of 2 characters in the line of the page which contains the data from the ROW80_11 FID. The third update in the example is to write the 1-character string 7 at an offset of 5 characters in the line of the page which contains the data from the ROW80_11 FID. Thus, updates for page data are very concise.

Modifying Log Entry Format

You can modify the default format of log entries in two ways.

Set the environment variable ESP_RMDS_SUBSCRIBE_SYMBOL_FORMAT to 1 to configure your system to log messages that show what values flow to the Event Stream Processor on a single line rather than the default multi line format. When messages are written to a log file, this can make it easier to scan for specific items.

Use the -P option to the **esp_rmds** command to specify specify the number of decimal places that appear on output for double type variables.

By default, log messages that show what values flow to the Event Stream Processor are written in multi line format as shown.

```
(38079.526) @2 INFO:Publishing VOD.mGBPd 3 of 9 on stream1 as UPSERT
_ITEM_NAME_ STRING: VOD.mGBPd
_SEQUENCE_NUMBER_ INTEGER: 953
ROW80_7 STRING: off:53 size:2 value:45
```

If you set the environment variable ESP_RMDS_SUBSCRIBE_SYMBOL_FORMAT to 1 these messages are written are written in single-line format.

```
(17.794) @5 DEBUG:stream1 p values: _ITEM_NAME_=VOD.mGBPd
_SEQUENCE_NUMBER_=2
ROW 80_3=off:78 size:2 value:20
```

The -P option can alter the manner in which `double` datatype variables appear, as shown by ask and last are in the following example. This affects only the way variables appear; it does not alter the contents.

```
<RowDefinition id="marketfeed_RowDef">
<Column name="symbol" datatype="string" />
<Column name="service" datatype="string" />
<Column name="seq" datatype="integer" />
<Column name="stale" datatype="integer" />
<Column name="bid" datatype="money" />
<Column name="ask" datatype="double" />
<Column name="last" datatype="double" />
<Column name="volume" datatype="integer" />
<Column name="when" datatype="timestamp" />
</RowDefinition>
```

If you accept the default precision, variables of type `double` (for example, ASK in the following example) are written with three digits to the right of the decimal

```
(5.089) @5 INFO:Publishing EURJPY= 7 of 9 on stream1 as UPSERT
(5.090) @5 DEBUG:stream1 p values: _ITEM_NAME_=EURJPY=
_SEQUENCE_NUMBER_=1 _ITEM_STALE_=0 BID=137.4800 ASK=137.530
ACVOL_1=0
ACTIV_DATE+TIMACT=2008-10-06T21:07:00.000 (1223327220000)
```

If you specify the option -P 7 when enter the **esp_rmds** command, variables of type `double` (for example, ASK in the following example) are written with seven digits to the right of the decimal. Variables of other types are not affected.

```
(4.913) @5 INFO:Publishing EURJPY= 7 of 9 on stream1 as UPSERT
(4.913) @5 DEBUG:stream1 p values: _ITEM_NAME_=EURJPY=
_SEQUENCE_NUMBER_=1 _ITEM_STALE_=0 BID=137.5200 ASK=137.5700000
ACVOL_1=0 ACTIV_DATE+TIMACT=2008-10-06T20:55:00.000 (1223326500000)
```

Reuters Logging

Turn Reuters logging on or off using the Reuters-side configuration file.

You can configure the adapter's interface to RMDS to write to a logging facility. In the Reuters-side configuration file (`rfasub.cfg` and `rfapub.cfg` are the ones provided with the adapter), you can turn logging on or off and specify a path and file name of the log file. The Reuters interface also supports a set of "message files."

The Reuters-side configuration file contains a set of configuration entries for the Reuters "Logger" facility.

```
\Logger\AppLogger\fileLoggerEnabled                = true
\Logger\AppLogger\fileLoggerFilename              = "rfasub.{p}.log"
```

This configuration turns on Reuters logging for the Reuters Marketfeed adapter. The log messages are written to the `rfasub.`**PID**`.log` file, where **PID** is the adapter's process ID.

The first line in this set, `\Logger\AppLogger\windowsLoggerEnabled = false`, pertains to a Windows logging facility that is not supported for the Reuters Marketfeed adapter.

These example lines are from `rfasub.cfg`, the file that configures an adapter that subscribes to RMDS. The configuration file for publication, `rfapub.cfg`, contains the same configuration lines (except that the value for **fileLoggerFilename** is rfapub.{p}.log.

The same file contains configuration entries for Component Loggers, as follows:

```
\Logger\ComponentLoggers\Connections\messageFile   ="config/
messages/RFA7_Connections.mc"
\Logger\ComponentLoggers\Adapter\messageFile       ="config/
messages/RFA7_Adapter.mc"
\Logger\ComponentLoggers\SessionCore\messageFile   ="config/
messages/RFA7_SessionLayer.mc"
\Logger\ComponentLoggers\SSLED_Adapter\messageFile ="config/
messages/RFA7_SSLED_Adapter.mc"
```

Log Messages

Examples of typical entries from the adapter log file for the Reuters Marketfeed adapter.

The actual format and working of the log messages, as well as the nature of the events logged and the log levels associated with these events, may change in subsequent releases of the adapter.

- **Message: –** `NOTICE:Item BARC.VX is closed: No Quality of Service is available to process subscription, timeout expired`
- **Cause: –** the value for the Reuters user name in the Reuters config file is incorrect (verify the case-sensitivity) or the Reuters Service name in the map file is incorrect.
- **Message: –** `DEBUG: Immediate flush for low latency`

- **Cause: –** data received from RMDS is being sent to Event Stream Processor immediately.
- **Message: –** `NOTICE:XMLRPC ERROR-116: The connection to the server could not be established. Please make sure the server is up, and check the specified host name/port, user name/password, and encryption settings. If a host name is specified, make sure that it can be resolved through a DNS lookup. (5.092) @1 INFO:Could not connect to SP; (tigris: 12190 cimtest) will retry in 5 seconds.`
- **Cause: –** cannot connect to the server running Event Stream Processor.
- **Message: –** `Ignoring market data event because no significant fields updated`
- **Cause: –** the adapter received data from Reuters, but none of the fields were of interest to Event Stream Processor stream, so no data was sent.
- **Message: –** `ERROR: Error publishing: PUBLICATION ERROR-442: The send method of this publication object failed.`
- **Cause: –** connection to Event Stream Processor unsuccessful during a message transmission.
- **Message: –** `ERROR:Mismatch between SAP Sybase Event Stream Processor stream (9 columns) and adapter (31 columns for stream: stream1)`
- **Cause: –** the number of columns defined in the adapter did not match the number of columns in the stream.
- **Message: –** `WARNING: Event Stream Processor down, dropping all subscriptions`

  followed by multiple iterations of a message similar to:

  `DEBUG: Unsubscribing item: EUR= service: IDN_RDF`
- **Cause: –** lost connection to Event Stream Processor. Stopping subscriptions to RMDS data since the adapter has nowhere to put it.
- **Message: –** `WARNING: Discarding data rec'd after unsubscribe`
- **Cause: –** before the adapter shut off the subscription, additional data arrived. The data has been discarded because there is no connection to Event Stream Processor.
- **Message: –** `DEBUG: Processing update for EUR= from service IDN_RDF`
- **Cause: –** an update for RIC "EUR=" on service named "IDN_RDF" has arrived.
- **Message: –** `WARNING: Event Stream Processor down, dropping all subscriptions`

  followed by numerous repetitions of:

```
DEBUG: Unsubscribing item: EUR= service: IDN_RDF
```

- **Cause: –** lost connection to Event Stream Processor. Stopping subscriptions to RMDS data since the adapter has nowhere to put it.
- **Message: –** `WARNING: Discarding data rec'd after unsubscribe`
- **Cause: –** before the adapter shut off the subscription, additional data arrived. The data has been discarded, because there is no connection to Event Stream Processor.
- **Message: –** `EMERGENCY: Fatal Error at line 0, column 0 of config file: An exception occurred! Type:RuntimeException, Message:The primary document entity could not be opened. Id=/home/sybase/adapter/trunk/src/ReutersAdapter/xxsubexample.xml`
- **Cause: –** specified configuration file is unavailable.
- **Message: –** `EMERGENCY: Fatal Error at line 0, column 0 of config file: An exception occurred! Type:RuntimeException, Message:The primary document entity could not be opened. Id=/home/sybase/adapter/trunk/src/ReutersAdapter/xxsubexample.xml`
- **Cause: –** specified config file is unavailable.

Reuters OMM Adapter

The SAP Sybase Event Stream Processor Reuters OMM adapter is a software interface between Event Stream Processor and the Reuters Market Data System (RMDS). It uses the Reuters Open Message Model (OMM) message format.

You can configure the adapter as an input or output adapter. The input adapter subscribes to one or more Reuters Instrument Codes (RICs) on the RMDS to provide input to Event Stream Processor. The output adapter publishes output from Event Stream Processor to the RMDS. This enables Event Stream Processor to use the speed and reliability of Reuters' infrastructure to deliver data.

The Reuters OMM Input adapter supports schema discovery. Run two adapter instances if you require both input and output capabilities.

The adapter runs only on Solaris and Linux operating systems but you can use it with Event Stream Processor software running on Solaris, Linux, or Windows.

See also

- *Adapter Support for Schema Discovery* on page 1005

Requirements

The Reuters OMM input and output adapters have several requirements.

An input adapter requires:

- An RMDS market data connection that uses the Reuters Open Message Model (OMM) protocol
- A working subscription for data on one or more financial instruments

An output adapter requires:

- A working connection with support for sending data to RMDS using the OMM protocol

General Configuration

Enable user access for each user account that runs the Reuters OMM adapter, and configure an input connection from Reuters and an output connection to Reuters.

Enabling User Access

Enable user access for each user account that uses the Reuters OMM adapter.

1. Ensure the user account has permission to execute the installed software.
2. Create an environment variable, $ESP_RMDSOMM_HOME, and set to the full path name of the directory in which you placed the adapter distribution file.
3. (Optional) Add the environment variable to your shell profile.
4. Event Stream Processor supports RSA, Kerberos, and LDAP authentication. If your installation uses one of these authentication methods, ensure the user account is set up to work with that method of authentication.

Configuring an Input Connection from Reuters

Modify the sample configuration file for your site's RMDS connection. If you have multiple adapters using multiple RMDS connections, you may need a separate and uniquely named configuration file for each one. For a configuration file with a different name, either change the entry in the input adapter map file or specify that file name using the -f option to the **esp_rmdsomm** command.

Prerequisites

- Create (or choose) a directory in which to store your site-specific configuration files.
- Create an environment variable (MY_CONFIG) and set it to the full path name of that directory.

Task

During the installation process, a sample configuration file (`rmdsomm.cfg`) was placed in the `$ESP_RMDSOMM_HOME/config` directory. This file follows the Reuters format for configuration files and includes this section for your site-specific information:

```
##
## Site-specific values for OMM Inbound - subscribing from RMDS
##
\Connections\Connection_RSSL\connectionType = "RSSL"
### Caution: post value comments like below confuse RFA parsing
causing coredump
#\Connections\Connection_RSSL\hostName       = "localhost" ## not
here
\Connections\Connection_RSSL\hostName       = "localhost"
\Connections\Connection_RSSL\rsslPort       = "14002"
\Connections\Connection_RSSL\connectRetryInterval = 7000
\Sessions\Session1\connectionList           = "Connection_RSSL"
```

1. Obtain this information from your system administrator:
   - Name of the server from which you receive RMDS OMM data
   - Port number on that machine to which your system connects
   - Name of the Reuters service to which you subscribe
2. Make a copy of the sample configuration file in your `$MY_CONFIG` directory.
   ```
   cp $ESP_RMDSOMM_HOME/config/rmdsomm.cfg $MY_CONFIG
   ```
3. Use a text editor to open the configuration file.
4. In the `\Connections\Connection_RSSL\rsslPort` line, replace the default port number (14002) with the port used by your Reuters connection, if different.
5. In the `\Connections\Connection_RSSL\hostName` line, replace tigris.mycompany.com with the name of your server that receives OMM data from RMDS (keep the surrounding quotation marks).

   If your system has more than one server receiving data from RMDS, include all of their names in a comma-separated list, in priority order.
6. (Optional) In the `\Logger\AppLogger\fileLoggerFilename` line, change the name of the log file.

   The default file name `rfasub.{p}.log`, includes the string {p} which the Reuters library replaces with the UNIX process ID when it creates the log file.
7. Save the modified file.

   The other parameters in the configuration file also affect the functioning of the Reuters OMM adapter, and you may want to modify them as well.

Configuring an Output Connection to Reuters

Modify the sample configuration file for your site's RMDS connection. If you have multiple adapters using multiple RMDS connections, you may need a separate and uniquely named configuration file for each one. For a configuration file with a different name, either change the

entry in the output adapter map file or specify that file name using the -F option to the **esp_rmdsomm** command.

Prerequisites

- Create (or choose) a directory in which to store your site-specific configuration files.
- Create an environment variable (MY_CONFIG) and set it to the full path name of that directory.

Task

During the installation process a sample configuration file, `rmdsomm.cfg`, was placed in the `$ESP_RMDSOMM_HOME/config` directory. This file follows the Reuters format for configuration files, and includes sections for site-specific information for noninteractive and interactive publishing to RMDS.

1. Obtain this information from your system administrator:
   - Port number at which the src_dist or RMDS infrastructure server listens for updates from the Reuters OMM adapter
   - Name of the server that receives updates from Event Stream Processor
2. Decide whether to publish to RMDS interactively or non-interactively.
3. If you have not already done so when specifying an input connection from Reuters, make a copy of the sample configuration file in your `$MY_CONFIG` directory.

   ```
   cp $ESP_RMDSOMM_HOME/config/rmdsomm.cfg $MY_CONFIG
   ```
4. Use a text editor to open the configuration file.
   a) If you are going to publish to RMDS interactively, go to the site-specific information section for interactive publishing. In the `\Connections\Connection_RSSL_PROV\connectionType` line, refer to the value "RSSL_PROV," which is the Reuters term for an information provider.

   ```
   ##
   ## Site-specific values for OMM Outbound - Interactive
   publishing to RMDS
   ##
   # Interactive publisher
   \Connections\Connection_RSSL_PROV\connectionType = "RSSL_PROV"
   ## grab a free port until the MDH is setup with 2nd src_dist
   instance
   \Connections\Connection_RSSL_PROV\rsslPort = "14007"
   \Connections\Connection_RSSL_PROV\connectRetryInterval = 7000
   \Connections\Connection_RSSL_PROV\hostName =
   "myhost.mycompany.com"
   \Sessions\SessionOMMProv\connectionList =
   "Connection_RSSL_PROV"
   ```

In the `\Connections\Connection_RSSL_PROV\rsslPort` line, replace the default port number (14007) with the port number at which your IPC server listens for updates from the Reuters OMM adapter, if different.

b) If you are going to publish to RMDS non-interactively, go to the site-specific information section for noninteractive publishing. In the `\Connections\Connection_RSSL_CPROV\connectionType` line, refer to the value "RSSL_CPROV," which is the Reuters term for a client provider.

```
##
## Site-specific values for OMM Outbound - Non-interactive
publishing to RMDS
##
# non-interactive publisher
\Connections\Connection_RSSL_CPROV\connectionType =
"RSSL_CPROV"
\Connections\Connection_RSSL_CPROV\hostName =
"myhost.mycompany.com"
\Connections\Connection_RSSL_CPROV\rsslPort = "14010"
\Connections\Connection_RSSL_CPROV\connectRetryInterval = 7000
\Sessions\SessionOMMCProv\connectionList =
"Connection_RSSL_CPROV"
```

In the `\Connections\Connection_RSSL_CPROV\rsslPort` line, replace the default port number (14010) with the port number at which your IPC server listens for updates from the Reuters OMM adapter, if different.

5. To change the name of the log file, go to the local file logging section.

```
##
## General values
##
## local file logging
\Logger\AppLogger\windowsLoggerEnabled = false
\Logger\AppLogger\fileLoggerEnabled = true
\Logger\AppLogger\fileLoggerFilename = "rfa.{p}.log"
```

In the `\Logger\AppLogger\fileLoggerFilename` line, replace the default name, `rfapub.{p}.log`, with the name you want to use. The Reuters library replaces the {p} string in the default file name with the UNIX Process ID when it creates the log file.

6. Save the modified file.

Enabling Kerberos Authentication for the Reuters OMM Adapter

Enable Kerberos authentication for the Reuters OMM input and output adapters by setting the necessary environment variables and specifying the -G option.

Prerequisites

Ensure you have a 64-bit version of Kerberos installed in the machine.

Task

1. Set the following environment variables:
   a) Set the ESP_SERVICE_NAME environment variable to set the service principal name.
   b) Set the ESP_GSSAPI_LIB environment variable to point to the shared library provided by the Kerberos install. The library contains the GSSAPI function implementations.

      Note: If using a Kerberos library that depends on additional libraries, set the PATH environment variable for Windows or the LD_LIBRARY_PATH environment variable for Solaris and Linux.
   c) Set the KRB5CCNAME environment variable to point to the ticket cache.
   d) Set the KRB5_CONFIG environment variable to point to the configuration file used by the Kerberos library.
2. Specify the -G option.

Input Adapter Configuration

Configure an input adapter to push data from the Reuters Market Data Service (RMDS) to Event Stream Processor.

Before configuring an input adapter, decide what data you need and how you want to set up your system.

You need to know the following about the Event Stream Processor instance from which you receive data.

- Possible security options in a cluster environment, and the workspace and project name.
- What type of authentication mechanism (Kerberos, RSA, LDAP, or Native OS (user name/password) does it use?

Data Decisions

Decide how the incoming Reuters data fits into the project.

Also decide whether you require Level 1 or Level 2 data. For Level 2 data, use the OMM Adapter, and for Level 1 data, you can use either OMM MarketPrice messages or the Reuters Marketfeed adapter.

Decision	Description
Venues	Decide which venues are of interest (for example, NYSE, NASDAQ, Toronto, and so on).

Decision	Description
RICs and FIDs	Determine what market data you need. Specifically, which Reuters Instrument Codes (RICs) you want the adapter to provide to Event Stream Processor, and which Reuters Field IDs (FIDs) for these instruments you want to use.
Streams	The Reuters adapter can furnish data to one or more streams on Event Stream Processor. To use the Reuters Market Data provided by the adapter, decide which existing data streams to map to the adapter's data feed or define one or more new streams.

Administrative Decisions

You have several administrative decisions to make in regards to the project.

Decision	Description
Session Name	An arbitrary string used to link the project and the adapter map file. Use the session name consistently. The adapter supports only one session per adapter instance.
Directories for logging and stream output	The adapter writes its own log messages and can generate a separate set of Reuters log messages. In the configuration, specify whether and where to write these log files.
SAP user account	Specify a valid Event Stream Processor user account for the adapter to use.

Input Adapter Map File

The map file configures the interface between the Reuters OMM adapter and Event Stream Processor. It specifies which source streams receive data from RMDS via the adapter, and it maps specific RMDS Field Identifiers (FIDs) to specific columns in that source stream.

The input adapter map file must accomplish two major tasks:

- Match incoming data elements to columns in one or more streams defined in the Event Stream Processor configuration file.
- Ensure that each update from the adapter can be converted into a record that provides a unique key for each stream being populated, as defined by the stream's column definitions.

Data Structures

Data structures have three important structural aspects: data columns, datatypes, and key values.

- Each data stream includes one or more data columns.
- Each column has a datatype.

- Each row has a unique key value. The source stream definition designates one or more columns as "key" columns. Data must be fed to a source stream.

Incoming RMDS Data

When the adapter subscribes to RMDS for a certain RIC, RMDS first sends an initial image containing all available market data for that RIC. After that, RMDS sends an update only when any values for a subscribed RIC change.

Each FID defined for RMDS has a datatype.

Market Data Field Mapping

Map each column in the target Event Stream Processor stream to a Reuters FID or a "pseudofield."

Find the appropriate FID for each column in the stream. The datatype of the Event Stream Processor column must be compatible with the datatype of the Reuters FID that feeds it.

Here are possible matches between FID datatypes and Event Stream Processor datatypes:

Event Stream Processor Datatype	Reuters Datatype
`integer`	`enumeration`, `time_seconds`
`integer` or `long`	`uint32`
`long`	`uint64`
`money`, `float`, `integer`, or `long`	`real32`
`money` or `float`	`real64`
`string`	`ASCII_string`, `RMTES_string`
`date` or `timestamp`	`date, time`

Note: OMM supports milliseconds as part of a time field. When mapping to or from a timestamp column, milliseconds are preserved.

Reuters Instrument Code Mapping

The identifier of each incoming RMDS update is the Reuters Instrument Code (RIC).

Map the RIC to a column of datatype `string` in the stream. If the stream you want to map to does not have a suitable column, either add a column to the stream or map to a different stream.

Matching the Stream's Key

The adapter map file must configure the adapter so that every update sent to the Event Stream Processor stream includes a field or combination of fields that conform to the unique key defined for that stream. To make this more flexible, the adapter configuration mechanism supports "pseudofields."

The market data updates that the adapter receives from RMDS are mapped to columns in the Event Stream Processor stream using the dataField or dateTimeField element in the map file. RMDS also provides nonmarket data information and each update includes a RIC. Additionally, you can configure the adapter to add a sequence number to each update.

To make these data items available to the mapping process, the map file mechanism supports these elements called "pseudofields."

Field	Description
`dataField`	Type: Datatype is determined at runtime from FIDs and stream schema. (Required) Values such as PRICE, SIZE
`dateTimeField`	Type: `string` (Optional) The date and time.
`itemName`	Type: `string` (Required) The RIC.
`imageField`	Type: `integer` (Required for Level 2 data) Flag to indicate if an entry is an image.
`itemStale`	Type: `integer` (Optional) The item state.
`marketByOrderKeyField`	Type: `integer` (Required for Level 2 MARKET_ BY_ORDER messages) Secondary key for Level 2 messages.
`marketByPriceKeyField`	Type: `integer` (Required for Level 2 MARKET_ BY_PRICE messages) Secondary key for Level 2 messages.

Field	Description
`marketMakerKeyField`	Type: `integer` (Required for MARKET_MAKER messages) Secondary key for Level 2 messages.
`nullField`	(optional) A null value.
`respTypeNumField`	Type: `integer` (Optional) Identifies type of message.
`sequenceNumber`	Type: `long` (Optional) A unique number, assigned sequentially by the adapter to each incoming event whether it causes an update or not.
`serviceName`	Type: `string` (Optional) The name of the service from which RMDS received the market data from this RIC.
`updateNumber`	Type: `long` (Optional) A unique number, assigned sequentially by the adapter to each incoming update.

Getting Stream Information from the Project

Gather the necessary information about the Reuters stream.

The first step in configuring the input adapter is to determine the source streams on Event Stream Processor that will receive the RMDS Market Data. If the Event Stream Processor project does not already include one or more streams for this purpose, define a new stream (or streams) for use with the Reuters adapter.

After you have chosen (or defined) the streams that will receive data from the Reuters OMM adapter, collect information about that stream from your project file. The Event Stream Processor project file contains one or more stream definitions. Each stream definition specifies a data stream that is instantiated when Event Stream Processor is started. The stream definition comprises:

- A unique ID for the stream
- A database store and output file for the stream data
- A list of the columns used as the unique key value for each row in the data stream

Once you have decided which streams will carry the RMDS data provided by the Reuters adapter, get information from the stream definition in the project file. There is no standard for project file names. Two Event Stream Processor installations may have completely different stream definitions, but the definition of any stream includes the same basic set of components.

These instructions refer to the example project to show what components of the stream configuration you must identify to configure the Reuters OMM adapter.

1. Open the project to which the adapter provides data. The Reuters OMM adapter distribution includes an example project, `$ESP_RMDSOMM_HOME/examples/example.ccl`, that contains schema definitions for three streams.
2. Find the name of the source stream. The opening SourceStream tag specifies the name of the stream as the value of the id attribute. The first source stream in this example is named "marketByOrderStream."

   The stream used for subscription by the Reuters OMM adapter must always be a source stream.
3. Determine the key fields. Check each column entry between the opening and closing SourceStream tags to see if the key attribute is set to true. In this example, "marketByOrderStream" has one key field: symbol.
4. Carefully note the number and order of the column entries in the source stream definition.

   In the input adapter map file, list the same set of data in the same order.

Creating the Input Map File

Use the procedure in the sample adapter map files provided in the `examples` subdirectory to create your own adapter map file.

1. Select or create a directory for your adapter map file.
2. Copy the contents of the `$ESP_RMDSOMM_HOME/examples` directory to that directory.
3. Use a text editor to modify the example files as necessary for your installation.

Running the Input Adapter

Run the Reuters OMM input adapter once you have configured it.

Prerequisites

Configure an adapter.

Task

1. Ensure that **esp_server** is running and that the project has been loaded and started.
2. If the Event Stream Processor is running with RSA authentication, start the adapter using:

   ```
   esp_rmdsomm -a in -f mapfile -p cluster_host:cluster_port/
   workspace/project \
    -k <private_rsa_key_file> -c username
   ```
3. If Event Stream Processor is running with LDAP, or Native OS (user name/password) authentication, start the adapter using:

```
esp_rmdsomm -a in -f mapfile -p cluster_host:cluster_port/
workspace/project \
-c username:password
```

4. The adapter starts the subscription by first connecting to Event Stream Processor and then connecting to RMDS. Both connections must be operational for any data to flow.

   If you plan to direct the adapter's log output to stderr, as shown here, you may want to redirect stderr to a log file (for example, append >& myrmdsommlog & to the command line shown above).

Testing the Adapter

If the adapter is not working as expected, you can perform a quick sanity check by executing the **esp_rmdsomm** command and verifying whether the adapter is sending Reuters market data to Event Stream Processor.

- Execute **esp_rmdsomm**:

  ```
  esp_rmdsomm -v
  ```

  This command returns the version information. Ensure that the Event Stream Processor to which you are connecting is compatible with your version of the adapter.
- There are three quick ways to verify that the Reuters OMM adapter is sending Reuters market data to Event Stream Processor:
  - Use the SAP Sybase Event Stream Processor Studio or the **esp_subscribe** command to check the output of the stream configured to receive Reuters data.
  - Use the tail command on the redirected adapter log file (specified in the adapter map file) or the Reuters subscriber log (specified in the configuration file `rmdsomm.cfg`) for activity.
  - Run the **esp_rmdsomm** command with the -d7 option to produce verbose output.

Multiple RICs

When configuring an input adapter, specify multiple RICs if you are interested in more than one symbol.

There are several ways to do this:

- Specify each individual RIC by entering the name directly into the map file or use an XML ENTITY include file.
- Create a dynamic watch list, which employs Event Stream Processor to specify the list of RICs.
- Use a combination of the options above.

Individual RICs

Enter an item element declaration for each RIC you want in the itemList section of the map file.

Here is an example of this:

```
<itemLists service="SSL_PUB" stream="marketByOrderStream">
<itemList>
<item name="CSCO.O"/>
<item name="K.N"/>
<item name="KBN.N"/>
<item name="KBR.N"/>
<item name="ACAM.ARC"/>
<item name="IBM.ARC"/>
</itemList>
</itemLists>
```

It can become difficult to create and maintain your list of RICs this way if it is very large or changes frequently, for example, if you are attempting to track all of the stocks traded on the NYSE. All RICs for the same stream must use the same FID set. Since FIDs often vary by venue, use a different itemList and streamMap for each venue.

Creating a Dynamic Watch List

Creating a dynamic watch list is a bit more complex than creating individual RICs, but is also more flexible. You can specify a custom list of RICs.

Prerequisites

Define source stream (named MyInfoStream) to receive the data, and manually edit the list of RICs.

Task

This method is also dynamic: when inserts or deletes occur on the stream configured using these steps, RMDS subscriptions to the appropriate RICs are started or stopped.

1. Define a stream on Event Stream Processor (for example, MyListStream) which publishes to the adapter the list of RICs to which you wish to subscribe. This stream requires these columns:

Column	Description
symbol	Specifies an RIC symbol ticker (for example, CSCO.O) to which the adapter should subscribe.
service	Specifies the RMDS service on which to subscribe to obtain data for that RIC.
stream	Specifies the name of the stream (for example, MyInfoStream) on which the adapter publishes data for this RIC.

   The stream can also include an optional fourth column, rfaQueue.
2. Define a second stream on Event Stream Processor (for example, MyInfoStream) that receives the data requested by the first stream.

3. Edit the map file to include the subscription.

```
<subscriptions>
<subscription name="subscription1" flags="BASE" >
<stream name="MyListStream" >
<name column="3" /> <!-- symbol -->
<field column="1" name="service"/>
<field column="2" name="stream"/>
</stream>
</subscription>
</subscriptions>
```

4. Specify the set of RICs you want and send them to the first stream you created (for example, MyListStream) to subscribe to them.
   a) Create a file with the same six columns that the stream expects in comma-separated values (CSV) format. The columns are: stream from which you are receiving data, opcode (the `p` in the example is for UPSERT), service, symbol, and stream to which you are sending data.

      For example, use a text editor to open a new file (`RIClist.csv`) and put in these lines:

```
MyListStream,p,,IDN_RDF,MyInfoStream,CSCO.O
MyListStream,p,,IDN_RDF,MyInfoStream,K.N
MyListStream,p,,IDN_RDF,MyInfoStream,KBN.N
MyListStream,p,,IDN_RDF,MyInfoStream,KBN.R
MyListStream,p,,IDN_RDF,MyInfoStream,ACAM.ARC
MyListStream,p,,IDN_RDF,MyInfoStream,IBM.ARC
```

   b) Send the data from the file to Event Stream Processor using the **esp_convert** and **esp_upload** commands. The following example assumes that you have installed all SAP command line tools in the default directories and added those directories to your PATH variable. If you have not, prepend the appropriate path to each command shown in this example.

      For example, to send the file created in the previous step to Event Stream Processor running on port 11180 of your local server, enter:

```
cat RIClist.csv | esp_convert -c user:password -d "," \
-p localhost:11180/ws1/p1 | esp_upload -c user:password -p
localhost:11180/ws1/p1
```

   c) Start the adapter:

```
esp_rmdsomm -f mapfile -d7 -c user:password \
-p localhost:11180/ws1/p1 >& logfile &
```

      If the adapter and Event Stream Processor are on different machines, enter the name of the remote host rather than localhost after the -p in the previous command.

Performance Tuning

There are several attributes you can use to fine-tune performance in an input adapter.

Attribute	Description
`flushInterval`	Specify an interval of time in microseconds (for example, 5000 microseconds = 5 milliseconds) to wait while accumulating data. At the end of this interval, any accumulated events are sent to Event Stream Processor. Send events less often to allow more events to be placed into a message resulting in a communications overhead savings. Use a nonzero flushInterval to make even accumulation time-based.
`maxRecordsPerBlock`	Specify the maximum number of accumulated events that the adapter should send to Event Stream Processor at a time. When the number of accumulated events is larger than this value, the envelope or transaction is broken into fragments that are less than or equal to the specified value. For example, if accumulated event counts of more than 1024 (which would immediately fill the Event Stream Processor Gateway's inbound queue) are expected, set maxRecordsPerBlock to a value like 500 to prevent the inbound queue from filling.
`pendingLimit`	Specify a threshold for the number of events that must accumulate before they are sent to Event Stream Processor. Set this parameter to zero to publish each event immediately when it happens (providing the lowest latency), at the expense of high network overhead (a TCP/IP packet for each update). If you set this parameter to a larger value, the adapter waits until that number of events have accumulated, packs them efficiently in TCP/IP packets, and sends them to Event Stream Processor. This saves communication work but increases latency on both the adapter and Event Stream Processor.

Attribute	Description
`sendAsTransactions`	This parameter controls whether events are sent as an envelope or a transaction. You can specify this parameter on a per-stream basis. Set this parameter to true for Event Stream Processor to treat a group of updates as a single transaction. Transactions typically cause application-level workload savings, since Event Stream Processor collapses multiple updates to the same value in a transaction to a single update. If a transaction contains a delete, additional savings are achieved since updates prior to the delete can be discarded. If you set this parameter to false and you are not in low-latency mode (pendingLimit and flushInterval both set to zero), then use the maxRecordsPerBlock to control the size of the envelope. You still gain the communications overhead savings mentioned above, but not the transactional savings. This is the preferred configuration for applications that require every event to be sent separately , such as a market data compliance application. As a general rule, for quote-based applications, where only the most recent update matters, use transactions to be most efficient. For trades, however, where every event must be processed separately to compute a total volume, use envelopes instead.

When you use both flushInterval and pendingLimit, no event waits longer than the time indicated in the flushInterval before being sent, and as long as the number of events specified in pendingLimit arrive, they are sent immediately. The adapter waits for the amount of time specified in the flushInterval and, if any events have accumulated, it sends them. If the number of pendingLimit events, or more accumulate while the adapter is sending the earlier events, the new events are sent immediately (without waiting for the flushInterval). If fewer than the number of pendingLimit events accumulate while the adapter is sending events, it waits for the flushInterval to elapse.

You can also use the rfaQueue attribute at the itemLists, itemList, or item element level. When specified, the rfaQueue attribute causes the element to be subscribed from Reuters on a named rfaQueue. Each rfaQueue is processed by its own thread within the Reuters adapter. Spreading requests across multiple threads can reduce latency and improve overall adapter throughput at the cost of greater CPU usage.

Since all images and updates come from Reuters on the same queue, the integrity of the order of arrival is maintained for any individual RIC. If you do not specify an rfaQueue for any of the elements, a single default queue (named "defaultQueue") is used for all RICs.

Output Adapter Configuration

Configure an output adapter to push data from Event Stream Processor to RMDS.

Before configuring an output adapter, decide which data to provide and how you want to set up your system.

You need to know the following about the Event Stream Processor instance from which you receive data.

- Possible security options in a cluster environment, and the workspace and project name.
- What type of authentication mechanism (Kerberos, RSA, LDAP, or Native OS (user name/password) does it use?

Data Decisions

Identify which columns from which streams in Event Stream Processor to publish data from.

The Reuters OMM adapter can rearrange the columns from a stream in any order. Its output can also include constants, and the published output can include values from more than one stream.

Consider these items when planning the output of the Reuters OMM Output adapter:

- For each stream for which to publish data, you must specify a unique key in the output adapter map file. Since this adapter sends data to RMDS, the unique identifier should be an RIC. For Market Price data the key can be just the RIC. For Level 2 data, the key must contain additional fields: MarketbyPrice requires PRICE and SIDE, and MarketbyOrder requires ORDER_ID, in addition to the RIC.
- Each data column you want to publish from any stream must map to a unique FID.
- Data from one column can be repeated in the published output, giving you a way to publish a DateTime value as separate Date and Time values.
- If the stream you are working with receives data about the same FID from more than one service, you can configure the adapter to differentiate these data items by service and transmit each service's data separately.
- The first time the Reuters OMM adapter publishes to RMDS, it publishes values for all the columns for which it is configured. After that initial image, the adapter only publishes updates for individual columns as these updates occur.

The datatype of the Event Stream Processor column must be compatible with the Reuters FID datatype that feeds it. This table shows possible matches between Event Stream Processor and FID datatypes:

Event Stream Processor Datatype	Reuters Datatype
`integer`	`enumeration`,`time_seconds`,`uint32`, `uint64`, or `real32`

Event Stream Processor Datatype	Reuters Datatype
`long`	`int64` or `uint64`
`money, float`	`real32` or `real64`
`string`	`ASCII_string`, `RMTES_string`
`date, timestamp`	`date`, `time`

Administrative Decisions

You have several administrative decisions to make in regards to the project.

Decision	Description
Session Name	An arbitrary string used to link the project and the adapter map file. Use it consistently.
Directories for logging and stream output	The adapter writes its own log messages and can generate a separate set of Reuters log messages. In the configuration, specify if and where these log files should be written.
SAP user account	Specify a valid Event Stream Processor user account for the adapter to use.

Reuters Information

You need several pieces of information from Reuters to enable the Reuters OMM adapter to publish to the RMDS.

- The name of the Reuters service on which the adapter transmits data
- Up-to-date lists of valid Reuters Instrument Codes (RICs) and Field Identifiers (FID) used by RMDS
- The Product Permission Code assigned by Reuters

The adapter does not work with the Reuters Data Access Control System (DACS), so the Product Permission Code is needed to allow access to the information you are transmitting on the RMDS.

A list of FIDs, `$ESP_RMDSOMM_HOME/config/RDMFieldDictionary`, has been supplied as part of the Reuters adapter distribution. You can obtain the latest list and other information from your Reuters technical contact.

Getting Stream Information from the Project

Gather the necessary information from the project.

The first step in configuring the output adapter is determining which data elements from which streams on the Event Stream Processor are to be published. After you have chosen (or defined)

a project containing the items for publication over RMDS via the Reuters adapter, collect information from the streams from which to obtain the data to send to RMDS.

Each stream definition specifies a data stream that is instantiated when Event Stream Processor is started up. The stream definition:

- Specifies a unique ID for the stream
- Identifies the columns used as the unique key value for each row in the data stream

Once you have decided which streams will provide the information to be sent to RMDS by the Reuters adapter, get information from the stream definition in the project file. There is no standard for project file names. Two Event Stream Processor installations may have completely different stream definitions, but the definition of any stream includes the same basic set of components.

1. Open the project from which the adapter is obtaining data. The Reuters OMM adapter distribution includes an example project in the `$ESP_RMDSOMM_HOME/examples/example.ccl` file.
2. From the definition of each stream defined in the project:
   a) Obtain the name of the stream from the id attribute in the opening tag of that stream.
   b) Verify that the key attribute is set to true for the column containing the RIC and note the column. In this example, the “marketByOrderStream” has the RIC in the column named “symbol,” which is identified as a key field.
   c) Decide what data, if any, you want the adapter to send to RMDS.
3. Carefully note which streams contain data you want to send to RMDS, and where in the stream definition it is located.

   In the output adapter map file, reference each of the columns you want to publish.

Creating the Output Map File

Create an adapter map file to configure the interface between the output adapter and Event Stream Processor.

There are sample adapter map files in the examples subdirectory.

1. Select or create a directory for your adapter map file.
2. Copy the contents of the `$ESP_RMDSOMM_HOME/examples` directory to that directory.
3. Use the text editor to modify the example files as necessary for your installation.

Running the Output Adapter

Run the adapter once you have configured it.

Prerequisites

Configure an adapter.

Task

1. Ensure that **esp_server** is running and that the project has been loaded and started.
2. Start the adapter:

   ```
   esp_rmdsomm -a out -f mapfile -p cluster_host:cluster_port/
   workspace/project
   ```

   The exact usage of the command depends on how you started your Event Stream Processor. You must invoke the adapter with compatible options. The command string shown invokes neither encryption nor authentication: you can specify either or both.

   Note: If you plan to direct the adapter's log output to stderr, as shown here, you may want to redirect stderr to a log file (for example, append **>& myrmdsommlog &** to the command line shown above).

Testing the Adapter

If the adapter is not working as expected, you can perform a quick sanity check by executing the **esp_rmdsomm** command and verifying whether the adapter is sending Reuters market data to Event Stream Processor.

- Execute **esp_rmdsomm**:

  ```
  esp_rmdsomm -v
  ```
- This command returns the adapter release number and the revision number of the source tree separated by an underscore character. Ensure that your version of the adapter is compatible with your version of Event Stream Processor.
- There are several ways to verify that the Reuters OMM adapter is publishing to RMDS:
  - Use the **tail** command on the adapter log file to which console output was redirected or any of the Reuters publisher log files (specified in `rmdsomm.cfg`) to look for activity.
  - Use the **esp_subscribe** command to look at the outbound stream and verify that values are changing.
  - Use RMDS tools to subscribe to RICs provided by the output adapter.
  - Use an input adapter to subscribe to the output adapter via the RMDS Market Data Hub (MDH).

Performance Tuning

You can improve the performance of output adapters by using multiple threads.

The subscriptions section of the output adapter map file can contain more than one subscription. Each subscription is instantiated on a separate thread so you can specify multiple subscription sections to gain the performance advantage of running on multiple threads.

Split Adapter Map Files

It can be advantageous to put part of your input or output adapter map file in a separate file.

For example, you might want to keep a subscription configuration in a map file, but break out the list of RICs you want the adapter to subscribe to.

The sample files in `$ESP_RMDSOMM_HOME/examples` demonstrate how this facilitates reuse. The `pubexample.omm.map.xml` map file references three "map fragment" files: `mbo.s.mf.xml`, `mbp.s.mf.xml`, and `mp.s.mf.xml` file. The `mbo.s.mf.xml` is also referenced by three other map files.

Map file fragments are reusable blocks of XML for constructing adapter map files, using the XML entity mechanism. File names are of the form `description.parent_element.mf.xml`. Some current descriptions are:

- **mbo –** MarketByOrder
- **mbp –** MarketByPrice
- **mp –** MarketPrice

And current parent_elements are:

- **sd –** Event Stream Processor model stream definition
- **sms –** Subscriber's streamMaps section
- **rfa –** common config section
- **sm –** Subscriber's streamMap
- **il –** Subscriber's itemList
- **s –** Publisher's stream

Therefore, it is evident that the `mbo.sm.mf.xml` file is a subscriber map fragment containing streamMap elements for a MARKET_BY_ORDER message.

Creating a Subordinate Map File

Create a subordinate map file to hold part of the map file configuration.

1. Go to the directory that contains the map File.
2. Create a new file with the extension `.xml`.

   It is not necessary to add a declaration of the XML version.
3. Insert the selected content from the map file into the new file.

   The content you add depends on which part of the map file you have decided to store separately.
4. (Optional) Add a comment to the new file.
5. Save the file when you are done.

Modifying the Main Map File

Modify the main map file to reference the subordinate file.

1. Make sure the first line of the main map file is:
   ```
   <?xml version="1.0"?>
   ```
2. Between the XML version declaration and the opening adapter tag, add these lines:
   ```
   <!DOCTYPE adapter SYSTEM "adapter.dtd" [
   ]>
   ```
3. For each subordinate map file:
   a) Between the two lines just added, add:
      ```
      <!ENTITY SUBREF SYSTEM "SUBFILE">
      ```
      where SUBREF is a string to reference the subordinate file and SUBFILE is the path and filename of the subordinate file itself. Enclose the path and filename in quotation marks.
   b) Remove the content that you put in the subordinate map file.
   c) Insert a string like the following to include the content from the subordinate map file:
      ```
      &SUBREF;
      ```
      where SUBREF is the string you specified to reference the subordinate file.

Command Usage

Usage information for the **esp_ommsample** and **esp_rmdsomm** utilities.

esp_ommsample

The **esp_ommsample** utility displays data received from the Reuters Market Data System (RMDS) to stdout.

Synopsis

```
esp_ommsample -u username [ OPTION ...]
```

Description

The **esp_ommsample** utility operates as a data sink from RMDS for OMM messages. It enables you to see which fields are delivered and their values without setting up a Reuters OMM adapter and model.

esp_ommsample prints data to stdout, getting its configuration from the command line. You can run it for a specified period of time or stop it using Ctrl+C.

Required Arguments

- **-u username –** specify the user name with which to connect to RMDS [ENTER_VALID_USERNAME].

Options

- **-a FID_dictionary –** specify a dictionary to use instead of the default dictionary (`./config/RDMFieldDictionary`).
- **-A applicationId –** specify an ApplicationId to override the default (256).
- **-c Reuters config file –** specify the path and file name of the Reuters configuration file. Since this file is usually shared with the Reuters OMM adapter, this is, by default, set to `./config/rmdsomm.cfg`.
- **-e enum_defs –** specify a file name to override the default (`./config/enumtype.def`).
- **-f format –** specify the format (0, 1, 2, or 3) for update messages. The default, 0, is a multiline format with each value on a separate line. Specify 1 to get all of the values on one line, for example:

  ```
  207 TRIN.O|TRDPRC_1=1.14|BID=1.13|ASK=1.17|ACVOL_1=1000|
  ASK_TIME=10:26:2|
  ```

  The RIC (TRIN.O) is prefaced by a millisecond timestamp and followed by FID=value pairs, delimited by "|". Specify 2 to have the FID numbers included along with the field names: field[FID]=value. Specify 3 for the tersest format: FID=value.

  You can use separator characters for environment variables.
  ESP_OMMSAMPLE_PAIR_SEPARATOR defaults to '='.
  ESP_OMMSAMPLE_FIELD_SEPARATOR defaults to '|'.
  ESP_OMMSAMPLE_TIMESTAMP_SEPARATOR defaults to ' '.
- **-h –** print this help message and exit.
- **-I instanceId –** specify an InstanceId to override the default (1).
- **-m type –** specify the message type (MMT) to use, where type is one of:

  ```
    1 = MarketPrice
    m = MarketMaker
    o = MarketByOrder
  [ p = MarketByPrice ]
    s = SymbolList
  ```
- **-p period –** specify the length of time (in seconds) to listen to updates before terminating. The default is 120.
- **-P position –** Specify a position to override the default (yourIP/net).
- **-r service –** specify the RMDS service: one that is a valid service name for your site. This value defaults to DF_EAP_LAB1, which is a service available on Reuters test lab.
- **-s symbol [symbol ...] –** specify one or more symbols (RICs) to which to subscribe. A space-separated list likely needs quotes to protect it from the shell.
- **-S file –** specify a file containing symbols (RICs) to which to subscribe. These are added to any RICs that have been specified via the -s option.
- **-v –** Show the version number and exit.

Examples

```
cd $ESP_RMDSOMM_HOME/bin
./esp_ommsample -u myUsername -r MY_SERVICE -m 1 -s GOOG.O >&
esp_ommsample.out &
```

esp_rmdsomm

The Reuters OMM Adapter adapts data from the Reuters Market Data System (RMDS) to the Event Stream Processor and vice versa.

Synopsis

```
esp_rmdsomm -f mapFile -p host:port/workspace/project [ OPTION ...]
```

Description

The **esp_rmdsomm** command can start an adapter as either a data source or sink to or from the Event Stream Processor to or from Reuters Market Data System (RMDS). To both subscribe from and publish data to RMDS, you must run two separate RMDS OMM adapter instances.

The metadata describing the connection has several parts, including a map file, configuration file, and possibly a configuration stream resident on a running instance of the Event Stream Processor.

Reuters OMM has several domains. Currently, only MARKET_PRICE, MARKET_BY_PRICE, and MARKET_ BY_ORDER are fully supported. MARKET_MAKER is supported only for inbound streams. See the Reuters documentation for more information, including what FIDs to expect on the message domains.

The process runs as a daemon, getting its configuration from a map file. It handles SIGHUP; so you can enter `kill -s SIGHUP pid` on Linux or `kill -s HUP pid` on Solaris (where pid is the process ID of the **esp_rmdsomm** daemon, which you can obtain using the **ps** command) to gracefully shut dow the adapter. Using the KILL signal rather than the HUP signal may prevent a complete clean up of system resources.

There are three directories underneath the directory where the adapter is installed containing additional information: `doc, examples, and config`. The `doc` directory contains Reuters README files that describe various configuration options. The `examples` directory contains several example map files that demonstrate many features. The `config` directory contains example RMDS configuration files. Minimally, you must modify the RMDS config file with your site's specific information. Typically, you must also modify the map file to match the Event Stream Processor.

Required Arguments

- **-f mapFile –** specify the map file containing the metadata required to map the market data to/from RMDS.

- **-p hostname:port/workspace/project –** specify the URI to connect to the server (cluster manager). For example, -p localhost:19011/default/prj1 specifies a project called prj1 in the default workspace of an ESP cluster server using port 19011 on your localhost.

Options

- **-a in|out|interactive –** specifies whether the RMDS OMM adapter instance is passing data in to the Event Stream Processor or receiving data passed out from it. Valid values are `in`, `out` and `interactive`. Since the default value is `in`, this option is typically omitted when subscribing to market data.

  For backward compatibility, "subscribe" (in) and "publish" (out) are still allowed, but the options have been deprecated.
- **-c user[:password] –** if you are using an authentication method that requires credentials (such as Kerberos or RSA), this option passes those authentication credentials to Event Stream Processor. If Event Stream Processor successfully authenticates with these credentials, the connection is maintained, otherwise Event Stream Processor immediately closes the connection.
- **-d debugLevel –** sets the debug level. The valid range is 0–7, with 0 being minimal and 7 being verbose. By default it is set to 4.
- **-e –** negotiates encrypted OpenSSL sockets for all communication with the Event Stream Processor, which must be started in encrypted mode when using this option.
- **-F configFile –** specifies the RMDS Configuration file, overriding the configuration file specified in the map file.
- **-g gatewayHost –** explicitly specifies the Event Stream Processor gateway host.
- **-h –** print a short help message describing the syntax of this command.
- **-k privateRsaKeyFile –** perform authentication using the RSA private key file mechanism instead of password authentication. The privateRSAKeyFile must specify the absolute path file name of the private RSA key file. With this option enabled, the user name must be specified with the -c option, but the password is not required. In addition, Event Stream Processor must be started with the -k option.
- **-l 0|1|2|3 –** specify the location to which log messages get sent. Use 0 for no log messages, 1 to send to stderr only (the default), 2 to send to syslog only, and 3 to send to both stderr and syslog.
- **-r resubscribeInterval –** specify how many seconds to wait (default is 300) between attempts to resubscribe to a RIC. (If a subscription to a RIC is marked CLOSED or CLOSEDRECOVER, you must resubscribe to that RIC for data to flow.) To disable resubscription attempts, specify `0` as the value. Periodically resubscribing can compensate for a temporary condition where the source is not ready for subscribers. Each unsuccessful resubscribe attempt generates a failure event which may result in a status update marking the item stale.

- **-s streamName –** specify the stream to be used when running in discovery mode. This option is used by the connector start mechanism and specifies the single stream for which mapped columns have been discovered.
- **-v –** print the version of the RMDS OMM adapter and exit.
- **-w retrySeconds –** specify the number of seconds to wait between retries when connecting to the Event Stream Processor. The default is 5. Specify 0 to try only once.
- **-x optName –** specify various extra settings; use `-x help` to see a list of possible values.
- **-z publishCount –** specify the number of (2x) values to pass to the Event Stream Processor before terminating. By default this is 0, which means never terminate.
- **-Z subscribeCount –** Specify the number of (2x) values to pass to RMDS before terminating. By default this is 0, which means never terminate.

Examples

To start a Reuters OMM input adapter on the machine where you enter the command, using port 1099 and running project proj1 in workspace work02 using the `myMap.xml` map file:

```
esp_rmdsomm -c user:passwd -f myMap.xml -p localhost:1099/work02/
proj1 -d 7 &> omm.in.log &
```

To start a Reuters OMM outbound adapter on a host named loki, using port 2010 and running project proj3 in workspace work01 using the `myMap.xml` map file:

```
esp_rmdsomm -a out -c user:passwd -f myMap.xml -p loki:2010/work01/
proj3 -d 7 &> omm.out.log &
```

Environment Variables

The Reuters OMM adapters use environment variables to specify behavior.

Environment Variable	Used By	Description
ESP_ACCUMULATOR_DELAY	Input	(Expert) Delay connection to the Event Stream Processor (seconds).
ESP_DISABLE_REPORT_ENCOD-ING_NULL	Output	Stop warning about blank-to-null con-versions (bool) [false].
ESP_FLUSH_INTERVAL	Input	Override the publication flushInterval (microseconds).
ESP_INTRASUBSCRIBE_DELAY	Input	Override the map attribute (millisec-onds).
ESP_LOG_CONFIG_EVENTS	Both	Set log level (1–7; 2x) for config event processing [-1]
ESP_MAX_RECORDS_PER_BLOCK	Input	Override the publication maxRecord-sPerBlock (count).

Environment Variable	Used By	Description
ESP_PENDING_LIMIT	Input	Override the publication pendingLimit.
ESP_RETRY_INTERVAL	Both	Override the publication retryInterval.
ESP_RMDSOMM_DISPATCH	Both	(Expert) Dispatch RFA every N milli-seconds [10,000].
ESP_RMDSOMM_EVENT_TRACE	Both	(Expert) Enable RFA event tracing every N event (int).
ESP_RMDSOMM_HOME	Both	Specify the installation directory.
ESP_RMDSOMM_PUBLISH_DE-BUG_LEVEL	Output	Set to 7 to see values [not in -opt].
ESP_RMDSOMM_PUBLISH_DE-BUG_SYMBOLS	Output	Contains a space-delimited list of sym-bols that are used when default behavior is overriden. If this environment variable is not set, all symbols are used.
ESP_RMDSOMM_SUBSCRIBE_DE-BUG_LEVEL	Input	Set to 7 to see values [not in -opt].
ESP_RMDSOMM_SUBSCRIBE_DE-BUG_SYMBOLS	Input	Contains a space-delimited list of sym-bols that are used when default behavior is overriden. If this environment variable is not set, all symbols are used.
ESP_RMDSOMM_SUBSCRIBE_SYM-BOL_FORMAT	Input	Specify symbol list format: 0 for multi-line; 1 for single line.
ESP_SEND_AS_TRANSACTIONS	Input	Override the map attribute.
ESP_SHOW_FIELD_INFO	Input	Show FID, column, spColumn, and stream name [false].
ESP_SHOW_SP_EVENT_DATA	Output	Set log level (1–7) for events from the Event Stream Processor [-1].

Input Adapter Map File

Shows the structure of the map file for the Reuters OMM Input adapter.

```
adapter                                          (required, limit one)
  |----publication                               (required, limit one)
  |----streamMaps                                (required, limit one)
  |      '----streamMap                          (required)
  |             |----dataField                   (required)
```

```
  |              |----hiResTimestampField     (optional)
  |              |----imageField              (required for L2 data)
  |              |----itemName                (required, limit one)
  |              |----itemStale               (optional)
  |              |----marketByOrderKeyField   (required)
  |              |----marketByPriceKeyField   (required)
  |              |----marketMakerKeyField     (required)
  |              |----nullField               (optional)
  |              |----respTypeNumField        (optional)
  |              |----sequenceNumber          (optional)
  |              |----serviceName             (optional)
  |              '----updateNumber            (optional)
  |----rfa                                    (required, limit one)
  '----itemLists                              (required, limit one)
         '----itemList                        (required)
                '----item                     (optional)
```

adapter

The adapter element is the root element of the map file.

Summary

```
adapter                                     (required, limit one)
  |----publication                          (required, limit one)
  |----streamMaps                           (required, limit one)
  |      '----streamMap                     (required)
  |              |----dataField             (required)
  |              |----hiResTimestampField   (optional)
  |              |----imageField            (required for L2 data)
  |              |----itemName              (required, limit one)
  |              |----itemStale             (optional)
  |              |----marketByOrderKeyField (required)
  |              |----marketByPriceKeyField (required)
  |              |----marketMakerKeyField   (required)
  |              |----nullField             (optional)
  |              |----respTypeNumField      (optional)
  |              |----sequenceNumber        (optional)
  |              |----serviceName           (optional)
  |              '----updateNumber          (optional)
  |----rfa                                  (required, limit one)
  '----itemLists                            (required, limit one)
         '----itemList                      (required)
                '----item                   (optional)
```

Parent

None

Children

The following child elements are defined for adapter. All of these elements must be present, and in the order specified.

Name	Requirement
publication	Exactly one required
streamMaps	Exactly one required
rfa	Exactly one required
itemLists	Exactly one required

Attributes

Name	Description	Requirement
name	A string that uniquely identifies this adapter (included in log entries)	Optional

Notes
None

Example
See the examples given for each of the component elements of the map.

dataField
In the streamMap definition, the dataField element maps a Reuters Field ID (FID) to one column in a source stream.

Summary

```
adapter                                     (required, limit one)
  |----publication                          (required, limit one)
  |----streamMaps                           (required, limit one)
  |       '----streamMap                    (required)
  |              |----dataField             (required)
  |              |----hiResTimestampField   (optional)
  |              |----imageField            (required for L2 data)
  |              |----itemName              (required, limit one)
  |              |----itemStale             (optional)
  |              |----marketByOrderKeyField (required)
  |              |----marketByPriceKeyField (required)
  |              |----marketMakerKeyField   (required)
  |              |----nullField             (optional)
  |              |----respTypeNumField      (optional)
  |              |----sequenceNumber        (optional)
  |              |----serviceName           (optional)
  |              '----updateNumber          (optional)
  |----rfa                                  (required, limit one)
  '----itemLists                            (required, limit one)
         '----itemList                      (required)
                '----item                   (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
name	The Reuters FID that identifies the data item that appears in this column of the source stream	Required
key	True or false, depending on whether this column is part of the source stream's unique key	See Notes

Notes
Each element in the **streamMap** section of the input adapter map file must represent a column in the row definition of the target source stream. (The order of the **streamMap** elements must mirror the order of the columns in the source stream.) If the column in the source stream is a data item (Bid, Ask, and so on), the corresponding **streamMap** entry must be a **dataField** element for which the name attribute identifies a specific FID. Any time RMDS publishes an update tagged with that FID, the adapter sends it to Event Stream Processor source stream as a value in the corresponding column.

Use the key attribute to set the value to true. If this column is not part of the stream's key, you can omit the key attribute.

The adapter uses the Event Stream Processor schema.

Example

```
<streamMap name="marketByOrder">
        <itemName key="true"/>
        <!-- serviceName / -->
        <sequenceNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <dataField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

This example maps columns 4–8 of the marketByOrder stream to the Reuters FIDs BID, ASK, TRDPRC_1, and ACVOL_1.

dateTimeField

In a streamMap, the **dateTimeField** element maps a Reuters date or time FID (or one of each) to a date column, a timestamp column, or both, in an Event Stream Processor source stream.

Summary

```
adapter                                      (required, limit one)
  |----publication                           (required, limit one)
  |----streamMaps                            (required, limit one)
  |       '----streamMap                     (required)
  |              |----dataField              (required)
  |              |----hiResTimestampField    (optional)
  |              |----imageField             (required for L2 data)
  |              |----itemName               (required, limit one)
  |              |----itemStale              (optional)
  |              |----marketByOrderKeyField  (required)
  |              |----marketByPriceKeyField  (required)
  |              |----marketMakerKeyField    (required)
  |              |----nullField              (optional)
  |              |----respTypeNumField       (optional)
  |              |----sequenceNumber         (optional)
  |              |----serviceName            (optional)
  |              '----updateNumber           (optional)
  |----rfa                                   (required, limit one)
  '----itemLists                             (required, limit one)
         '----itemList                       (required)
                '----item                    (optional)
```

Parent

streamMap

Children

None

Attributes

Name	Description	Requirement
dateName	The FID of the date value provided by RMDS	See Note
timeName	The FID of the time value provided by RMDS	See Note

Notes

The dateTime datatype, which combines both date and time, is the most commonly used datatype for date/time information in Event Stream Processor data streams. In most cases, however, the updates provided by RMDS and brought in to the Event Stream Processor by the Reuters OMM adapter use separate FIDs for date and time.

To address this discrepancy, the map file provides the **dateTimeField** element, which provides separate attributes for date and time, allowing you to map two FIDs (one for date, one for time) to the same column in the source stream definition.

If dateTime is used, it must be used alone. The dateName and timeName attributes can be used either separately or together. One of these three attributes must be used.

The value for each FID must match one listed in the FID list referenced in the Reuters-side configuration file (the FID list provided with the adapter is named `appendix_a`). This file is referenced in the `rmdsomm.cfg` configuration file.

Example

```
<streamMap name="marketByOrderStream">
        <itemName key="true"/>
        <!-- serviceName / -->
        <sequenceNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <dataField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

This example maps the TIMACT and ACTIV_DATE FIDs together to the ninth column of the Event Stream Processor source stream marketByOrderStream.

hiResTimestampField

The hiResTimestampField element substitutes a high-resolution timestamp for the regular timestamp.

Summary

```
adapter                                     (required, limit one)
  |----publication                          (required, limit one)
  |----streamMaps                           (required, limit one)
  |      '----streamMap                     (required)
  |             |----dataField              (required)
  |             |----hiResTimestampField    (optional)
  |             |----imageField             (required for L2 data)
  |             |----itemName               (required, limit one)
  |             |----itemStale              (optional)
  |             |----marketByOrderKeyField  (required)
  |             |----marketByPriceKeyField  (required)
  |             |----marketMakerKeyField    (required)
  |             |----nullField              (optional)
  |             |----respTypeNumField       (optional)
  |             |----sequenceNumber         (optional)
  |             |----serviceName            (optional)
  |             '----updateNumber           (optional)
  |----rfa                                  (required, limit one)
  '----itemLists                            (required, limit one)
```

```
        '----itemList                          (required)
               '----item                       (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
name	long relative timestamp	required

Notes
This element can be used only on Solaris machines.

Example

```
<streamMap name="marketByOrderStream">
        <itemName key="true"/>
        <!-- serviceName / -->
        <hiResTimestampField name="TIME"/>
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <dataField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

imageField

The **imageField** element indicates whether or not the Event Stream Processor row is part of a snapshot initial image.

Summary

```
adapter                                    (required, limit one)
  |----publication                         (required, limit one)
  |----streamMaps                          (required, limit one)
  |      '----streamMap                    (required)
  |             |----dataField             (required)
  |             |----hiResTimestampField   (optional)
  |             |----imageField            (required for L2 data)
  |             |----itemName              (required, limit one)
  |             |----itemStale             (optional)
  |             |----marketByOrderKeyField (required)
  |             |----marketByPriceKeyField (required)
  |             |----marketMakerKeyField   (required)
  |             |----nullField             (optional)
  |             |----respTypeNumField      (optional)
```

```
  |             |----sequenceNumber          (optional)
  |             |----serviceName             (optional)
  |             '----updateNumber            (optional)
  |----rfa                                   (required, limit one)
  '----itemLists                             (required, limit one)
         '----itemList                       (required)
                '----item                    (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
name	Integer (1 if part of image, 0 if not)	Required for Level 2

Notes
The project should treat all rows of a snapshot image as one transaction.

Example

```
        <name column="0"/>
        <keyField column="1" name="mbpkey" />
        <imageField column="2" />
<!-- summary fields -->
        <stale column="4" />
        <field column="5" name="CURRENCY" />
        <field column="6" name="ACTIV_DATE" />
        <field column="7" name="PROD_PERM" />
<!-- end summary fields -->
```

item
The item element identifies an RIC to which the Reuters OMM adapter subscribes.

Summary

```
adapter                                    (required, limit one)
  |----publication                         (required, limit one)
  |----streamMaps                          (required, limit one)
  |       '----streamMap                   (required)
  |             |----dataField             (required)
  |             |----hiResTimestampField   (optional)
  |             |----imageField            (required for L2 data)
  |             |----itemName              (required, limit one)
  |             |----itemStale             (optional)
  |             |----marketByOrderKeyField (required)
  |             |----marketByPriceKeyField (required)
  |             |----marketMakerKeyField   (required)
```

```
  |             |----nullField                (optional)
  |             |----respTypeNumField         (optional)
  |             |----sequenceNumber           (optional)
  |             |----serviceName              (optional)
  |             '----updateNumber             (optional)
  |----rfa                                    (required, limit one)
  '----itemLists                              (required, limit one)
         '----itemList                        (required)
                '----item                     (optional)
```

Parent
itemList

Children
None

Attributes

Name	Description	Requirement
name	An RIC to which the adapter will subscribe	Required
rfaQueue	A name for the rfaQueue, which, if provided, replaces the default rfaQueue name and cause a separate thread to be used for this queue	Optional
service	The name of a Reuters Service that provides incoming data through RMDS	Optional if already specified in the parent itemList or itemLists element, otherwise required
stream	The source stream on which updates for this RIC are be brought to the Event Stream Processor	Optional if already specified in the parent itemList or itemLists element, otherwise required

Notes
The value for the name attribute should be a valid RIC on the service.

If you specify a stream name here, updates for this RIC are brought in to the Event Stream Processor on that stream. If you do not specify a stream here, the stream specified at the **itemList** level is used.

The stream you specify must match a **streamMap** defined elsewhere in the map file.

Example

```
<itemLists service="SSL_PUB" stream="marketByOrderStream">
        <itemList service="IDN_RDF" >
                <item name="EUR=" />
                <item name="EURJPY=" stream="stream6" />
        </itemList>
</itemLists>
```

These two **item** elements subscribe the adapter to the RICs EUR= and EURJPY=. The EUR= updates are sent to the stream marketByOrderStream which was set in the **itemLists** element. The EURJPY= updates are sent to the stream stream6, since the **item** level stream attribute overrides the **itemLists** level attribute.

itemList

The **itemList** element contains one or more instances of the **item** element.

Summary

```
adapter                                          (required, limit one)
  |----publication                               (required, limit one)
  |----streamMaps                                (required, limit one)
  |       '----streamMap                         (required)
  |              |----dataField                  (required)
  |              |----hiResTimestampField        (optional)
  |              |----imageField                 (required for L2 data)
  |              |----itemName                   (required, limit one)
  |              |----itemStale                  (optional)
  |              |----marketByOrderKeyField      (required)
  |              |----marketByPriceKeyField      (required)
  |              |----marketMakerKeyField        (required)
  |              |----nullField                  (optional)
  |              |----respTypeNumField           (optional)
  |              |----sequenceNumber             (optional)
  |              |----serviceName                (optional)
  |              '----updateNumber               (optional)
  |----rfa                                       (required, limit one)
  '----itemLists                                 (required, limit one)
         '----itemList                           (required)
                '----item                        (optional)
```

Parent

itemLists

Children

Name	Requirement
item	zero or more required

Attributes

Name	Description	Requirement
rfaQueue	A name for the rfaQueue, which if provided, it replaces the default rfaQueue name and causes a separate thread to be used for this queue	optional

Name	Description	Requirement
service	The name of a Reuters Service that provides incoming data through RMDS	optional if already specified in the parent itemLists element or in all child item elements, otherwise required
stream	The name of an Event Stream Processor source stream that will receive updates on the RICs specified in this list of items	optional if already specified in the parent itemLists element or in all child item elements, otherwise required

Notes

Configure the adapter to push updates for every item in this section to that stream (although you can override this specification at the item level) by specifying a stream name for this element.

The adapter supports more than one itemList element under itemLists; this allows you to configure one instance of the adapter to direct updates from two or more groups of RICs to different Event Stream Processor source streams.

The stream you specify must match one of the streamMaps defined elsewhere in the map file (by the value of the streamMap's name attribute).

Use the rfaQueue attribute to control scalability.

Example

```
<itemLists service="SSL_PUB" stream="marketByOrderStream">
        <itemList service="IDN_RDF" >
        <item name="EUR=" />
        <item name="EURJPY=" stream="stream6" />
        </itemList>
</itemLists>
```

This **itemList** element sets the service attribute to IDN_RDF, overriding the SSL_PUB service attribute defined in the parent **itemLists** element.

itemLists

The **itemLists** element contains one or more instances of the **itemList** element.

Summary

```
adapter                                     (required, limit one)
  |----publication                          (required, limit one)
  |----streamMaps                           (required, limit one)
  |      '----streamMap                     (required)
  |             |----dataField              (required)
  |             |----hiResTimestampField    (optional)
  |             |----imageField             (required for L2 data)
```

```
|             |----itemName                 (required, limit one)
|             |----itemStale                (optional)
|             |----marketByOrderKeyField    (required)
|             |----marketByPriceKeyField    (required)
|             |----marketMakerKeyField      (required)
|             |----nullField                (optional)
|             |----respTypeNumField         (optional)
|             |----sequenceNumber           (optional)
|             |----serviceName              (optional)
|             '----updateNumber             (optional)
|----rfa                                    (required, limit one)
'----itemLists                              (required, limit one)
       '----itemList                        (required)
             '----item                      (optional)
```

Parent
adapter

Children

Name	Requirement
itemList	One required, two or more supported

Attributes

Name	Description	Requirement
name	A string that appears in any adapter-related log entries	Optional
rfaQueue	A name for the rfaQueue, which if provided, it replaces the default rfaQueue name and causes a separate thread to be used for this queue (a default that you can override at the item level)	Optional
service	The name of a Reuters Service that provides incoming data through RMDS (a default that you can override at the item level)	Optional if specified in the child itemLists or item elements or both, so that all child item elements either specify or inherit it, otherwise required
stream	The name of an Event Stream Processor source stream that receives updates on the RICs specified in the item lists in this section (a default that you can override at the item level)	Optional if specified in the child itemLists and/or item elements so that all child item elements either specify or inherit it, otherwise required

Notes
Each **itemList** instance in this section is a list of one or more RICs to which the adapter subscribes.

Get the value for service from your Reuters administrator.

Example

```
<itemLists service="SSL_PUB" stream="marketByOrderStream">
        <itemList service="IDN_RDF" >
                <item name="EUR=" />
                <item name="EURJPY=" stream="stream6" />
        </itemList>
</itemLists>
```

This **itemLists** element sets the service attribute to SSL_PUB and the stream attribute to marketByOrderStream. These attributes are either inherited, or overridden at the **itemList** and/or **item** level.

itemName

In the **streamMap** definition, the **itemName** element identifies the row in the Event Stream Processor source stream that carries the RIC from the RMDS update.

Summary

```
adapter                                     (required, limit one)
  |----publication                          (required, limit one)
  |----streamMaps                           (required, limit one)
  |      '----streamMap                     (required)
  |             |----dataField              (required)
  |             |----hiResTimestampField    (optional)
  |             |----imageField             (required for L2 data)
  |             |----itemName               (required, limit one)
  |             |----itemStale              (optional)
  |             |----marketByOrderKeyField  (required)
  |             |----marketByPriceKeyField  (required)
  |             |----marketMakerKeyField    (required)
  |             |----nullField              (optional)
  |             |----respTypeNumField       (optional)
  |             |----sequenceNumber         (optional)
  |             |----serviceName            (optional)
  |             '----updateNumber           (optional)
  |----rfa                                  (required, limit one)
  '----itemLists                            (required, limit one)
         '----itemList                      (required)
                '----item                   (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
key	True or false, depending on whether or not this column is part of the source stream's unique key	See first note

Notes

You need not use the **key** attribute; it is present for backward compatibility.

Insert the **itemName** element in the streamMap to correspond with the column in the source stream that carries the RIC or symbol. If this column is part of the source stream's key, set the **key** attribute to true.

This element is one of the "pseudofields" that specify data items that are not part of the data feed coming directly from RMDS.

Example

```
<streamMap name="marketByOrderStream">
        <itemName key="true"/>
        <!-- serviceName / -->
        <sequenceNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <dataField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

The first column of the source stream is identified as the one that carries the RIC value of any update from the adapter. It is also identified as part of the stream's key.

itemStale

In the **streamMap** definition, the **itemStale** element identifies a column in the Event Stream Processor source stream that carries an indicator of whether or not incoming RMDS data has gone stale.

Summary

```
adapter                                     (required, limit one)
  |----publication                          (required, limit one)
  |----streamMaps                           (required, limit one)
  |      '----streamMap                     (required)
  |             |----dataField              (required)
  |             |----hiResTimestampField    (optional)
  |             |----imageField             (required for L2 data)
  |             |----itemName               (required, limit one)
  |             |----itemStale              (optional)
  |             |----marketByOrderKeyField  (required)
```

```
  |             |----marketByPriceKeyField   (required)
  |             |----marketMakerKeyField     (required)
  |             |----nullField               (optional)
  |             |----respTypeNumField        (optional)
  |             |----sequenceNumber          (optional)
  |             |----serviceName             (optional)
  |             '----updateNumber            (optional)
  |----rfa                                   (required, limit one)
  '----itemLists                             (required, limit one)
         '----itemList                       (required)
                '----item                    (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
name	A string that appears in any adapter-related log entries	optional

Notes
Use this element in the **streamMap** if one of the columns in the source stream is a "stale" column.

RMDS itself does not supply a stale flag with regular market data, although it may pass along such a flag if it is provided by another service you are subscribing to via RMDS. If this element is used in the **streamMap**, the adapter sends a non-zero update value if it receives a stale flag from RMDS, or stops receiving any data from RMDS. It uses three sets of bits to indicate stale reasons.

Adapter Status Bits	Description
0	Unknown = initial state
1	ConnectionInLoss
2	ConnectionOutLoss
3–7	Reserved

Data Status Bits	Description
8	Data suspect
9	Unspecified (initializing)

Data Status Bits	Description
10–15	Reserved

Stream Status Bits	Description
16	Unspecified = initializing
17	NonStreaming = configured as snapshot only
18	ClosedRecover = stream is closed but can be retried
19	Closed = stream is closed and not coming back
20	Redirected = part of failing over state
21	Stale = for OMM
22–24	Reserved

Example

```
<streamMap name="marketByOrderStream">
        <itemName key="true"/>
        <!-- serviceName / -->
        <sequenceNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <dataField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

The fourth column of the source stream is identified as the one that is updated if the adapter receives a "stale" notification or stops receiving data from RMDS.

marketByOrderKeyField

The **marketByOrderKeyField** element is a secondary key for messages of the MARKET_BY_ORDER domain.

Summary

```
adapter                                     (required, limit one)
  |----publication                          (required, limit one)
  |----streamMaps                           (required, limit one)
  |      '----streamMap                     (required)
  |             |----dataField              (required)
  |             |----hiResTimestampField    (optional)
  |             |----imageField             (required for L2 data)
  |             |----itemName               (required, limit one)
  |             |----itemStale              (optional)
  |             |----marketByOrderKeyField  (required)
```

```
  |             |----marketByPriceKeyField   (required)
  |             |----marketMakerKeyField     (required)
  |             |----nullField               (optional)
  |             |----respTypeNumField        (optional)
  |             |----sequenceNumber          (optional)
  |             |----serviceName             (optional)
  |             '----updateNumber            (optional)
  |----rfa                                   (required, limit one)
  '----itemLists                             (required, limit one)
         '----itemList                       (required)
                '----item                    (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
name	string	required for Level 2

Notes
Typically, the ORDER_ID FID is specified as the secondary key.

Example

```
<streamMaps>
    <streamMap name="MarketByOrderStream"
messageType="MARKET_BY_ORDER">
        &marketByOrder;
    </streamMap>
    <streamMap name="MarketByPriceStream"
messageType="MARKET_BY_PRICE">
        &marketByPrice;
    </streamMap>
    <streamMap name="MarketMakerStream" messageType="MARKET_MAKER">
        &marketMaker;
    </streamMap>
</streamMaps>
```

marketByPriceKeyField
The marketByPriceKeyField element is a secondary key for messages of the MARKET_BY_PRICE domain.

Summary

```
adapter                                  (required, limit one)
  |----publication                       (required, limit one)
```

```
  |----streamMaps                              (required, limit one)
  |       '----streamMap                       (required)
  |              |----dataField                (required)
  |              |----hiResTimestampField      (optional)
  |              |----imageField               (required for L2 data)
  |              |----itemName                 (required, limit one)
  |              |----itemStale                (optional)
  |              |----marketByOrderKeyField    (required)
  |              |----marketByPriceKeyField    (required)
  |              |----marketMakerKeyField      (required)
  |              |----nullField                (optional)
  |              |----respTypeNumField         (optional)
  |              |----sequenceNumber           (optional)
  |              |----serviceName              (optional)
  |              '----updateNumber             (optional)
  |----rfa                                     (required, limit one)
  '----itemLists                               (required, limit one)
         '----itemList                         (required)
                '----item                      (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
name	PRICE + SIDE as a string	required for Level 2

Notes
This element is not meant to be parsed by the Event Stream Processor; it is used only as a secondary key to keep orderbook rows for the same RIC.

Example

```
<streamMaps>
    <streamMap name="MarketByOrderStream"
messageType="MARKET_BY_ORDER">
        &marketByOrder;
    </streamMap>
    <streamMap name="MarketByPriceStream"
messageType="MARKET_BY_PRICE">
        &marketByPrice;
    </streamMap>
    <streamMap name="MarketMakerStream" messageType="MARKET_MAKER">
        &marketMaker;
    </streamMap>
</streamMaps>
```

marketMakerKeyField

The marketMakerKeyField element is a secondary key for messages of the MARKET_MAKER domain.

Summary

```
adapter                                      (required, limit one)
  |----publication                           (required, limit one)
  |----streamMaps                            (required, limit one)
  |       '----streamMap                     (required)
  |              |----dataField              (required)
  |              |----hiResTimestampField    (optional)
  |              |----imageField             (required for L2 data)
  |              |----itemName               (required, limit one)
  |              |----itemStale              (optional)
  |              |----marketByOrderKeyField  (required)
  |              |----marketByPriceKeyField  (required)
  |              |----marketMakerKeyField    (required)
  |              |----nullField              (optional)
  |              |----respTypeNumField       (optional)
  |              |----sequenceNumber         (optional)
  |              |----serviceName            (optional)
  |              '----updateNumber           (optional)
  |----rfa                                   (required, limit one)
  '----itemLists                             (required, limit one)
         '----itemList                       (required)
                '----item                    (optional)
```

Parent

streamMap

Children

None

Attributes

Name	Description	Requirement
name	string, typically MMID	required for Level 2

Notes

None

Example

```
<streamMaps>
    <streamMap name="MarketByOrderStream"
messageType="MARKET_BY_ORDER">
        &marketByOrder;
    </streamMap>
```

```
    <streamMap name="MarketByPriceStream"
messageType="MARKET_BY_PRICE">
        &marketByPrice;
    </streamMap>
    <streamMap name="MarketMakerStream" messageType="MARKET_MAKER">
        &marketMaker;
    </streamMap>
</streamMaps>
```

nullField

In a **streamMap**, the **nullField** element acts as a placeholder that always delivers a NULL value to the Event Stream Processor source stream. This lets you add extra fields to a source stream to get the configuration you want.

Summary

```
adapter                                          (required, limit one)
  |----publication                               (required, limit one)
  |----streamMaps                                (required, limit one)
  |      '----streamMap                          (required)
  |             |----dataField                   (required)
  |             |----hiResTimestampField         (optional)
  |             |----imageField                  (required for L2 data)
  |             |----itemName                    (required, limit one)
  |             |----itemStale                   (optional)
  |             |----marketByOrderKeyField       (required)
  |             |----marketByPriceKeyField       (required)
  |             |----marketMakerKeyField         (required)
  |             |----nullField                   (optional)
  |             |----respTypeNumField            (optional)
  |             |----sequenceNumber              (optional)
  |             |----serviceName                 (optional)
  |             '----updateNumber                (optional)
  |----rfa                                       (required, limit one)
  '----itemLists                                 (required, limit one)
         '----itemList                           (required)
                '----item                        (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
name	A string that appears in any adapter-related log entries	optional
dateName	A string that appears in any adapter-related log entries	optional

Name	Description	Requirement
timeName	A string that appears in any adapter-related log entries	optional

Notes

When experimenting with a project, you can replace a **dataField** or **dateTimeField** element with a **nullField** to temporarily stop feeding data into any column of the stream.

You need not modify any attribute(s) of the **dataField** or **dateTimeField** you are temporarily replacing, as the following example shows.

Example

```
<streamMap name="marketByOrderStream">
        <itemName key="true"/>
        <!-- serviceName / -->
        <sequenceNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <nullField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

The sixth column of the source stream is identified as a placeholder that receives a null value in each update from the adapter. It includes the name of the **dataField** that it replaces for debugging purposes.

publication

The **publication** element specifies basic publishing information for this instance of the adapter.

Summary

```
adapter                                     (required, limit one)
  |----publication                          (required, limit one)
  |----streamMaps                           (required, limit one)
  |      '----streamMap                     (required)
  |             |----dataField              (required)
  |             |----hiResTimestampField    (optional)
  |             |----imageField             (required for L2 data)
  |             |----itemName               (required, limit one)
  |             |----itemStale              (optional)
  |             |----marketByOrderKeyField  (required)
  |             |----marketByPriceKeyField  (required)
  |             |----marketMakerKeyField    (required)
  |             |----nullField              (optional)
  |             |----respTypeNumField       (optional)
  |             |----sequenceNumber         (optional)
  |             |----serviceName            (optional)
  |             '----updateNumber           (optional)
```

```
  |----rfa                                        (required, limit one)
  '----itemLists                                  (required, limit one)
         '----itemList                            (required)
                '----item                         (optional)
```

Parent
adapter

Children
None

Attributes

Name	Description	Requirement
flushInterval	Specify the number of microseconds the adapter allows events to accumulate before sending them to the Event Stream Processor. A nonzero flushInterval makes event accumulation time-based.	Optional (the default is 1000)
intraSubscribeDelay	Specify the number of milliseconds the adapter pauses between subscription requests.	Optional (the default is 100)
maxRecordsPerBlock	Specify the maximum number of accumulated events that the adapter sends to the Event Stream Processor at a time. This reduces the size of each transaction or envelope fragment when there is a large number of accumulated events. For example, if 140 events have accumulated and maxRecordsPerBlock is set to 50, the adapter sends the envelope or transaction as three fragments.	Optional (the default is 256)
name	Specify a string that identifies the adapter instance in log file entries.	Optional
pendingLimit	Specify the number of events that may accumulate before the adapter sends them in to the Event Stream Processor. Using a pendingLimit makes the event accumulation count-based.	Optional (the default is 256)
retryInterval	Specify the number of seconds for which the adapter waits between attempts to connect to RMDS before shutting down.	Optional (the default is 5)

Name	Description	Requirement
sendAsTransactions	Set to true to treat a group of updates as a single transaction or false to treat them as separate rows within an envelope.	Optional (the default is false)

Notes

You can optimize the adapter's performance using the pendingLimit and flushInterval attributes, along with the maxRecordsPerBlock and sendAsTransactions attributes from the Pub/Sub interface that the adapter uses to communicate with the Event Stream Processor. See *Performance Tuning* for details.

Some venues send initial images as multipart messages, which may produce large data sets. The intraSubscribeDelay attribute provides the ability to pace these subscriptions and prevents the adapter from being overwhelmed by initial images. The default value is zero, which is suitable for short RIC lists. When intraSubscribeDelay is set to a nonzero value, the adapter pauses between subscription requests for the specified number of milliseconds. The suggested value is ten (10).

Example

```
<publication name="RMDS Adapter - low latency" retryInterval="5"
        flushInterval="0" pendingLimit="0" sendAsTransactions="0" />
```

respTypeNumField

The **respTypeNumField** element populates a column with the RMDS respTypeNum value.

Summary

```
adapter                                     (required, limit one)
  |----publication                          (required, limit one)
  |----streamMaps                           (required, limit one)
  |      '----streamMap                     (required)
  |             |----dataField              (required)
  |             |----hiResTimestampField    (optional)
  |             |----imageField             (required for L2 data)
  |             |----itemName               (required, limit one)
  |             |----itemStale              (optional)
  |             |----marketByOrderKeyField  (required)
  |             |----marketByPriceKeyField  (required)
  |             |----marketMakerKeyField    (required)
  |             |----nullField              (optional)
  |             |----respTypeNumField       (optional)
  |             |----sequenceNumber         (optional)
  |             |----serviceName            (optional)
  |             '----updateNumber           (optional)
  |----rfa                                  (required, limit one)
  '----itemLists                            (required, limit one)
         '----itemList                      (required)
                '----item                   (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
name	A string used in adapter-related log entries	optional

Notes
For an initial snapshot image, **respTypeNumField** has a value of 1 for UNSOLICITED or 0 for SOLICITED. Updates may have other values. See the RMDS documentation for more details.

Example

```
<itemName key="true" /> <!-- str: the RIC -->
<marketByPriceKeyField key="true"/> <!-- str: SIDE + PRICE as a key
-->
<imageField name="imageIn" />
<updateNumber name="upd" /> <!-- generated by Adapter -->
<respTypeNumField name="rtn" />
```

rfa

The **rfa** element links the subscriber map file to the Reuters-side configuration file.

Summary

```
adapter                                     (required, limit one)
  |----publication                          (required, limit one)
  |----streamMaps                           (required, limit one)
  |       '----streamMap                    (required)
  |              |----dataField             (required)
  |              |----hiResTimestampField   (optional)
  |              |----imageField            (required for L2 data)
  |              |----itemName              (required, limit one)
  |              |----itemStale             (optional)
  |              |----marketByOrderKeyField (required)
  |              |----marketByPriceKeyField (required)
  |              |----marketMakerKeyField   (required)
  |              |----nullField             (optional)
  |              |----respTypeNumField      (optional)
  |              |----sequenceNumber        (optional)
  |              |----serviceName           (optional)
  |              '----updateNumber          (optional)
  |----rfa                                  (required, limit one)
  '----itemLists                            (required, limit one)
         '----itemList                      (required)
                '----item                   (optional)
```

Parent
adapter

Children
None

Attributes

Name	Description	Requirement
config	The absolute path and file name of the Reuters-side configuration file for subscription (the sample file supplied with the adapter is at `$ESP_RMDSOMM_HOME/config/rmdsomm.cfg`).	Required
configDatabaseName	Must be set to RFA.	Required
enumFile	The full path name of the Reuters-supplied file that lists each enumerated type along with the range of values it can take	See first Note
fidFile	The full path name of the Reuters-supplied file that lists all of the valid FIDs	See second Note
sessionName	A reference to a session name defined in the Reuters-side configuration file for subscription	Required
blank	Specifies a marker to use for blanks	Optional
blankInt32	Specifies a marker to use for blank `Int32` fields	Optional
blankInt64	Specifies a marker to use for blank `Int64` fields	Optional
blankMoney	Specifies a marker to use for blank `Money` fields	Optional
blankString	Specifies a marker to use for blank String fields	Optional
blankDate	Specifies a marker to use for blank `Date` fields	Optional
blankTimestamp	Specifies a marker to use for blank `Timestamp` fields	Optional
blankDouble	Specifies a marker to use for blank `Double` fields	Optional

Notes
The default enumFile is `$ESP_RMDSOMM_HOME/config/enumtype.def`.

The default fidFile is `$ESP_RMDSOMM_HOME/config/RDMFieldDictionary`.

You can specify another file for either of these defaults.

Example

```
<rfa config="$ESP_RMDSOMM_HOME/config/rmdsomm.cfg"
        sessionName="Session1" />
```

This example points the Reuters OMM adapter to the Reuters-side configuration in the file `rmdsomm.cfg`. The list line in this configuration file is:

```
\Sessions\Session1\connectionList                          =
"Connection_SSLED"
```

This line defines a session name that is referenced by other lines in the configuration file. When the map file references a session name in the sessionName attribute, it links the adapter to the Reuters-side configuration parameters identified by that name.

sequenceNumber

In the **streamMap** definition, the **sequenceNumber** element maps a column in Event Stream Processor source stream that is populated by a unique number generated by the adapter, not provided as part of the data from RMDS.

Summary

```
adapter                                     (required, limit one)
  |----publication                          (required, limit one)
  |----streamMaps                           (required, limit one)
  |      '----streamMap                     (required)
  |             |----dataField              (required)
  |             |----hiResTimestampField    (optional)
  |             |----imageField             (required for L2 data)
  |             |----itemName               (required, limit one)
  |             |----itemStale              (optional)
  |             |----marketByOrderKeyField  (required)
  |             |----marketByPriceKeyField  (required)
  |             |----marketMakerKeyField    (required)
  |             |----nullField              (optional)
  |             |----respTypeNumField       (optional)
  |             |----sequenceNumber         (optional)
  |             |----serviceName            (optional)
  |             '----updateNumber           (optional)
  |----rfa                                  (required, limit one)
  '----itemLists                            (required, limit one)
         '----itemList                      (required)
                '----item                   (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
key	true or false, depending on whether this column is part of the source stream's unique key	See Note
name	a string that appears in log entries	Optional

Notes

The adapter maintains a separate counter for each RIC to which it is subscribed. Each time it receives an update for an RIC, it increments its counter for that RIC. This number is the one sent to the source stream column mapped by the **sequenceNumber** element.

Source stream definitions include a column specification similar to:

```
<Column datatype="long" name="Id"/>
```

This line specifies a unique ID for the source stream. The **sequenceNumber** pseudo-field is a good match for this column in the input adapter map file.

You must use the key attribute to set the value to true. If this column is not part of the stream's key, you can omit this attribute.

Example

```
<streamMap name="marketByOrderStream">
        <itemName key="true"/>
        <!-- serviceName / -->
        <sequenceNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <dataField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

The third column of the source stream is mapped to the sequence number provided by the adapter. This column is also identified as part of the source stream's unique key.

serviceName

In the **streamMap** definition, the **serviceName** element maps a column in the Event Stream Processor source stream to the service identifier that the adapter provides.

Summary

```
adapter                                        (required, limit one)
  |----publication                             (required, limit one)
  |----streamMaps                              (required, limit one)
  |      '----streamMap                        (required)
  |             |----dataField                 (required)
```

```
  |             |----hiResTimestampField     (optional)
  |             |----imageField              (required for L2 data)
  |             |----itemName                (required, limit one)
  |             |----itemStale               (optional)
  |             |----marketByOrderKeyField   (required)
  |             |----marketByPriceKeyField   (required)
  |             |----marketMakerKeyField     (required)
  |             |----nullField               (optional)
  |             |----respTypeNumField        (optional)
  |             |----sequenceNumber          (optional)
  |             |----serviceName             (optional)
  |             '----updateNumber            (optional)
  |----rfa                                   (required, limit one)
  '----itemLists                             (required, limit one)
         '----itemList                       (required)
                '----item                    (optional)
```

The identifier provided by **serviceName** can potentially be used to provide namespace scope for a RIC that was provided by two different services to which you subscribed.

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
key	true or false, depending on whether this column is part of the source stream's unique key	see Notes

Notes
You must use the key attribute to set the value to true. If this column is not part of the stream's key, you can omit this attribute.

Example

```
<streamMap name="marketByOrderStream">
        <itemName key="true"/>
        <!-- serviceName / -->
        <sequenceNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <dataField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

In this example, no column of the source stream is mapped to the service name provided by the adapter because it is commented out.

streamMap

The **streamMap** element of the input map file defines the mappings between the columns of an Event Stream Processor source stream and the RMDS FIDs being subscribed to by the adapter.

Summary

```
adapter                                      (required, limit one)
  |----publication                           (required, limit one)
  |----streamMaps                            (required, limit one)
  |      '----streamMap                      (required)
  |             |----dataField               (required)
  |             |----hiResTimestampField     (optional)
  |             |----imageField              (required for L2 data)
  |             |----itemName                (required, limit one)
  |             |----itemStale               (optional)
  |             |----marketByOrderKeyField   (required)
  |             |----marketByPriceKeyField   (required)
  |             |----marketMakerKeyField     (required)
  |             |----nullField               (optional)
  |             |----respTypeNumField        (optional)
  |             |----sequenceNumber          (optional)
  |             |----serviceName             (optional)
  |             '----updateNumber            (optional)
  |----rfa                                   (required, limit one)
  '----itemLists                             (required, limit one)
         '----itemList                       (required)
                '----item                    (optional)
```

Parent

streamMaps

Children

The following child elements are defined for **streamMap**. These child elements can occur in any order, but for a specific **streamMap**, the order of the child elements must mirror the order of the columns of the source stream (as defined in the project). This is how the adapter is configured to deliver RMDS updates to the appropriate rows in the source stream.

Name	Requirement
dataField	One required, two or more supported
dateTimeField	Zero or more supported
imageField	Required for Level 2 data
itemName	One required, two or more supported
itemStale	Zero or one supported

Name	Requirement
marketByOrderKeyField	Required for Level 2 MARKET_BY_ORDER messages
marketByPriceKeyField	Required for Level 2 MARKET_BY_PRICE messages
marketMakerKeyField	Required for Level 2 MARKET_MAKER messages
nullField	Zero or more supported
respTypeNumField	Zero or more supported
sequenceNumber	Zero or more supported
serviceName	Zero or more supported
updateNumber	Zero or more supported

Attributes

Name	Description	Requirement
name	Identifies the source stream to which the RMDS updates are mapped; must match the name of a source stream defined in the Event Stream Processor project	Required
sendAsTransactions	True to treat a group of updates as a single transaction, or false to treat them as separate rows within an envelope	Optional (the default is false)

Notes

None

Example

```
<streamMaps>
        <streamMap name="marketByOrderStream">
                <itemName key="true"/>
                <!-- serviceName / -->
                <sequenceNumber />
                <itemStale/>
                <dataField name="BID"/>
                <dataField name="ASK"/>
                <dataField name="TRDPRC_1"/>
                <dataField name="ACVOL_1"/>
                <dateTimeField timeName="TIMACT"
dateName="ACTIV_DATE"/>
```

```
        </streamMap>
</streamMaps>
```

This example maps a set of the adapter's updates to an Event Stream Processor source stream named marketByOrderStream. All updates going to this source stream are added using the upsert opcode.

The RICs for which updates are sent to this source stream are specified in an **itemList** elsewhere in the map file that also references marketByOrderStream.

streamMaps

The **streamMaps** element of the input map file contains one or more **streamMap** elements.

Summary

```
adapter                                        (required, limit one)
  |----publication                             (required, limit one)
  |----streamMaps                              (required, limit one)
  |       '----streamMap                       (required)
  |              |----dataField                (required)
  |              |----hiResTimestampField      (optional)
  |              |----imageField               (required for L2 data)
  |              |----itemName                 (required, limit one)
  |              |----itemStale                (optional)
  |              |----marketByOrderKeyField    (required)
  |              |----marketByPriceKeyField    (required)
  |              |----marketMakerKeyField      (required)
  |              |----nullField                (optional)
  |              |----respTypeNumField         (optional)
  |              |----sequenceNumber           (optional)
  |              |----serviceName              (optional)
  |              '----updateNumber             (optional)
  |----rfa                                     (required, limit one)
  '----itemLists                               (required, limit one)
         '----itemList                         (required)
                '----item                      (optional)
```

Parent

adapter

Children

Name	Requirement
streamMap	One required, two or more supported

Attributes

None

Notes

Each **streamMap** instance in this section maps incoming FIDs from the Reuters adapter to columns in an Event Stream Processor source stream.

A stream must have a **streamMap**.

Example

```
<streamMaps>
        <streamMap name="marketByOrderStream">
                <itemName key="true"/>
                <!-- serviceName / -->
                <sequenceNumber />
                <itemStale/>
                <dataField name="BID"/>
                <dataField name="ASK"/>
                <dataField name="TRDPRC_1"/>
                <dataField name="ACVOL_1"/>
                <dateTimeField timeName="TIMACT"
dateName="ACTIV_DATE"/>
        </streamMap>
</streamMaps>
```

updateNumber

In the **streamMap** definition, the **updateNumber** element maps a column in Event Stream Processor source stream that are populated by a unique number generated by the adapter, not provided as part of the data from RMDS.

Summary

```
adapter                                      (required, limit one)
  |----publication                           (required, limit one)
  |----streamMaps                            (required, limit one)
  |      '----streamMap                      (required)
  |             |----dataField               (required)
  |             |----hiResTimestampField     (optional)
  |             |----imageField              (required for L2 data)
  |             |----itemName                (required, limit one)
  |             |----itemStale               (optional)
  |             |----marketByOrderKeyField   (required)
  |             |----marketByPriceKeyField   (required)
  |             |----marketMakerKeyField     (required)
  |             |----nullField               (optional)
  |             |----respTypeNumField        (optional)
  |             |----sequenceNumber          (optional)
  |             |----serviceName             (optional)
  |             '----updateNumber            (optional)
  |----rfa                                   (required, limit one)
  '----itemLists                             (required, limit one)
         '----itemList                       (required)
                '----item                    (optional)
```

Parent
streamMap

Children
None

Attributes

Name	Description	Requirement
key	true or false, depending on whether this column is part of the source stream's unique key	See Notes
name	A string that appears in log entries	Optional

Notes
The adapter infers whether or not a column is part of the stream's unique key from the schema; the key attribute is included here only for backward compatibility.

The adapter maintains a separate counter for each RIC to which it is subscribed. Each time it receives an update for a RIC, it increments its counter for that RIC. This number is sent to the column in the stream mapped by the **updateNumber** element.

Many source stream definitions include a column specification similar to:

```
<Column datatype="int64" name="Id"/>
```

This line specifies a unique ID for the source stream. The **updateNumber** pseudo-field is a good match for this column in the input adapter map file.

Example

```
<streamMap name="marketByOrderStream">
        <itemName key="true"/>
        <updateNumber />
        <itemStale/>
        <dataField name="BID"/>
        <dataField name="ASK"/>
        <dataField name="TRDPRC_1"/>
        <dataField name="ACVOL_1"/>
        <dateTimeField timeName="TIMACT" dateName="ACTIV_DATE"/>
</streamMap>
```

In this example, the second column of the source stream is mapped to the update number provided by the adapter. This column is also identified as part of the source stream's unique key. To see additional examples, look in the `$ESP_RMDSOMM_HOME/examples` directory.

Output Adapter Map File XML Syntax

The syntax of the map file for a Reuters OMM output adapter.

```
adapter                          (required, limit one)
  |----rfa                                 (required, limit one)
  '----subscriptions                       (required, limit one)
         '----subscription                 (required)
                '----stream                (required)
                       |----name           (required, limit one)
                       |----stale          (optional)
                       |----field          (required)
                       '----constant       (optional)
```

adapter

The **adapter** element is the root element of the map file.

Summary

```
adapter                          (required, limit one)
  |----rfa                                 (required, limit one)
  '----subscriptions                       (required, limit one)
         '----subscription                 (required)
                '----stream                (required)
                       |----name           (required, limit one)
                       |----stale          (optional)
                       |----field          (required)
                       '----constant       (optional)
```

Nest all configuration elements between the start and end **adapter** tags.

Parent

None

Children

The following child elements are defined for **adapter**. All of these elements must be present in the specified order.

Name	Requirement
rfa	Exactly one required
subscriptions	Exactly one required

Attributes

None

Notes

None

Example
See the examples for the child elements.

constant
The **constant** element defines a data item with a constant value that will be published to RMDS by the adapter.

Summary

```
adapter                          (required, limit one)
  |----rfa                                 (required, limit one)
  '----subscriptions                       (required, limit one)
         '----subscription                 (required)
                '----stream                (required)
                       |----name           (required, limit one)
                       |----stale          (optional)
                       |----field          (required)
                       '----constant       (optional)
```

Parent
stream

Children
None

Attributes

Name	Description	Requirement
name	The name associated with this data item in the image published by the adapter	Required
value	The value of this constant (always the same whenever this data item is published to RMDS)	Required

Notes
On start-up, the adapter publishes a complete image to RMDS, containing all data items defined in the map file. After that, the adapter publishes updated values for data items only when they change, unless Event Stream Processor goes stale and then recovers. This means that the value for **constant** is published only when a complete image is published.

Example

```
<stream name="stream1" >
        <name column="0"/>
        <stale column="3" name="ACVOL_1"/>
        <field column="1" name="DSPLY_NAME" />
        <field column="4" name="BID" precision="47" />
        <field column="5" name="ASK" precision="0" />
```

```
        <field column="6" name="TRDPRC_1"/>
        <field column="7" name="ACVOL_1"/>
        <constant name="PROD_PERM" value="1"/>
</stream>
```

This example defines a constant called PROD_PERM, with the constant value 1, to be published with data values from stream1 under the publication name subscription1.

field

In a **stream** definition in an output adapter map file, **field** specifies a column in a stream to publish.

Summary

```
adapter                          (required, limit one)
  |----rfa                                 (required, limit one)
  '----subscriptions                       (required, limit one)
         '----subscription                 (required)
                '----stream                (required)
                       |----name           (required, limit one)
                       |----stale          (optional)
                       |----field          (required)
                       '----constant       (optional)
```

Parent
stream

Children
None

Attributes

Name	Description	Requirement
column	A number that represents the position of the source column in the stream being published from (the first column in the stream has the number 0)	Either column or columnName is required
columnName	The name of the column in the Event Stream Processor stream that carries the stream's unique identifier	Either column or columnName is required
name	The FID that identifies this data value when published to RMDS	Required
precision	An integer that specifies the total number of digits after the decimal point in the published value (for example, 1.23 has a precision of 2)	Optional

Notes
The precision attribute should be included only for columns of datatype `double`.

Example

```
<stream name="stream1" >
        <name column="0"/>
        <stale column="3" name="ACVOL_1"/>
        <field column="1" name="DSPLY_NAME" />
        <field column="4" name="BID" precision="47" />
        <field column="5" name="ASK" precision="0" />
        <field column="6" name="TRDPRC_1"/>
        <field column="7" name="ACVOL_1"/>
        <constant name="PROD_PERM" value="1"/>
</stream>
```

The adapter is configured to publish updates from the fourth, fifth, sixth and seventh columns of the Event Stream Processor stream named stream1 as data items named BID, ASK, TRDPRC_1, and ACVOL_1, respectively.

name

In a **stream** definition in an output adapter map file, **name** specifies the column in the source stream that provides the value to use to identify each update.

Summary

```
adapter                          (required, limit one)
  |----rfa                                 (required, limit one)
  '----subscriptions                       (required, limit one)
         '----subscription                 (required)
                '----stream                (required)
                       |----name           (required, limit one)
                       |----stale          (optional)
                       |----field          (required)
                       '----constant       (optional)
```

Parent
stream

Children
None

Attributes

Name	Description	Requirement
column	A number that represents the position of the column in the stream that carries the stream's unique identifier (the first column in the stream is number 0)	Either column or columnName
colum-nName	The name of the column in the stream that carries the stream's unique identifier	Either column or columnName

Notes

The output adapter uses RMDS as a simple message bus; published updates need not conform to Reuters protocols. This means that the column specified by this element does not have to be a Reuters RIC, but it must follow Reuters RIC syntax.

If the source stream's unique key is a composition of two or more columns, you can use the name element in combination with one or more instances of the service element to configure the adapter to publish updates with completely unique names.

Example

```
<stream name="stream1" >
        <name column="0"/>
        <stale column="3" name="ACVOL_1"/>
        <field column="1" name="DSPLY_NAME" />
        <field column="4" name="BID" precision="47" />
        <field column="5" name="ASK" precision="0" />
        <field column="6" name="TRDPRC_1"/>
        <field column="7" name="ACVOL_1"/>
        <constant name="PROD_PERM" value="1"/>
</stream>
```

This example identifies the first column of stream1 as its unique identifier or "key" column.

rfa

The **rfa** element provides information for configuring the Reuters side of the adapter, including an explicit reference to the Reuters-side configuration file.

Summary

```
adapter                         (required, limit one)
  |----rfa                                (required, limit one)
  '----subscriptions                      (required, limit one)
         '----subscription                (required)
                '----stream               (required)
                       |----name          (required, limit one)
                       |----stale         (optional)
                       |----field         (required)
                       '----constant      (optional)
```

Parent

adapter

Children

None

Attributes

Name	Description	Requirement
serviceName	A service name that is included in the header of every update sent out by the Reuters OMM adapter	Optional
config	The absolute path and file name of the Reuters-side configuration file for publication (the sample file supplied with the adapter is at `$ESP_RMDSOMM_HOME/config/rmdsomm.cfg`)	Required
sessionName	A reference to a session named defined in the Reuters-side configuration file for publication	Required
configDatabaseName	A reference to the Reuters database name	Optional
blankDate	A marker to use for blank `Date` fields	Optional
blankDouble	A marker to use for blank `Double` fields	Optional
blankInt32	A marker to use for blank `Int32` fields	Optional
blankInt64	A marker to use for blank `Int64` fields	Optional
blankMoney	A marker to use for blank `Money` fields	Optional
blankString	A marker to use for blank `String` fields	Optional
blankTimestamp	A marker to use for blank `Timestamp` fields	Optional
enumFile	The full path name of the Reuters-supplied file that lists each enumerated type along with the range of values it can take	Optional (the default is the enumType definition)
fidFile	The full path name of the Reuters-supplied file that lists all of the valid FIDs	optional (the default is the RDMField Dictionary)

Notes

None

Example

```
<rfa serviceName="IDN_RDF"
        config="$ESP_RDMSOMM_HOME/config/rmdsomm.cfg"
        sessionName="Session1" configDatabaseName="RFA" />
```

This example points the Reuters OMM adapter to the Reuters-side configuration in the file `rmdsomm.cfg key`. The first five uncommented lines in this configuration file are:

```
\Connections\Connection_RSSL\connectionType = "RSSL"
\Connections\Connection_RSSL\hostName = "tigris.mycompany.com"
\Connections\Connection_RSSL\rsslPort = "14002"
\Connections\Connection_RSSL\connectRetryInterval = 7000
\Sessions\Session1\connectionList = "Connection_RSSL"
```

The last of these lines implicitly defines a session name that is defined as the **sessionName** in the map file. The other three lines from `rmdsomm.cfg` key on this session name. This is how the value for **sessionName** ties this publication section of the map file to a configuration set in the `.cfg` file.

When the adapter publishes using this configuration, each update is identified with the **serviceName** "IDN_RDF."

stale

In a **stream** definition in an output adapter map file, the **stale** element identifies a column in the source stream for which the value changes from 0 to 1 if the stream goes stale.

Summary

```
adapter                          (required, limit one)
  |----rfa                                 (required, limit one)
  '----subscriptions                       (required, limit one)
         '----subscription                 (required)
                '----stream                (required)
                       |----name           (required, limit one)
                       |----stale          (optional)
                       |----field          (required)
                       '----constant       (optional)
```

A stream is considered to have gone stale if, for example, one of the stream's data sources is no longer being updated.

Parent

stream

Children

None

Attributes

Name	Description	Requirement
column	A number that represents the position of the column with the secondary key value (the first column in the stream has the number 0)	Required

Name	Description	Requirement
name	A string that identifies the stale column so that it may be mapped to a FID (published)	Optional

Notes
None

Example

```
<stream name="stream1" >
        <name column="0"/>
        <stale column="3" name="ACVOL_1"/>
        <field column="1" name="DSPLY_NAME" />
        <field column="4" name="BID" precision="47" />
        <field column="5" name="ASK" precision="0" />
        <field column="6" name="TRDPRC_1"/>
        <field column="7" name="ACVOL_1"/>
        <constant name="PROD_PERM" value="1"/>
</stream>
```

This example identifies the third column of stream1 as its stale column. If the stale column is specified, the column value is published and the RIC is marked stale.

stream
In a subscription section in an output adapter map file, identifies the stream from which the adapter gets the data it publishes to RMDS.

Summary

```
adapter                          (required, limit one)
  |----rfa                                 (required, limit one)
  '----subscriptions                       (required, limit one)
         '----subscription                 (required)
                '----stream                (required)
                       |----name           (required, limit one)
                       |----stale          (optional)
                       |----field          (required)
                       '----constant       (optional)
```

Parent
subscription

Children

Name	Requirement
name	One
stale	Optional

Name	Requirement
field	One or more
constant	Optional

Attributes

Name	Description	Requirement
exitOnStreamExit	This is a boolean attribute. When true, esp_rmdsomm terminates if the stream exits, Event Stream Processor exits, or the connection is lost.	Optional (default is false)
finalizer	This string specifies an action to take if the specified number of heartbeat milliseconds elapse without an event being published to Event Stream Processor.	Optional
heartbeat	This integer specifies how many milliseconds to wait without an event being published to Event Stream Processor before executing the finalizer action .	Optional
name	The name of the stream from which the adapter receives the data it publishes on RMDS	Required
ESPExitOnStreamDrop	This is a boolean attribute. When true, Event Stream Processor exits if this subscription drops its connection.	Optional (default is false)
ESPQueueSize	The size, in bytes, of Event Stream Processor queue	Optional (default is 8000)

Notes

The value of the **name** attribute must be defined in Event Stream Processor project.

Any stream in Event Stream Processor project can map to only one **stream** section in the map file.

Example

```
<stream name="stream1">
        <name column="0"/>
        <field column="4" name="TRDPRC_1"/>
        <field column="9" name="BID" precision="5"/>
</stream>
```

This example configures Event Stream Processor to publish data from a stream named stream1.

subscription

The **subscription** element contains one or more instances of the **stream** element; enabling you to configure the adapter to receive data from one or more streams.

Summary

```
adapter                          (required, limit one)
  |----rfa                                 (required, limit one)
  '----subscriptions                       (required, limit one)
         '----subscription                 (required)
                '----stream                (required)
                       |----name           (required, limit one)
                       |----stale          (optional)
                       |----field          (required)
                       '----constant       (optional)
```

The output adapter map file can contain two or more **subscription** sections. At runtime, the publishing mechanism for each **subscription** section is instantiated on a separate thread, which provides scalability.

Parent

subscriptions

Children

Name	Requirement
stream	One or more

Attributes

Name	Description	Requirement
name	A name for this subscription that appears in updates published on RMDS and in log file entries	Required

Notes

None

Example

```
<subscriptions>
    <subscription name="subscription1" >
        <stream name="stream1" >
            <name column="0"/>
            <field column="4" name="BID"/>
            <field column="5" name="ASK"/>
            <field column="6" name="TRDPRC_1"/>
```

```
            <field column="7" name="ACVOL_1"/>
            <constant name="PROD_PERM" value="1"/>
        </stream>
    </subscription>
</subscriptions>
```

This example configures the adapter to publish some columns from stream1 on Event Stream Processor using the name subscription1.

subscriptions

The **subscriptions** element contains one or more **subscription** elements.

Summary

```
adapter                          (required, limit one)
  |----rfa                                 (required, limit one)
  '----subscriptions                       (required, limit one)
         '----subscription                 (required)
                '----stream                (required)
                       |----name           (required, limit one)
                       |----stale          (optional)
                       |----field          (required)
                       '----constant       (optional)
```

Parent

adapter

Children

Name	Requirement
subscription	One or more

Attributes

None

Notes

Each **subscription** instance in this section defines one set of data that the adapter publishes to RMDS.

Example

See the example for an individual **subscription** instance.

Logging Facilities

The Reuters OMM adapter supports two different logging mechanisms.

In addition to its own logging mechanism, the Reuters OMM adapter can utilize Reuters-side logging. You can use both of these mechanisms to check the adapter's performance and diagnose problems.

You can configure these logs to be written to stderr, syslog, or both.

Adapter Logging

The Reuters OMM adapter supports the same options for logging as the Event Stream Processor.

The **-d** option sets the debug level (0=emergency messages only, 7=all messages).

The **-l** option tells the adapter to write log messages to stderr, syslog, both, or neither. If you use the **-l** option to direct adapter log messages to stderr, you may also want to redirect stderr to a file.

The name attribute of the **publication** element in the input adapter map file specifies a descriptive text string that is logged to help identify how the adapter was configured. For example, lines 3–6 of `subexample.xml` specify the **publication** element for a subscribing instance of the Reuters OMM adapter, as follows:

```
<publication
    name="RMDS OMM Adapter"
    retryInterval="5"
/>
```

As the adapter connects with and interacts with Event Stream Processor, this configuration causes the adapter to write log messages similar to:

```
(0.123) @1 INFO: Configuring publication with name RMDS Adapter exp
```

The first two fields are the timestamp (in seconds since start-up) and the thread number, respectively. The base time for the timestamp, along with other information, is written to the log file on startup as shown in the example below. To convert the timestamp to a date and time, simply add the number of seconds to the base time.

```
(63359098041.768) @1 NOTICE:Base time is 10/08/08-17:27:21
(0.001) @1 NOTICE:insta-a sub -c cimtest:-- -d 7
-f /home/sybase/support/1.0.3/ReutersOMMAdapter/marketprice.map.xml
-l 1 -p tigris:12192 -P 1
(0.001) @1 NOTICE:pid=28649
(0.001) @1 DEBUG:Using ESP_RMDSOMM_SUBSCRIBE_DEBUG_LEVEL=7ll/
i86pc_64_spro/bin/rmdsomm version:
1.0.3a-alpha_r18674M
```

Page Data and Partial Page Updates

Some Reuters data comes as pages which use the partial page format. Each page consists of multiple lines; initially sent as a snapshot. Page data is supported without any special configuration. The following extract from an adapter log file shows the delivery of the initial page image (which is highlighted).

```
(27.729) @6 INFO:Publishing VOD.mGBPd 21 of 21 on stream1 as UPSERT
_ITEM_NAME_ STRING: VOD.mGBPd
_SERVICE_NAME_ STRING: IDN_RDF
_SEQUENCE_NUMBER_ INT32: 1
_ITEM_STALE_ INT32: 0
ROW80_1 STRING: VOD.mGBPd SI Quote Publication
```

```
ROW80_2 STRING:
ROW80_3 STRING: DATE:03/07/2008 Time:11:09
ROW80_4 STRING:
ROW80_5 STRING: Time Venue SI Bid Size Bid Price Ask Price Ask Size
Status
ROW80_6 STRING: ==== ===== == ======== ========= ========= ========
======
ROW80_7 STRING: 110937 GSILGB2XXXX GSIL 1 150.9000 150.9500 1 OPEN
ROW80_8 STRING: 070021 SBILGB2LXXX CITI OPEN
ROW80_9 STRING: 110909 CSFBGB2LXXX CSFB 329 150.7000 151.1500 329
OPEN
ROW80_10 STRING: 110942 DEUTGB22ZEQ DBBL 528 150.6500 151.2000 527
OPEN
ROW80_11 STRING: 110946 ABNAGB22XXX ABNV 483306 150.9000 150.9500
483306 OPEN
ROW80_12 STRING: 110936 UBSWGB2LEQU UBSI 1 149.7682 152.1325 1 OPEN
ROW80_13 STRING: 110828 SBUKGB21XXX CITI 20600 150.9000 151.0000
20600 OPEN
ROW80_14 STRING: 110937 SLIIGB2LXXX LEHM 3750 150.9000 150.9500 15
OPEN
ROW80_15 STRING:
ROW80_16 STRING:
ROW80_17 STRING:
(27.730) @6 DEBUG:Immediate flush for low latency; opcode=p
```

Each line of the page has its own FID to facilitate line-oriented deltas to the page. The adapter parses the partial page updates from Reuters and produces strings like the ones shown highlighted in the following extract from an adapter log file.

```
(49.934) @6 DEBUG:Processing update for VOD.mGBPd from service
IDN_RDF
(49.934) @6 INFO:Publishing VOD.mGBPd 4 of 21 on stream1 as UPSERT
_ITEM_NAME_ STRING: VOD.mGBPd
_SEQUENCE_NUMBER_ INT32: 2
ROW80_3 STRING: off:78 size:2 value:10
ROW80_11 STRING: off:2 size:3 value:101
(49.934) @6 DEBUG:Immediate flush for low latency; opcode=p
(50.315) @6 DEBUG:Processing update for VOD.mGBPd from service
IDN_RDF
(50.315) @6 INFO:Publishing VOD.mGBPd 3 of 21 on stream1 as UPSERT
_ITEM_NAME_ STRING: VOD.mGBPd
_SEQUENCE_NUMBER_ INT32: 3
ROW80_11 STRING: off:5 size:1 value:7
(50.315) @6 DEBUG:Immediate flush for low latency; opcode=p
```

The first update in the example is to write the 2-character string 10 at an offset of 78 characters in the line of the page which contains the data from the ROW80_3 FID. The second update in the example is to write the 3-character string 101 at an offset of 2 characters in the line of the page which contains the data from the ROW80_11 FID. The third update in the example is to write the 1-character string 7 at an offset of 5 characters in the line of the page which contains the data from the ROW80_11 FID. Thus, updates for page data are very concise.

Modifying Log Entry Format

You can modify the default format of log entries in two ways.

Set the environment variable ESP_RMDS_SUBSCRIBE_SYMBOL_FORMAT to 1 to configure your system to log messages that show what values flow to the Event Stream Processor on a single line rather than the default multiline format. When messages are written to a log file, this can make it easier to scan for specific items.

Use the -P option to the **esp_rmdsomm** command to specify the number of decimal places that appear on output for double type variables.

By default, log messages that show what values flow to Event Stream Processor are written in multiline format as shown:

```
(38079.526) @2 INFO:Publishing VOD.mGBPd 3 of 9 on stream1 as UPSERT
_ITEM_NAME_ STRING: VOD.mGBPd
_SEQUENCE_NUMBER_ INT32: 953
ROW80_7 STRING: off:53 size:2 value:45
```

If you set the environment variable ESP_RMDS_SUBSCRIBE_SYMBOL_FORMAT to 1 these messages are written are written in single-line format as shown:

```
(17.794) @5 DEBUG:stream1 p values: _ITEM_NAME_=VOD.mGBPd
_SEQUENCE_NUMBER_=2
ROW 80_3=off:78 size:2 value:20
```

The -P option can alter the manner in which `double` datatype variables appear, as shown by ask and last are in the following example. This affects only the way variables appear; it does not alter the contents.

```
<RowDefinition id="omm_RowDef">
<Column name="symbol" datatype="string" />
<Column name="service" datatype="string" />
<Column name="seq" datatype="integer" />
<Column name="stale" datatype="integer" />
<Column name="bid" datatype="money" />
<Column name="ask" datatype="double" />
<Column name="last" datatype="double" />
<Column name="volume" datatype="integer" />
<Column name="when" datatype="timestamp" />
</RowDefinition>
```

If you accept the default precision, variables of type `double` (for example, ASK in the following example) are written with three digits to the right of the decimal

```
(5.089) @5 INFO:Publishing EURJPY= 7 of 9 on stream1 as UPSERT
(5.090) @5 DEBUG:stream1 p values: _ITEM_NAME_=EURJPY=
_SEQUENCE_NUMBER_=1 _ITEM_STALE_=0 BID=137.4800 ASK=137.530
ACVOL_1=0
ACTIV_DATE+TIMACT=2008-10-06T21:07:00.000 (1223327220000)
```

If you specify the option -P 7 when enter the **esp_rmdsomm** command, variables of type `double` (for example, ASK in the following example) are written with seven digits to the right of the decimal. Variables of other types are not affected.

```
(4.913) @5 INFO:Publishing EURJPY= 7 of 9 on stream1 as UPSERT
(4.913) @5 DEBUG:stream1 p values: _ITEM_NAME_=EURJPY=
_SEQUENCE_NUMBER_=1 _ITEM_STALE_=0 BID=137.5200 ASK=137.5700000
ACVOL_1=0 ACTIV_DATE+TIMACT=2008-10-06T20:55:00.000 (1223326500000)
```

Reuters Logging

Turn Reuters logging on or off using the Reuters-side configuration file.

You can configure the Reuters OMM adapter's interface to RMDS to write to a logging facility. In the Reuters-side configuration file (`rmdsomm.cfg` is the one provided with the adapter), you can turn logging on or off and specify a path and file name of the log file. The Reuters interface also supports a set of "message files."

The Reuters-side configuration file contains a set of configuration entries for the Reuters "Logger" facility.

```
\Logger\AppLogger\fileLoggerEnabled                = true
\Logger\AppLogger\fileLoggerFilename              = "rfasub.{p}.log"
```

This configuration turns on Reuters logging for the Reuters OMM adapter. The log messages are written to the `rfasub.`**PID**`.log` file, where **PID** is the adapter's process ID.

The first line in this set, `\Logger\AppLogger\windowsLoggerEnabled = false`, pertains to a Windows logging facility that is not supported for the Reuters OMM adapter.

These example lines are from `rmdsomm.cfg`, the file that configures an adapter that subscribes to RMDS.

The same file contains configuration entries for Component Loggers, as follows:

```
\Logger\ComponentLoggers\Connections\messageFile   =\
   "config/messages/RFA7_Connections.mc"
\Logger\ComponentLoggers\Adapter\messageFile       =\
   "config/messages/RFA7_Adapter.mc"
\Logger\ComponentLoggers\SessionCore\messageFile   =\
   "config/messages/RFA7_SessionLayer.mc"
\Logger\ComponentLoggers\SSLED_Adapter\messageFile =\
   "config/messages/RFA7_SSLED_Adapter.mc"
```

Log Messages

Examples of typical entries from the adapter log file for the Reuters OMM adapter.

The actual format and working of the log messages, as well as the nature of the events logged and the log levels associated with these events, may change in subsequent releases of the adapter.

- **Message: –** `NOTICE:Item BARC.VX is closed: No Quality of Service is available to process subscription, timeout expired`
- **Cause: –** the value for the Reuters user name in the Reuters config file is incorrect (verify the case-sensitivity) or the Reuters Service name in the map file is incorrect.
- **Message: –** `DEBUG: Immediate flush for low latency`
- **Cause: –** data received from RMDS is being sent to Event Stream Processor immediately.

- **Message: –** `NOTICE:XMLRPC ERROR-116: The connection to the server could not be established. Please make sure the server is up, and check the specified host name/port, user name/password, and encryption settings. If a host name is specified, make sure that it can be resolved through a DNS lookup. (5.092) @1 INFO:Could not connect to SP; (tigris: 12190 cimtest) will retry in 5 seconds.`
- **Cause: –** cannot connect to the server running Event Stream Processor.
- **Message: –** `Ignoring market data event because no significant fields updated`
- **Cause: –** the adapter received data from Reuters, but none of the fields were of interest to Event Stream Processor stream, so no data was sent.
- **Message: –** `ERROR: Error publishing: PUBLICATION ERROR-442: The send method of this publication object failed.`
- **Cause: –** connection to Event Stream Processor unsuccessful during a message transmission.
- **Message: –** `ERROR:Mismatch between SAP Sybase Event Stream Processor stream (9 columns) and adapter (31 columns for stream: stream1)`
- **Cause: –** the number of columns defined in the adapter did not match the number of columns in the stream.
- **Message: –** `WARNING: Event Stream Processor down, dropping all subscriptions`

  followed by multiple iterations of a message similar to:

  `DEBUG: Unsubscribing item: EUR= service: IDN_RDF`
- **Cause: –** lost connection to Event Stream Processor. Stopping subscriptions to RMDS data since the adapter has nowhere to put it.
- **Message: –** `WARNING: Discarding data rec'd after unsubscribe`
- **Cause: –** before the adapter shut off the subscription, additional data arrived. The data has been discarded because there is no connection to Event Stream Processor.
- **Message: –** `DEBUG: Processing update for EUR= from service IDN_RDF`
- **Cause: –** an update for RIC "EUR=" on service named "IDN_RDF" has arrived.
- **Message: –** `WARNING: Event Stream Processor down, dropping all subscriptions`

  followed by numerous repetitions of:

  `DEBUG: Unsubscribing item: EUR= service: IDN_RDF`

- **Cause: –** lost connection to Event Stream Processor. Stopping subscriptions to RMDS data since the adapter has nowhere to put it.
- **Message: –** `WARNING: Discarding data rec'd after unsubscribe`
- **Cause: –** before the adapter shut off the subscription, additional data arrived. The data has been discarded, because there is no connection to Event Stream Processor.
- **Message: –** `EMERGENCY: Fatal Error at line 0, column 0 of config file: An exception occurred! Type:RuntimeException, Message:The primary document entity could not be opened. Id=/home/sybase/adapter/trunk/src/ReutersAdapter/xxsubexample.xml`
- **Cause: –** specified configuration file is unavailable.
- **Message: –** `EMERGENCY: Fatal Error at line 0, column 0 of config file: An exception occurred! Type:RuntimeException, Message:The primary document entity could not be opened. Id=/home/sybase/adapter/trunk/src/ReutersAdapter/xxsubexample.xml`
- **Cause: –** specified config file is unavailable.

RTView Adapter

The SAP Sybase Event Stream Processor RTView adapter is an external adapter that streams data from Event Stream Processor to the RTView® Enterprise Dashboard. RTView Enterprise software from SL Corp. is required to operate this adapter.

While this document provides information on configing the RTView software for use with the adapter, you should also consult your SL Corp. documentation for complete details and the most up-to-date information.

Datatype Mapping for the RTView Adapter

Event Stream Processor datatypes map to RTView datatypes.

Event Stream Processor Datatype	SL RTView Datatype
`boolean`	`boolean`
`integer`	`integer`
`long`	`long`
`float`	`double`

Event Stream Processor Datatype	SL RTView Datatype
`date`	`date`
`timestamp`	`date`
`string`	`string`
`money, money(1)...money(15)`	`double`
`binary`	`string`
`interval`	`long`
`bigdatetime`	`date`

Installing the RTView Adapter

To install the RTView adapter, set the environment variables for the adapter and the RTView software.

Prerequisites

- Install either version 5.8 or 5.9 of the Enterprise RTView software from SL Corporation on the client machine.
- Install Java Software Development Kit 1.7 (or higher) on the client machine.
- Set the JAVA_HOME environment variable to the root directory of the installation.

Task

1. Create an environment variable called RTVIEWADAPTER_HOME and set its value to the folder `%ESP_HOME%\adapters\rtview` for Windows and `$ESP_HOME/adapters/rtview` for Unix.
2. Verify that the RTV_HOME environment variable is set to the location of the enterprise RTView installation. It should be set automatically during installation.
3. Add the RTView `lib` and `bin` folders to the PATH environment variable. For example, update PATH to `$RTV_HOME/bin;$RTV_HOME/lib;$PATH` for Unix and `%RTV_HOME%\bin;%RTV_HOME%\lib;%PATH%` for Windows.

Configuration: Creating and Updating an SAP Connection

Create and update the `ESPOPTIONS.ini` configuration file with connection information for the Server.

You can create multiple connections to the Server. Each connection has a specific server, host, and properties. You require at least one connection to the Server.

1. To start a new dashboard project, create a new folder for it. To open an existing project, select **Start > Run**.
2. Start the project in the RTView Display Builder by typing:

   ```
   %RTVIEWADAPTER_HOME%/bin/start_builder.bat <project_filepath>
   [<rtv_file_name>.rtv]
   ```

   - `<project_filepath>` is the absolute file path of the project folder.
   - `<rtv_file_name>` is the name of an existing dashboard project. When creating a new .rtv file, do not supply a file name: the RTView Display Builder opens into a blank dashboard project called `unnamed.rtv`, which you can then save with a desired name into the new dashboard project folder.

   Note: On Windows, if you type the start builder command by using the project folder argument without a file name, it looks for a file named "<projectfolder>.rtv". If such a file does not exist in the folder, you see a message that the file cannot be opened. Click **OK** to launch the builder. This is a known RTView issue.
3. In the RTView Display Builder, select **Tools > Options**.
4. In the left pane of the Application Options window, choose **ESP**.
5. In the ESP Connections tab, click **Add**.
6. To modify the properties of an existing connection, double-click the connection.
7. Fill in the appropriate connection information and click **OK**.
8. Click **Apply**, then **Save** to save the connection information in the `ESPOPTIONS.ini` configuration file. When asked if you want to save the configuration file to the `lib` folder in the adapter installation directory, click **No**. This ensures your connection information is applied only to the current project.

Next

After you have created and edited the connection, restart either the RTView Display Builder or the Server for the changes to take effect.

Event Stream Processor Parameters

Parameters that you can specify within the `ESPOPTIONS.ini` configuration file to create a connection to Event Stream Processor.

Parameter	Description
authType	Type: `choice` (Required) Specifies the method used to authenticate to Event Stream Processor. Default is UserPassword. • UserPassword – **user** and **password** parameters are required. • ServerRSA – **user**, **keyStore**, and **keyStoreFile** parameters are required. • Kerberos – **user**, **krbKdc**, **krbRealm**, **krbService**, and **krbCache** parameters are required.
projectUri	Type: `string` (Optional) Specifies the total project URI to connect to Event Stream Processor cluster. For example, `esp://hostname:port/workspace/project`. No default value.
keyStore	Type: `string` (Optional) Specifies the location of the RSA keystore, and decrypts the password value. Required if **authType** is set to ServerRSA. No default value.
keyStorePassword	Type: `string` (Optional) Specifies the keystore password, and decrypts the password value. Required if **authType** is set to ServerRSA. No default value.
krbKdc	Type: `string` (Optional) Specifies host name of Kerberos key distribution center. Required if **authType** is set to Kerberos. No default value.
krbRealm	Type: `string` (Optional) Specifies the Kerberos realm setting. Required if **authtype** is set to Kerberos. No default value.

Parameter	Description
krbService	Type: `string` (Optional) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **authtype** is set to Kerberos. No default value.
krbCache	Type: `string` (Optional) Specifies the location of the Kerberos ticket cache file. Required if **authtype** is set to Kerberos. No default value.
user	Type: `string` (Required) Specifies user name required to log in to Event Stream Processor (see **authType**). No default value.
password	Type: `string` (Optional) Specifies the password required to log in to Event Stream Processor. Required for UserPassword authentication scheme (see **authType**). No default value.
isEncrypted	Type: `string` (Optional) Specifies whether password is encrypted. Valid values are true or false. If set to true, password is an encrypted field. This ensures that the Server recognizes the password as encrypted text and is able to decrypt the password at runtime. Default value is true.
getBase	Type: `string` (Optional) Specifies whether the Server sends existing data in stream at the time the subscription is set up. Valid values are true or false. If set to true, the Server does not send existing data in the stream. Default value is false.
droppable	Type: `string` (Optional) Specifies whether the Server drops this connection if that client cannot keep up. If set to true, the Server drops the connection. Default value is false.
lossy	Type: `string` (Optional) Specifies whether the Server may discard records if the client cannot keep up. If set to true, the Server discards records. Default value is false.

Parameter	Description
shineThrough	Type: `string` (Optional) Used for update and upsert operations. If a field is set to shine through, then an update to an existing record does not affect the value of that field.
refreshInterval	Type: `integer` (Optional) Specifies the pulse interval. Event Stream Processor consolidates data and sends it periodically at intervals as specified in milliseconds. A value of 0 or less disables pulsing. No default value.
dateFormat	Type: `string` (Optional) Specifies date format. Default value is YYYY-MM-DDHH24:MI:SS:FF.
timestampFormat	Type: `string` (Optional) Specifies timestamp format. Default value is YYYY-MM-DDHH24:MI:SS:FF.
delimiter	Type: `string` (Required) Used in the publisher command line. Default value is ##.
defaultconnection	Type: `string` (Required) Specifies default connection settings used to connect to the Server. Default value is conn1.
retryInterval	Type: `integer` (Required) Specifies how long to wait to reconnect to the Server if the connection is broken. Default value is 0.
pollinterval (seconds)	Type: `integer` (Required)Specifies how long to wait to poll data. Default value is 0.

Operation

Once you have installed the RTView adapter, you can begin using the Display Builder and the Display Viewer.

Starting the RTView Display Builder

To build and run dashboard projects, start the Display Builder from the command line.

Prerequisites

Start the Server.

Task

Running the Display Builder from the adapter installation folder in the command line ensures that the Builder links to the Server upon start-up.

SAP recommends that you place each dashboard project in its own folder. You can then start the Display Builder from this folder.

1. To start a new dashboard project, create a new folder for it. To open an existing project, select **Start > Run**.
2. Start the project in the RTView Display Builder by typing:

```
%RTVIEWADAPTER_HOME%/bin/start_builder.bat <project_filepath>
[<rtv_file_name>.rtv]
```

   - `<project_filepath>` is the absolute file path of the project folder.
   - `<rtv_file_name>` is the name of an existing dashboard project. When creating a new .rtv file, do not supply a file name: the RTView Display Builder opens into a blank dashboard project called `unnamed.rtv`, which you can then save with a desired name into the new dashboard project folder.

Note: On Windows, if you type the start builder command by using the project folder argument without a file name, it looks for a file named "<projectfolder>.rtv". If such a file does not exist in the folder, you see a message that the file cannot be opened. Click **OK** to launch the builder. This is a known RTView issue.

Starting the RTView Display Viewer

To begin viewing runtime data, start the Display Viewer from the command line.

Prerequisites

Start the Server.

Task

Running the Display Viewer from the adapter installation folder in the command line ensures that the Viewer links to the Server upon start-up.

SAP recommends that you place each dashboard project in its own folder. You can then start the Display Viewer from this folder.

1. To start a new dashboard project, create a new folder for it. To open an existing project, select **Start > Run**.
2. Start the project in the RTView Display Viewer by typing:

```
%RTVIEWADAPTER_HOME%\bin\start_viewer.bat <project_filepath>
<rtv_file_name>.rtv
```

   - `<project_filepath>` is the absolute file path of the project folder.
   - `<rtv_file_name>` is the name of the .rtv file of the dashboard. You need to specify this to start the Viewer.

Creating Shortcuts for Dashboard Projects

Starts the Display Builder or Viewer from a convenient location.

The shortcut starts the Builder or Viewer and simultaneously opens a specified dashboard project.

1. At the location where you want to create the shortcut, select **File > New > Shortcut** from the menu bar, or right-click and select **New > Shortcut**.
2. In the Create Shortcut wizard, assign the shortcut a name, then enter `%RTVIEWADAPTER_HOME%\bin\start_builder.bat <project_filepath> [<rtv_file_name>.rtv]` as the location of the item.

   `<project_filepath>` is the absolute file path of the project folder.
   `<rtv_file_name>` is the name of the .rtv file of the dashboard you want to open.

   Click **Next**.
3. Right-click the shortcut and select **Properties**.
4. In the Properties window, set the Run field to **Minimized**.
5. Repeat steps 2—4 to create a corresponding shortcut that starts the dashboard project in the Display Viewer.

   Enter `%RTVIEWADAPTER_HOME%\bin\start_viewer.bat <project_filepath> <rtv_file_name>.rtv` as the location of the item.

Dashboard Objects and Data Streams

You can connect most RTView dashboard objects to Event Stream Processor data streams. With this connection, dashboard objects can receive real-time data from streams.

There are two approaches for connecting to streams, depending on the type of stream:

- If the stream produces updates and deletes against keyed entries, first set up an intermediate object known as a cache. You can then connect the dashboard objects.
- If the stream contains insert elements, such as a timeseries, connect the dashboard object directly to the stream.

Creating a Cache

Use the RTView Builder to create a cache in a separate .rtv file, then import the file into the main dashboard file.

A cache is a datasource that allows users high-speed analytic processing of real-time data and the comparison of current real-time values against historical data. It is an intermediate datasource when connecting a dashboard object to an Event Stream Processor data stream that produces updates and deletes against user-entered values. From the RTView Builder:

1. Select **File > New** to create a new .rtv file.
2. Select **Tools > Caches**, then click **Add** in the Caches tab at the bottom of the main Display Builder window.
3. Enter a name for the new cache and set its type to **Table**.
4. Edit the cache properties:

Property	Procedure
valueTable	**1.** Right-click **valueTable** and select **Attach To Data > ESP**. **2.** Select the name of the stream that the cache subscribes to. **3.** (Optional) Choose specific stream columns; however, if you select specific columns, you must select the primary-key columns.
indexColumnNames	**1.** Click the ellipsis (**...**) beside the **indexColumnNames** field. **2.** Specify the key columns for the stream, separating the column names with a semicolon. The column names are case-sensitive.
rowsToDeleteTable	rowsToDeleteTable is a data attachment that removes selected rows from the cache tables. Rows are removed if their index column values match those of a row in the table data coming from this attachment. **1.** Right-click **rowsToDeleteTable** and select **Attach To Data > ESP**. **2.** Event Stream Processor does not have a rowsToDeleteTable, so you must use a virtual table name. For example, to use the Positions stream name, use !Positions for the rowsToDeleteTable property.
maxNumberOfRows	**1.** Specify the number of rows of historical data to save. The default value is zero. Anything larger than zero enables storage of historical data. For every key value, N rows of history are maintained, where N is the number of rows specified.

Property	Procedure
Max Internal Buffer Size	(Optional) Max Internal Buffer Size limits the size of the buffer that the adapter uses to display data. This prevents memory growth when connecting the adapter to an ESP window with no retention or to an ESP stream. If multiple RTView tables using this property are attached to the same window or stream, only the highest value for this property is used. **1.** Specify the number of rows of data to store in the buffer. Set the value of this property to the same value as the maxNumberOfRows property. If the lossy parameter in the connection properties is set to true, use this property.

5. Save the file.
6. Import this file to the main dashboard file. From the main dashboard file:
   a) Select **Tools > Options > Caches**.
   b) Click **Add** and select the cache created in step 3.
   c) Click **Apply**, then **OK**.

   Note: If you make any changes to the cache after importing it to the main display file, select **Options > Caches > Refresh Selection**.
7. Save, then close, the file.
8. Repeat steps 1 to 7 for every cache you are creating.

Example: Attaching an Object to a Cache

Attach a dashboard table object to a previously created cache.

You cannot directly connect dashboard objects to streams that make updates or deletes against keyed entries. Connect a cache to the stream, then connect the object to the cache.

Attach any object to streams, either directly or through caches, by setting the object's value property under the Data heading in the Object Properties pane. For a table object, this property is called valueTable.

In this example, first attach a table to a dashboard, then use the valueTable property to attach the table to a previously created cache that connects to an Event Stream Processor stream.

1. From the **Tables** tab in the Object Palette, select a table, then click the canvas to add the table object to the dashboard.
2. Import the cache to the dashboard project:
   a) Select **Tools > Options > Caches**.
   b) Click **Add** and select the cache.
   c) Click **Apply**, then **OK**.
3. Right-click the valueTable property of the table object, and go to **Attach To Data > Cache**.
   This opens a dialog box to configure the datasource.

4. Choose the cache from step 2.
5. Choose **Current** in the Table field and select the columns you want to view.
6. (Optional) Select Filter Rows to Basic or Advanced to view a subset of the data.
7. Click **Apply**, then **OK**.

Example: Attaching an Object to a Stream

Attach a Dashboard table object directly to a stream that contains only inserts.

Attach any object to streams, either directly or through caches, by setting the object's value property under the Data heading in the Object Properties panel. For a table object, this property is called valueTable.

1. From the **Tables** tab in the Object Palette, select a table, then click the canvas to add the table object to the dashboard.
2. Right-click the valueTable property of the table object and select **Attach To Data > ESP**.
3. Select the connection to use, and the table and columns you want to view.
4. If desired, filter the rows and columns of the stream to view a subset of the data.

Example: Creating a Function

Create a function that returns a list of table values to populate a listbox object (labelled obj_c1tlb).

Functions allow you to automate common calculations, such as taking the average value of a table column and adding the values of multiple data attachments.

There are several functions grouped into two categories, Scalar Functions and Tabular Functions, that act on scalar and tabular data, respectively.

From the Display Builder:

1. Select **Tools > Functions**, or click the **Functions** tab at the bottom of the Display Builder window, and click **Add**.
2. Enter a name for the function and choose an appropriate function type. For example, choose Count Unique Values to return a list of unique values from a column in a table or stream. If you select this function, you are prompted to specify a table or stream column.
3. Right-click the Table Column field and select **Attach To Data > ESP**. Choose a connection to the Event Stream Processor stream name and a column defined in the stream.
4. Create a listbox object in the Controls tab of the Object Palette.
5. Right-click the **listValues** property of the listbox object properties, and select **Attach To Data > Functions**. Choose the function you created in step 2 and select **Value** in the Columns field to bind the function to the listbox object.
   Whenever a new value appears in a column in the stream that the function has been attached to, that value also appears automatically in the listbox object.

6. Set the `selectedValue` and `varToSet` properties of the listbox object so the selections made in the listbox can be used elsewhere, such as for publishing to Event Stream Processor.

Publishing to Event Stream Processor

Add user control objects to a dashboard project so you can actively interact with Event Stream Processor streams.

The RTView adapter supports publishing data to the Event Stream Processor from a dashboard. The dashboard sends an event through the RTView adapter to an input stream on the Server.

1. Add one or more control objects to the dashboard.

   Add a control for each field you want to set. You can use input boxes, picklists, listboxes, checkboxes, and buttons.

   a) Use the Tools menu in the Display Builder to create a variable for each field.
   b) Attach each control to a variable.
   c) Add variables by selecting **Tools > Variables** and specifying the name, initial value, and type of the variable.
   d) (Optional) Select **Tools > Functions** and create a function to determine the user choices in the control and attach the control to that function.
   e) (Optional) Link the property of other objects on the dashboard to the control.
2. Associate an action command with the Control Object by right-clicking **actionCommand** under the Interaction category, and selecting **Define Command > ESP**. Enter a publish command:

```
conn_name.publish ## opcode ## stream_name ## col_value_1 ##
col_value_2 ...
```

Parameter	Description
##	The argument delimiter, which you can set to anything. Default value is ##. Select **Tools > Options**, select **ESP** on the left pane, then click **Add** button. The delimiter must have a space before and after it.
conn_name	A predefined connection to be used for publishing.
opcode	The operation to perform. Valid values are: • 1 — INSERT • 3 — UPDATE • 5 — DELETE • 7 — UPSERT • 13 — SAFE DELETE
stream_name	The name of the target stream.

Parameter	Description
col_value_...	The value of a column. The number of column values must equal the number of columns in the target stream. You can specify NULL values by entering two single quotation marks with nothing between them (for example, '') . Default value is ''. The RTView Viewer publishes empty fields as two single quotes (''). Because the default nullValue property is also '', a NULL is inserted into Event Stream Processor for corresponding columns. To specify an empty string, change the Null Value property in the Add ESP Connection window to a different value. For datatypes other than string, two single quotation marks without a space between them (for example, '') always represents a NULL value.

- Specify date and timestamp values in the same format as in the Date and Timestamp format properties in the Add ESP Connection window. This is the same format specification the Java SimpleDateFormat object uses.
- If the publish command is successful, there is no response. If the command does not succeed, you see an appropriate error message.

Logging

The RTView adapter uses Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing the logging configuration is part of the RTView adapter distribution.

You can modify the logging levels of the RTView adapter in the `log4j.properties` configuration file, located in `%ESP_HOME%\adapters\rtview\config`. Set the *ADAPTER_CLASSPATH* environment variable to point to this directory.

The configuration file settings override the defaults set in `esp_adapter_rtview.jar`, the adapter's jar file.

The logging levels in `log4j.properties` are:

Level	Description
`ALL`	Logs all events.
`DEBUG`	Logs general debugging events.
`ERROR`	Logs potentially recoverable application errors.
`FATAL`	Logs severe errors that prevent the application from continuing.
`INFO`	Logs events for informational purposes.
`OFF`	Logs no events.

Level	Description
TRACE	Logs fine-grained debug messages that capture the flow of the application.
WARN	Logs events that possibly lead to errors.

Note: Setting to DEBUG or ALL may result in large log files. The default value is INFO.

Running the Publisher Example

Use the RTView Display Viewer to run the provided publisher example that comes with the RTView adapter.

1. Set the username and password in the example environment:

Operating System	Step
UNIX	**1.** Edit the set_example_env.sh script **2.** Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to sybase.
Windows	**1.** Edit the set_example_env.bat script **2.** Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to sybase.

2. Start Event Stream Processor with the provided rtviewadapter.ccx model, located in %RTVIEWADAPTER_HOME%\examples, and use port 19011.

Operating System	Step
Windows	**1.** Enter cd %RTVIEWADAPTER_HOME%\examples. **2.** Start the example cluster: start_server_cluster.bat **3.** Add project to the cluster, and start it on the cluster: start_project.bat
UNIX	**1.** Enter cd $RTVIEWADAPTER_HOME/examples. **2.** Start the example cluster: start_server_cluster.sh **3.** Start the project on the cluster: start_project.sh

3. Start the RTView Display Viewer from the command line:

Operating System	Step
Windows	`%RTVIEWADAPTER_HOME%\bin\start_viewer.bat %RTVIEWADAPTER_HOME%\examples publisher.rtv`
UNIX	`$RTVIEWADAPTER_HOME/bin/start_viewer.sh $RTVIEWADAPTER_HOME/examples publisher.rtv`

Note: You can start the RTView Display Builder by replacing the **start_viewer** command with the **start_builder** command.

4. Follow the on-screen instructions.
5. Verify that data has been successfully published to the Server.

Operating System	Step
Windows	`cd %RTVIEWADAPTER_HOME%\examples run_sub.bat`
UNIX	`cd $RTVIEWADAPTER_HOME/examples run_sub.sh`

Running the Subscriber Example

Use the RTView Display Viewer to run the provided subscriber example that comes with the RTView adapter.

1. Set the username and password in the example environment:

Operating System	Step
UNIX	**1.** Edit the `set_example_env.sh` script **2.** Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	**1.** Edit the `set_example_env.bat` script **2.** Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

2. Start the Event Stream Processor with the provided `rtviewadapter.ccx` model located in `%RTVIEWADAPTER_HOME%\examples`, and use port 19011.

Operating System	Step
Windows	1. Enter `cd %RTVIEWADAPTER_HOME%\examples` 2. Start the example cluster: `start_server_cluster.bat` 3. Add project to the cluster, and start it on the cluster: `start_project.bat`
UNIX	1. Enter `cd $RTVIEWADAPTER_HOME/examples` 2. Start the example cluster: `start_server_cluster.sh` 3. Start the project on the cluster: `start_project.sh`

3. Start the RTView Display Viewer from the command line:

Operating System	Step
Windows	`%RTVIEWADAPTER_HOME%\bin\start_viewer.bat %RTVIEWADAPTER_HOME%\examples subscriber.rtv`
UNIX	`$RTVIEWADAPTER_HOME/bin/start_viewer.sh $RTVIEWADAPTER_HOME/examples subscriber.rtv`

Note: You can start the RTView Display Builder by replacing the **start_viewer** command with the **start_builder** command.

4. Use the **esp_convert** and **esp_upload** utilities to load `input.xml` data, convert XML data to stream data, and feed this data into target streams.

Operating System	Step
Windows	`cd %RTVIEWADAPTER_HOME%\examples` `run_loaddata.bat`
UNIX	`cd $RTVIEWADAPTER_HOME/examples` `run_loaddata.sh`

5. (Optional) Verify that data is successfully loaded into streams.

Operating System	Step
Windows	`cd %RTVIEWADAPTER_HOME%\examples` `run_sub.bat`

Operating System	Step
UNIX	`cd $RTVIEWADAPTER_HOME/examples` `run_sub.sh`

6. Observe the stream data that displays in tables.

Known Limitations

Learn about the known limitations of the RTView adapter.

- To modify a connection that is already connected to the Server, restart either the Builder or the Server.
- You cannot publish values containing a period. As a workaround for double and money types, the adapter lets you type a comma for a decimal point instead of a period. There is no workaround for `string` datatypes.
- Because the RTView `double` datatype maps to the `money` datatype in Event Stream Processor, you may lose precision for data that has more than 15 digits.
- RTView adapter does not handle microsecond precision in their `date` datatype. As a result, the RTView `date` datatype maps to the `bigdatetime` datatype in Event Stream Processor, with millisecond precision. This is an RTView limitation to be resolved by SL Corporation.
- In the Add ESP Connection window, if you click **OK** before entering the connection name, you see an empty box.
- Due to a limitation in RTView APIS, the adapter rejects rows with NULL columns.

Sample Input and Output Adapter

Adapter type: SampleAdapter. Edit and use the Sample Input and Output adapter as practice for building an external managed adapter.

To run the adapter, write the following Java files in the adapter's source code:

- `SampleAdapter.java` - starts and stops the adapter process. To use this file, set the parameters you require for your implementation in the `SampleAdapter\examples\adapter.xml` file.
- `Marshalling.java` - translates and formats the data for Event Stream Processor or the destination of the data. To use this file, write an extended class of `Marshalling.java` for publishing, or an extended class of `Marshalling.java` for subscribing.
- `DataTransport.java` - accesses or stores the data. To use this file, write an extended class of `DataTransport.java`.

- `CsvTransfer.java` - extends `DataTransport.java`, reads the csv file, and sends lines to the `Marshalling.java` class.
- `toEsp.java` - extends `Marshalling.java`, creates the publisher from the project, and inputs the data into Event Stream Processor.
- `toCsv.java` - extends `Marshalling.java`, creates the subscriber, and stores an output file.

See also

- *Adapter Support for Schema Discovery* on page 1005

Configuring the Sample Adapter

Configure the settings for the Sample Input and Output adapter using the ESP Server and SAP Sybase Event Stream Processor Studio.

1. Start the ESP Server, and use the command prompt to set your directory to `Sybase\ESP-5_1\cluster\examples` on Windows, or `Sybase/ESP-5_1/cluster/examples` on Linux.
2. Set your ESP folder to %ESP_HOME% on Windows, or $ESP_HOME on Linux.
3. To add projects, execute this command.

   Using Windows:

   ```
   %ESP_HOME%\bin\esp_server.exe --cluster-node node1.xml --cluster-log-properties node1.log.properties
   ```

   Using Linux:

   ```
   $ESP_HOME/bin/esp_server --cluster-node node1.xml --cluster-log-properties node1.log.propertie
   ```

4. In the examples folder, configure the `adapter.xml` file:
   a) Enter an address for **espProjectUri**.
   b) Enter a user name for **espUser**.
   c) Enter a password for **espPassword**.
   d) Enter a sample input data for **input**.
   e) Enter an output from the subscriber process for **output**.
5. Start SAP Sybase Event Stream Processor Studio, and select **File > New > Project... > Browse...**. Open the folder `Sybase\ESP-5_1\adapters\examples\managed` on Windows, or `Sybase/ESP-5_1/adapters/examples/managed` on Linux.
6. Open the `model.ccl` file, and click the **Compile Project (F7)** icon on the Studio toolbar. The file is compiled into the `model.ccx` file.
7. Open the **SAP Sybase ESP Run-Test** perspective, add the server URL "localhost:19011", and connect.

8. Right-click on the URL server, select **Create a New Workspace**, and enter "default" for the name.
9. Right-click on the workspace named "default", and select **Load Project**.
10. Browse to the folder of the compiled file to select the `model.ccx` file. In Studio, the project appears as "model" under the "default" workspace.
11. Right-click on the "model" project, and select **Start Project**.

SAP HANA Output Adapter

The SAP HANA Output adapter loads information rapidly from Event Stream Processor into a SAP HANA database.

The SAP HANA Output adapter uses multiple parallel ODBC connections to load information into the SAP HANA server. This adapter does not support guaranteed delivery, and on UNIX platforms, only 64-bit ODBC drivers and driver managers are supported.

When the adapter gets a request to shutdown, it first processes the data it received before that request. The adapter gets the shutdown signal either when the project is stopping, the ESP server is shutting down, or the adapter itself is stopped. Because the amount of time it takes for the adapter to shutdown depends on the amount of data being processed, the adapter may appear to be slow shutting down.

Prerequisites

Before you run the adapter:

- Install revision 1.00.37 or higher of SAP HANA.
- Install an ODBC driver manager and a SAP HANA ODBC client manager.
- If you are running on UNIX, use unixODBC 2.3.O or higher:
  - If you are using version 2.3.0, add “Threading=0” in the `odbcinst.ini` file to ensure optimal adapter performance.
  - If you are using version 2.3.1, create a symbolic link under `<2.3.1 installation folder>/lib` as follows:

    ```
    ln -s libodbc.so.2.0.0 libodbc.so.1
    ```

    This link is required because ESP links to `libodbc.so.1`, which unixODBC 2.3.1 has renamed `libodbc.so.2`. With the link, ESP will now use `libodbc.so.2`.
- In a Linux environment, specify the SAP HANA client libraries in your LD_LIBRARY_PATH.
- To connect to SAP HANA through SSL on Linux, use OpenSSL 0.98 and ensure the symbolic links `/usr/lib64/libssl.so` and `/usr/lib64/libcrypto.so` link to the corresponding libraries of OpenSSL 0.98
- To connect to SAP HANA through SSL on Solaris, use OpenSSL 0.98 and:

- Ensure the symbolic links `/usr/sfw/lib/64/libssl.so` and `/usr/sfw/lib/64/libcrypto.so` link to the corresponding libraries of OpenSSL 0.98.
- Remove or rename the original libraries `/usr/sfw/lib/64/libssl.so.0.9.7` and `/usr/sfw/lib/64/libcrypto.so.0.9.7`.
- Before starting the cluster node, set the environment variable `LD_LIBRARY_PATH_64=$ESP_HOME/lib:/usr/sfw/lib/64/:[unixODBC installation]/lib:$LD_LIBRARY_PATH_64`.

Here is an example of a service entry for the SAP HANA Output adapter to connect to the SAP HANA database using an ODBC connection:

```
<Service  Name="HANA" Type="DB">
     <Parameter Name="DriverLibrary">esp_db_odbc_lib</Parameter>
     <Parameter Name="DSN">HANA</Parameter>
     <Parameter Name="User"><UserName></Parameter>
     <Parameter Name="Password"><Password></Parameter>
    </Service>
```

Note: On UNIX platforms, the value for the **DriverLibrary**parameter is **esp_db_odbc64_lib**.

Property Label	Description
DB Service Name	Property ID: **service** Type: `string` (Required for adapter operation and schema discovery) The name of the service entry that represents the SAP HANA database into which information will be loaded. Service entries are specified in the Event Stream Processor `services.xml` file. You must specify an ODBC service for the database service. See the *Administrators Guide* for more information. No default value.
Target Database Table Name	Property ID: **table** Type: `tables` (Required for adapter operation; optional only if you are using schema discovery) A string value representing the name of the SAP HANA database table into which the adapter loads data. No default value.

Property Label	Description

Field Mapping	Property ID: **permutation**
	
	Type: `permutation`
	
	(Advanced) Mapping between the Event Stream Processor schema columns and SAP HANA table columns. If no permutation is specified, the Event Stream Processor schema columns must match the SAP HANA table schema exactly (same column names, order, and count). The table and column names are case sensitive so to preserve original casing, use quotations. For example:
	`create table Table1 (Col1 integer)`
	`create table "Table1" ("Col1" integer)`
	
	creates two separate tables and can be accessed using these parameter values:
	`...properties table='TABLE1', permutation='ESPCol1=COL1';`
	`...properties table='Table1', permutation='ESPCol1=Col1';`
	
	If the Data Warehouse Mode property is OFF, and you want to specify a permutation, include a mapping for at least one column of the primary key of the ESP window attached to the adapter. Without mapping at least one primary key column, the adapter fails to start.
	
	The format for this property is: <esp_columnname>=<database_columnname>:<esp_columnname>=<database_columnname>
	
	Use a colon to separate mappings. No default value.

Property Label	Description
Bulk Batch Size	Property ID: **bulkBatchSize** Type: `integer` (Advanced) Number of rows that can be inserted in a database table partition before a commit occurs. When reconnecting, the adapter only sends the last number of rows specified in the **bulkInsertArraySize** parameter. To avoid data loss on reconnect, set **bulkBatchSize** to the same value as **bulkInsertArraySize**.
Bulk Insert Array Size	Property ID: **bulkInsertArraySize** Type: `integer` (Advanced) Number of rows simultaneously inserted into a database table partition. This option must be a divisor of the **bulkBatchSize** property.
Idle Buffer Write Delay in Msec	Property ID: **idleBufferWriteDelayMSec** Type: `integer` (Advanced) Specifies the number of milliseconds that a database table partition may sit idle with uncommitted data available for insert. Default value is 1000.
Buffer Age Limit in Msec	Property ID: **bufferAgeLimitMSec** Type: `integer` (Advanced) Forces the loading of any data that has been in existence longer than the time limit. Specify a value in milliseconds between 1 and 65535. Default value is 10000.
Delay Between Reconnection Attempts	Property ID: **reconnectAttemptDelayMSec** Type: `int` (Advanced)Number of milliseconds between attempts to reconnect to the SAP HANA server. Default value is 1000.

Property Label	Description
Maximum Number of Reconnection Attempts	Property ID: **maxReconnectAttempts** Type: `int` (Advanced) Number of attempts at reconnecting to the SAP HANA server before stopping. Use -1 to retry an unlimited number of times. Default value is 1.
Data Warehouse Mode	Property ID: **dataWarehouseMode** Type: `choice` (Advanced) Specifies the type of data warehousing mode the adapter uses. Valid values are: • • **ON –** updates are converted to inserts, and deletes are ignored. • • **INSERTONLY –** only inserts are processed, and updates and deletes are ignored. • • **OFF –** (default) all inserts, updates and deletes are processed as such. If you want to specify a field mapping, or permutation, map at least one column of the primary key of the ESP window attached to the adapter. Without mapping of at least one primary key column, the adapter fails to start.
Timestamp Column Name	Property ID: **timestampColumnName** Type: `string` (Advanced) If a column name is provided, the time at which the record is added to the bulk array is stored in that column of the database record. The column name for this property cannot be used in the mapping between Event Stream Processor schema columns and SAP HANA table columns (**permutation**). If this property is empty, there is no timestamp stored. The inserted value will be in UTC time zone. The column must be a timestamp column.

Property Label	Description
Only Base Content	Property ID: **onlyBase** Type:`boolean` (Advanced) If true, the adapter processes only the base data of the window or stream to which it is attached. No further message flow is processed. Default value is false.
Include Base Content	Property ID: **outputBase** Type: `boolean` (Optional) Determines whether to process the base data of the window or stream to which the adapter is connected. The adapter processes the base data of the window or stream to which it is attached. Default value is false.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.
Timezone For Statistics	Property ID: **timezoneForStats** Type: `string` (Advanced) Controls the time zone in which midnight statistics for the adapter roll over. For example, if the adapter is left at its default value of GMT, midnight statistics reset at midnight GMT. This setting does not affect any other dates processed or returned by this adapter. Default value is GMT.

Property Label	Description
Number of Threads	Property ID: **threadCount** Type: `integer` (Advanced) The number of threads processing incoming data to be loaded into the database. If specified and the count is not in the N < x <5N range, where N is the number of table partitions, a warning message is displayed. If a value of 0 is set or the property is not specified, N threads are created.

Custom Statistics

The SAP HANA Output adapter maintains statistics to show the status of the adapter and to track its loading activities. Enable the time granularity option in the project configuration (`ccr`) file to get these custom statistics reported by the _ESP_Adapter_Statistics metadata stream:

Rows	Bytes	Average Rows
• Received in the last hour • Received since midnight • Loaded in the last hour • Loaded since midnight • Skipped* in the last hour • Skipped* since midnight	• Received since midnight • Loaded since midnight • Skipped since midnight	• Per second received over the last minute • Per second received over the last hour

* A row is skipped when the opcode does not match the adapter's current warehousing mode. For example, if the data warehousing mode is INSERTONLY, and the adapter passes in a delete or an update, this results in a skipped row.

You can also obtain additional latency statistics through the _ESP_Connectors metadata stream. See the *Administrators Guide* for more information.

Datatype Mapping for the SAP HANA Output Adapter

Event Stream Processor datatypes map to SAP HANA datatypes.

The SAP HANA Output adapter supports all available Event Stream Processor datatypes. On initialization, the adapter ensures that the schema of the Event Stream Processor window or stream to which the adapter is attached and the schema of the target SAP HANA database table have compatible column datatypes.

References to a "decimal (floating)" datatype for SAP HANA mean specifying a decimal datatype without giving precision or scale.

Event Stream Processor Datatypes	SAP HANA Datatypes	Notes
`integer`	`tinyint*, smallint*, integer, bigint, decimal (if p-s is at least 10), decimal (floating), smalldecimal`	Any SAP HANA column datatypes that are marked with an asterisk have data-dependent limitations on the mapping. Due to the performance implications of the runtime checks on these mappings, SAP recommends that you do not use them unless modification of your database schema is not an option.
`long`	`bigint, decimal (if precision scale is 19+), decimal (floating)`	
`float`	`double, float (25+), real`	The mapping of an Event Stream Processor `float` to an SAP HANA `real` datatype uses a run time check to ensure that any value passed is within the range of +-3.4x10^38.
`date`	`date, time, seconddate, timestamp`	Values are truncated to fit into the destination datatype.
`timestamp`	`date, time, seconddate, timestamp`	Values are truncated to fit into the destination datatype.
`bigdatetime`	`date, time, seconddate, timestamp`	
`string`	`varchar, nvarchar, alphanum, char, nchar, shorttext`	If the destination column is not large enough to store the `string` value, the value is truncated to fit the column and a warning is written in the ESP Server logs.

Event Stream Processor Datatypes	SAP HANA Datatypes	Notes
`boolean`	`tinyint, smallint, integer, bigint`	
`money`	`decimal (if scale is at least 4, with runtime check that precision-scale is enough to store the integer value), decimal (floating), smalldecimal`	If the `decimal` column's precision is less than 19, `decimal` columns use run time checks to ensure that the whole-number precision (precision-scale) of the column is enough to hold the whole-number decimal places of each value that passes through the adapter. To avoid these checks and increase performance, use `decimal` columns with precision 19 or higher. `smalldecimal` has a precision 16, and always uses run time checks. The adapter does not permit mapping a `money` column to a `decimal` column where the scale is greater than the precision. For example, a `decimal(4,6)` column is not allowed.

Event Stream Processor Datatypes	SAP HANA Datatypes	Notes
`money(1-15)`	`decimal, decimal (floating), smalldecimal`	If scale is at least 1-15, with runtime check that precision-scale is enough to store the integer value. If the `decimal` column's precision is less than 19, `decimal` columns use run time checks to ensure that the whole-number precision (precision-scale) of the column is enough to hold the whole-number decimal places of each value that passes through the adapter. To avoid these checks and increase performance, use `decimal` columns with precision 19 or higher. `smalldecimal` has precision 16, and always uses run-time checks. The adapter does not permit mapping a `money` column to a `decimal` column where the scale is greater than the precision. For example, a `decimal(4,6)` column is not allowed.
`interval`	`bigint`	
`binary`	`binary, varbinary`	Passing in binary data longer than the size of the database column causes that row or batch to be rejected.

Performance and Tuning Tips for the SAP HANA Adapter

Tips for improving the performance of the SAP HANA Output adapter.

- SAP recommends that you install Event Stream Processor on a different server than your SAP HANA server.
- In HANA, use column-based tables instead of row-based tables.
- Partitioning a column based table can improve the performance of multi threaded access to the table. Create partitions that your SAP HANA server can handle.
- Using the **threadCount** property to adjust the number of threads used to load data in parallel, can affect the throughput of the adapter. SAP recommends that you start the value at the number of partitions on the table into which you are loading, and keep adjusting it, to approximately 5 times that number, until performance no longer improves. Keep in mind that there is a limit of 2 billion rows per partition in SAP HANA. See your SAP HANA documentation for more details.
- If you are running on UNIX, use unixODBC 2.3.O, as SAP HANA supports this version only.
- Test optimum performance by setting different values for the **bulkBatchSize**, **bulkInsertArraySize**, **idleBufferWriteDelayMSec**, and **bufferAgeLimitMSec** properties.
- Increasing **bulkBatchSize** increases throughput but also increases latency. Increase this number as needed based on what you require for latency. If you increase **bulkBatchSize**, also increase **bulkInsertArraySize** accordingly, as **bulkInsertArraySize** is a divisor of **bulkBatchSize**.
- If a batch contains a bad row, the individual bad row is discarded, the bad rows statistic is incremented, and an error message is logged to the Server log file.

SAP RFC Input and Output Adapter

The SAP Remote Function Calls (RFC) adapter is both an input and output adapter. The RFC Input adapter executes RFCs to import data from SAP systems into Event Stream Processor, while the RFC Output adapter exports data from Event Stream Processor into SAP systems.

Both adapters include an adapter configuration file and a mapping file. Use the adapter configuration file to set up the RFC Input and Output Transporter modules, the ESP Publisher and Subscriber modules, as well as establish a connection to Event Stream Processor. Use the mapping file to specify mapping definitions for RFCs, how ESP stream columns map to RFC parameters, how SAP table columns map to ESP stream columns, or how columns of the flattened data result set map to ESP streams.

You can either manually create a mapping file or use the schema discovery functionality in SAP Sybase Event Stream Processor Studio to automatically create one. See *Discovering Schema and Creating a Mapping File for the SAP RFC Adapter* in the *Studio Users Guide* for detailed instructions on creating a mapping file in SAP Sybase Event Stream Processor Studio.

The RFC Input adapter:

- Executes selected RFCs at predefined interval and publishes the retrieved data to streams and windows in Event Stream Processor
- Operates in three modes: generic RFC (includes RFC chaining capabilities), read table, and BW

The RFC Output adapter:

- Maps data from ESP streams and windows to input parameters when RFCs are invoked
- Operates in generic RFC mode only

The RFC Output adapter also supports guaranteed delivery (GD). It can batch up ESP data rows in the RFC parameter list and submit the data in a single RFC call. The **<BatchSize>** configuration parameter controls how many ESP data row the adapter batches before invoking the RFC.

Batching makes sense for an RFC that builds up table parameters from ESP column data so as to cut down the number of RFC calls involved. If data is mapped to non-table type parameters, values are overwritten as new ESP data rows are processed, so only use batching when data is treated as data rows in a table parameter.

The RFC Input and Output adapters report custom statistics. Enable the time granularity option in the project configuration (ccr) file to get these statistics reported by the _ESP_Adapter_Statistics metadata stream.

The RFC Input adapter reports these statistics:

- AdapterRunningTime
- TotalInputRowsNumber
- SuccessInputRowsNumber
- ErrorInputRowsNumber
- InputLatency

The RFC Output adapter reports these statistics:

- AdapterRunningTime
- TotalOutputRowsNumber
- SuccessOutputRowsNumber
- ErrorOutputRowsNumber
- OutputLatency

See also

- *Chapter 4, Guaranteed Delivery* on page 1013
- *Adapter Support for Schema Discovery* on page 1005

Generic RFC Mode

Both the RFC Input and Output adapter operate in generic RFC mode. This mode includes any RFCs available from the SAP system to which you are connecting the adapter.

In this mode, the input adapter invokes any RFCs you specify in the adapter configuration file and obtains output data from the export, changing, and table type RFC parameters. Specify these RFC parameters in the mapping file. The adapter converts this data into ESP data rows which then get published into the ESP streams to which it is connected. The adapter can use values from the mapping file as input to the RFC when making RFC calls. You can also use the mapping file to configure which ESP stream columns the RFC publishes its output data.

The output adapter receives data rows from the ESP stream to which it is connected and maps this data from the stream columns to RFC input parameter during RFC invocation. This is the only mode for the output adapter.

RFC Chaining

While in generic mode, the RFC Input Adapter can invoke multiple RFCs in sequence. This is called RFC chaining.

The RFCs are invoked in the order specified in the adapter configuration file. In the mapping file, you can set the output of an RFC as the input to a subsequent RFC. You can only do this if the data types of these parameters match, meaning they must both be structure or table parameters with the same line definition or have a compatible underlying ABAP type.

Read Table Mode

The read table mode invokes the RFC named RFC_READ_TABLE which retrieves columns of a specified SAP table and returns the data as a result set.

You can map the result set to an ESP stream or window using the **Fields** parameter. Specify the table name for this RFC using the **Table** parameter in the adapter configuration file. This mode is similar to reading a database table.

There are a number of limitations for this mode:

- The combined length of all columns returned by RFC_READ_TABLE cannot be more than 512 bytes.
- When defining the WHERE clause through the **Options** configuration parameter, each line of the clause cannot be more than 71 characters long. Work around this by breaking the clause into multiple lines using multiple **Option** configuration parameter.
- When specifying an expression in the WHERE clause, always include a space between the name of a column and the operator. For example, a space is required between LAND1 and '=' in clause LAND1 = 'FR'. Without this, RFC_READ_TABLE returns an error.

BW Mode

In BW mode, the adapter submits MDX (multidimensional expression) statements against defined BW queries (cubes).

It returns data sets which can be multi-dimensional and organized on multiple axes. However, Event Stream Processor can only handle data in a tabular format (2-dimensional) meaning an axis for columns and an axis for rows. The adapter "flattens out" any data set returned by the MDX query into a table with columns and rows so that it can be input into Event Stream Processor. You can specify which columns from this table map to ESP streams using the mapping file.

The adapter flattens a multidimensional data set as follows:

- A column is created for each dimension on an axis, with the values of this column being the member of the dimension indexing the cell value.
- A column named CELL_VALUE is created for the cell values of the data set. Since the cell can store values of different measures with different datatypes, the cell value column created is of type `string`.
- A row is created for each cell value in the data set.

Specify the MDX statement using the **MDX** parameter in the adapter configuration file.

There are a number of limitations when executing MDX statements against BW queries:

- Due to the way dataset is flattened, all the columns produced are of STRING type.
- Due to limitations on the RFC involved, break the statement up into multiple lines using the **Line** parameter, with each line containing no more than 75 characters.

Control Flow

The RFC adapter loads its configuration from a file (for example, `adapter_config.xml`) and validates it against the adapter schema (`framework.xsd`).

The Adapter Controller creates an instance of the RFC adapter, and receives and executes **start** and **stop** commands.

Figure 11: SAP RFC Input Adapter Control Flow

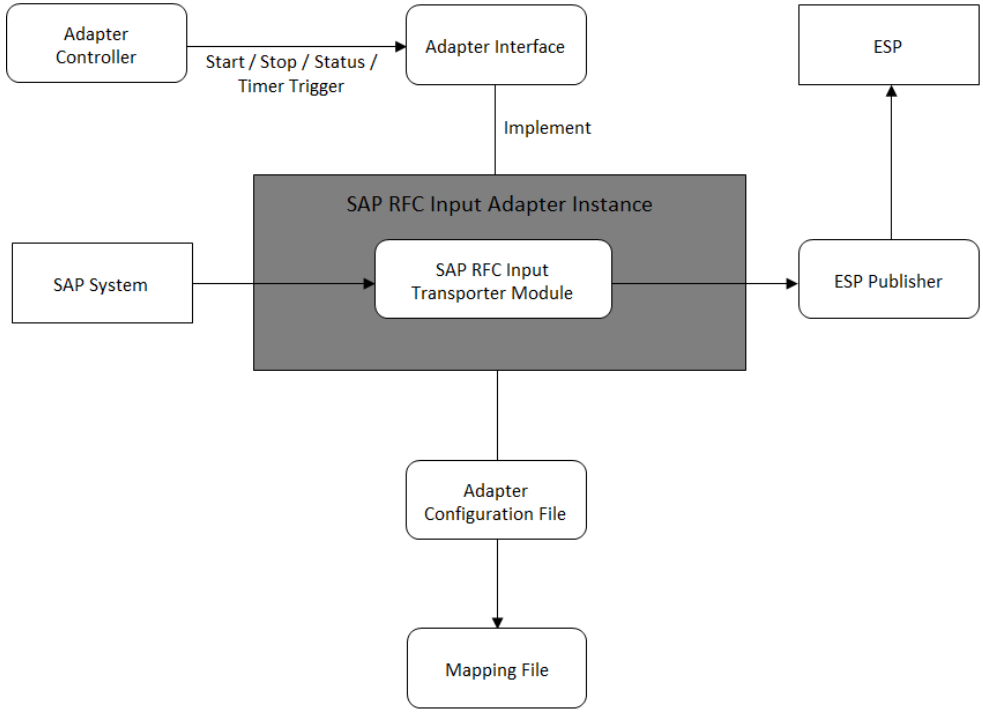
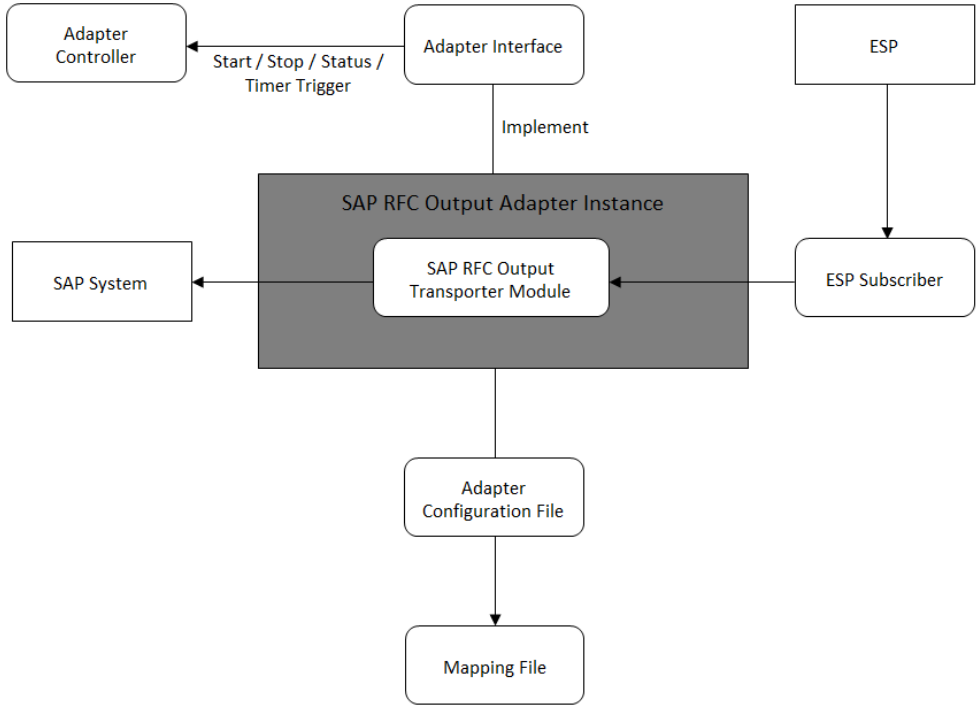


Figure 12: SAP RFC Output Adapter Control Flow



Start Command

When the **start** command is executed, it configures and starts the command and control interface, starts the adapter, establishes a connection to an SAP system, and connects the ESPPublisher and ESPSubscriber components to Event Stream Processor via the SDK interface.

The adapter ignores the **start** command if it is executed when there is a running instance of the adapter, and sends a warning.

You can have multiple instances of the adapter running. To do this, each adapter requires its own copy of the adapter configuration file which contains a unique controller port.

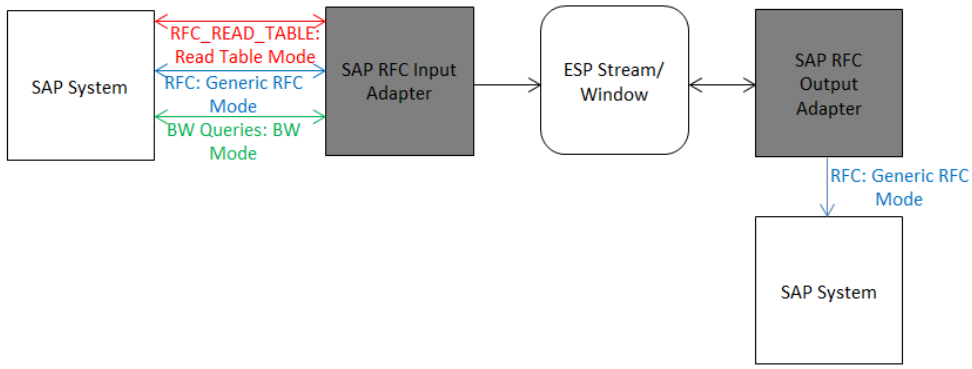
Stop Command

When the **stop** command is executed, it disconnects the ESPublisher and ESPSubscriber components from Event Stream Processor, waits until all the buffered rows between the modules are processed, disconnects the adapter from the SAP system, and terminates the adapter process. You can also execute the **immediatestop** command which is the same as the **stop** command except that it discards the buffered rows and stops adapter immediately.

If the **stop** or **immediatestop** commands are executed when there is no instance of a running adapter, the commands are ignored and a warning is sent.

Message Flow

When the SAP RFC adapter is started, it establishes a connection to an SAP system.



For the SAP RFC Input adapter, message flow begins as the adapter periodically executes RFCs and publishes data into the ESP streams and windows to which it is attached.

For the SAP RFC Output adapter, message flow begins as the adapter submits data from ESP streams and windows to an SAP system using RFCs.

Datatype Mapping for the SAP RFC Input Adapter

Event Stream Processor datatypes map to SAP ABAP datatypes.

The ABAP `T` (time) and `D` (date) datatypes have HHMMSS and YYYYMMDD formats respectively. The ESP `date` datatype to which they map has a YYYY-MM-DD HH:mm:ss format. When the ABAP `T` datatype is converted, the YYYY-MM-DD portion is set to 1970-01-01 (epoch). When the ABAP `D` datatype is converted, the HH:MM:SS portion is set to 00:00:00.

ABAP Datatype	Event Stream Processor Datatype
`I`	`integer`
`F`	`float`
`P`	`string`
`C`	`string`
`D`	`date`
`N`	`string`
`T`	`date`

ABAP Datatype	Event Stream Processor Datatype
X	binary
g(string)	string
y(xstring)	binary

Datatype Mapping for the SAP RFC Output Adapter

Event Stream Processor datatypes map to SAP ABAP datatypes.

money(15) maps to a g(string) ABAP datatype to preserve the full precision because ABAP P datatypes can only support up to 14 decimal points. All Event Stream Processor money datatypes can be mapped to g(string) as well.

Event Stream Processor Datatype	ABAP Datatypes
boolean	I
integer	I
long	P
float	F
money (legacy/1-14)	P
money(15)	g(string)
string	g(string)
date	g(string)
timestamp	g(string)
bigdatetime	g(string)
interval	P
binary	y(xstring)

RFC Adapter Directory Reference

The rfc directory contains configuration files, templates, and examples relating to the adapter.

```
bin\ (contains startup scripts and internationalization files; on
Windows, contains the native library from jco distribution)

config\ (contains an xsd file for validating mapping files)
```

```
examples\ (contains example configuration files for RFC input and
output adapters)

    bin\ (contains scripts such as adding,starting, or stopping
projects or nodes)

   bw\ (contains sample configuration files for the input adapter in
BW mode)

    chaining\ (contains sample configuration files for the input
adapter in generic RFC mode and supporting RFC chaining)

    output\ (contains sample configuration files for the output
adapter in generic RFC mode)

    output_gd\ (contains sample configuration files for the output
adapter with guaranteed delivery enabled)

   polling_input\ (contains sample configuration files for the input
adapter in generic RFC mode)

    read_table\ (contains sample configuration files for the input
adapter running in read table mode)

libj\ (contains the jar files required by the adapter)

(Solaris and Linux platforms only)lib/ (added manually after
installation; contains the jco native library)
```

Configuration

Configuration information for the SAP RFC Input and Output adapter.

Configuring the SAP RFC Input and Output adapter involves specifying values for modules and an SAP system (endpoint), as well as configuring the adapter to communicate with Event Stream Processor.

Enabling the RFC Adapter

If you plan to use the RFC adapter, you must copy some files from the SAP Java Connector (JCo) 3.0.9 to your Event Stream Processor installation.

Prerequisites

You must be authorized to download software from the SAP Service Marketplace (SMP).

Task

1. Point your browser to *https://websmp101.sap-ag.de/~form/sapnet?_SHORTKEY=01100035870000719293*
   SMP displays the **RFC Library** home page.
2. In the pane on the left, expand **SAP Java Connector > Tools & Services**.

SMP displays the **SAP Java Connector Download** page.

3. Under **Quick Links**, click on **Download SAP JCo Release 3.0**.
4. Scroll down to the operating system you are running and select the zip file for the type of processors in your system.
5. Select or create a folder, and extract the zip file to it.
6. Navigate to that folder, and copy the `sapjco3.jar` file to:

   Linux or Solaris: *`$ESP_HOME/adapters/rfc/libj`*

   Windows: *`%ESP_HOME%\adapters\rfc\libj`*
7. Then copy the `sapjco3.jar` file to:

   Linux or Solaris: *`$ESP_HOME/studio/plugins/com.sybase.cep.studio.libs_5.1.0.v(date)`*

   Windows: *`%ESP_HOME%\studio\plugins\com.sybase.cep.studio.libs_5.1.0.v(date)`*
8. Linux and Solaris only: Navigate to `$ESP_HOME/adapters/rfc` and create a directory named `lib`.
9. Linux and Solaris only: Copy the `libsapjco3.so` file to `$ESP_HOME/adapters/rfc/lib`.
10. Follow the installation instructions included with SAP JCo.

Adapter Control Port Parameter

Specify the optional **ControlPort** parameter in the adapter configuration file.

Parameter Name	Description
ControlPort	Type: `positive integer` (Required) Specifies the adapter command and control port. User commands are sent to this port on localhost. Default value is 19050.

SAP RFC Input Adapter Configuration

Configure the SAP RFC Input adapter by specifying values for the RFC Polling Input transporter, the ESP connector (EspPublisher), and Event Stream Processor.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

Transporter Module: RFC Polling Input Transporter

Parameter	Description
Module	(Required) Section containing all information for the RFC polling input transporter. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Polling	Section containing the polling parameters, **Enabled** and **TimeInterval**.
Enabled	Type: `boolean` (Required) Turns on polling for the RFC Input adapter. This parameter must always be enabled for the input adapter.

Parameter	Description
TimeInterval	Type: `integer` (milliseconds) (Required) The polling interval for the RFC Input adapter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **RFCInputTransporterParameters** element.
RFCInputTransporterParameters	(Required) Section containing parameters for the RFC Input transporter.
Host	Type: `string` (Required) The host name of the SAP system to which the adapter is connected.
SystemNumber	Type: `string` (Required) The system number of the SAP system to which the adapter is connected.
Client	Type: `string` (Required) The SAP client number for connecting to an SAP server.
SAPUser	Type: `string` (Required) The user name of the SAP system to which the adapter is connected.
SAPPassword	Type: `string` (Required) The password for the SAP system to which the adapter is connected. Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. If set to true, the password value is decrypted using the **Keystore** and **KeystorePassword** parameters. Default value is false.

Parameter	Description
Keystore	Type: `string` (Optional) Specify the location of a Java keystore file which contains the key used to encrypt or decrypt the password set in the **Password** parameter.
KeystorePassword	Type: `string` (Optional) Stores the password to the keystore file specified in the **Keystore** parameter.

RFC Input Transporter Parameters: Generic RFC Mode Parameters

Parameter	Description
RFC	(Required for the generic RFC mode only) Section containing the **Functions**, **Function**, and **Mapping** parameters for the adapter's generic RFC mode.
Functions	(Required for the generic RFC mode only) Section containing the RFCs you want the adapter to execute as specified by the **Function** parameter. You can specify multiple **Function** parameters. If you specify multiple RFCs, the RFCs execute in the order listed. This is called RFC chaining.
Function	Type: `string` (Required) Specify an RFC you wish the adapter to execute.
Mapping	Type: `string` (Required for adapter operation and schema discovery) Specify the name of the mapping file.

RFC Input Transporter Parameters: Read Table Mode Parameters

Parameter	Description
ReadTable	(Required for the read table mode only) Section containing the **Table**, **Mapping**, **RowCount**, **Fields**, **Field**, **Options**, and **Option** parameters for the adapter's read table mode.

Parameter	Description
Table	Type: `string` (Required for the read table mode) The table from which the adapter obtains data while in read table mode.
Mapping	Type: `string` (Required for adapter operation and schema discovery) Specify the name of the mapping file.
RowCount	Type: `integer` (Required) The number of rows to be read while the adapter is in read table mode.
Fields	(Optional) Section containing the **Field** parameter. If you do not specify this section, all columns from the table are read by the adapter. Specify this section to work around the total column size limit imposed by RFC_READ_TABLE in the case that a table has many columns.
Field	Type: `string` (Required) Specifies the names of the columns to be read by RFC_READ_TABLE.
Options	(Optional for the read table mode) Section containing the **Option** parameter.
Option	Type: `string` (Dependent required for the read table mode) Specifies the **WHERE** clause when querying the table during read table mode. If the **Options** parameter is specified, specify at least one **Option** parameter.

RFC Input Transporter Parameters: BW Mode Parameters

Parameter	Description
BW	(Required for the BW mode only) Section containing the **Query**, **MDX**, **Line**, and **Mapping** parameters for the adapter BW mode.

Parameter	Description
Query	Type: `string` (Required for the BW mode only) Specify the BW query to execute MDX statements against.
MDX	(Required for the BW mode only) Section containing the **Line** parameter which specfies the multidimension expression you want the adapter to execute.
Line	Type: `string` (Required for the BW mode only) Lines containing the MDX expression you want the adapter to execute. It is typical to have multiple lines as each line can contain no more than 75 characters.
Mapping	Type: `string` (Required for adapter operation and schema discovery) Specify the name of the mapping file.

ESP Connector Module: ESP Publisher

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **EspPublisherParameters** parameter.

Parameter	Description
EspPublisherParameters	(Required) Section containing parameters for the ESP publisher.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream to which the adapter publishes data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
MaxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.

Parameter	Description
UseTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.
SafeOps	Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT, and converts DELETE to SAFEDELETE. The default value is false.
SkipDels	Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. The default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the SAP RFC Input adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.

Parameter	Description
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.

Parameter	Description
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

SAP RFC Input Adapter Studio Properties

Adapter type: rfcinplugin. Set these properties for the SAP RFC Input adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Adapter Configuration File	Property ID: **config** Type: `filename` (Required for adapter operation and schema discovery) Specify the path to the adapter configuration file.
Adapter Mapping File	Property ID: **mapFilePath** Type: `filename` (Optional for adapter operation; required for schema discovery) Specify the path to and the name of the adapter mapping (xml) file. This filename must match the name of the mapping file specified in the adapter configuration file.
SAP Host	Property ID: **host** Type: `string` (Optional for adapter operation; required for schema discovery) Specify the host name or IP address of the SAP server that is being used for discovery. Default value is localhost.
SAP System Number	Property ID: **systemNumber** Type: `string` (Optional for adapter operation; required for schema discovery) Specify the system number or instance number of the SAP server that is being used for discovery. Default value is 00.
SAP Client	Property ID: **client** Type: `string` (Optional for adapter operation; required for schema discovery) Specify the client number for connecting to the SAP server that is being used for discovery. Default value is 000.

Property Label	Description
Username	Property ID: **user** Type: `string` (Optional for adapter operation; required for schema discovery) Specify the SAP system user being used for discovery. Default value is user.
Password	Property ID: **password** Type: `password` (Optional for adapter operation; required for schema discovery) Specify the password for the SAP system that is being used for discovery. No default value.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration Files for the RFC Input Adapter

Sample configuration files for the RFC Input adapter are located in the `examples` directory.

Here is a sample configuration file for a simple input adapter running in generic RFC mode.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>RFC Input Adapter</Name>
   <Description>This input adapter retrieves data from an SAP system
and publishes it to an ESP stream</Description>
    <ControlPort>19051</ControlPort>
    <Modules>
        <Module type="transporter">
            <InstanceName>RFCPollingInputTransporter</InstanceName>
            <Name>RFCInputTransporter</Name>
            <Polling>
                <Enabled>true</Enabled>
```

```
                <TimeInterval>2000</TimeInterval>
            </Polling>
            <Next>InStream_Publisher</Next>
            <Parameters>
                <RFCInputTransporterParameters>
                    <Host>hostname</Host>
                    <SystemNumber>00</SystemNumber>
                    <Client>000</Client>
                    <SAPUser>user</SAPUser>
                    <SAPPassword encrypted="false">password</
SAPPassword>
                    <Keystore>keystore.jks</Keystore>
                    <KeystorePassword>password</KeystorePassword>
                    <RFC>
                        <Functions>
                            <Function>STFC_DEEP_TABLE</Function>
                        </Functions>
                        <Mapping>mapping.xml</Mapping>
                    </RFC>
                </RFCInputTransporterParameters>
            </Parameters>
        </Module>

        <Module type="espconnector">
            <InstanceName>InStream_Publisher</InstanceName>
            <Name>EspPublisher</Name>
            <Parameters>
                <EspPublisherParameters>
                    <ProjectName>EspProject1</ProjectName>
                    <StreamName>MyInStream</StreamName>
                </EspPublisherParameters>
            </Parameters>
        </Module>
    </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject1</Name>
            <Uri>esp://localhost:19011/w1/p1</Uri>
            <!--can specify multiple uri's -->
            <Security>
                <User>sybase</User>
                <Password encrypted="false">sybase</Password>
                <AuthType>user_password</AuthType>
                <!--
                <RSAKeyStore>/keystore/keystore.jks</RSAKeyStore>
                <RSAKeyStorePassword>Sybase123</RSAKeyStorePassword>
                -->
                <!--
                <KerberosKDC>KDC</KerberosKDC>
                <KerberosRealm>REALM</KerberosRealm>
                <KerberosService>service/instance</KerberosService>
                <KerberosTicketCache>/tmp/krb5cc_user</
KerberosTicketCache>
                -->
                <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
```

```
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Here is a sample configuration file for a simple input adapter which supports calling multiple RFCs in sequence. This is also known as RFC chaining. An example would be having two RFCs: RFC1 and RFC2. RFC2 produces the data to submit to Event Stream Processor but it requires input data that is obtained from the output of RFC1. In this case, you would first call RFC1 and then RFC2 to come up with the data for publishing into Event Stream Processor.

To specify the output parameter of a previous RFC as input, in the `<parameter>` definition of the mapping file, set the valueType attribute to “chain”, the sourceType attribute to the output parameter type (for example, export, changing, or table), and the value of the element to the name of the output parameter of the previous RFC.

In an RFC chain, only the output from the last RFC in the chain can be mapped to an ESP stream. In this example, the sequence is ESP_DEV_CHAIN1 to ESP_DEV_CHAIN2 to ESP_DEV_CHAIN3.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>RFC Input Adapter</Name>
   <Description>This input adapter retrieves data from an SAP system
and publishes it to an ESP stream</Description>
    <ControlPort>19051</ControlPort>
    <Modules>
        <Module type="transporter">
            <InstanceName>RFCPollingInputTransporter</InstanceName>
            <Name>RFCInputTransporter</Name>
            <Polling>
                <Enabled>true</Enabled>
                <TimeInterval>30000</TimeInterval>
            </Polling>
            <Next>InStream_Publisher</Next>
            <Parameters>
                <RFCInputTransporterParameters>
                    <Host>hostname</Host>
                    <SystemNumber>00</SystemNumber>
                    <Client>000</Client>
                    <SAPUser>user</SAPUser>
                    <SAPPassword encrypted="false">password</
SAPPassword>
                    <Keystore>keystore.jks</Keystore>
                    <KeystorePassword>password</KeystorePassword>
                    <RFC>
                        <Functions>
                            <Function>ESP_DEV_CHAIN1</Function>
                            <Function>ESP_DEV_CHAIN2</Function>
                            <Function>ESP_DEV_CHAIN3</Function>
                        </Functions>
                        <Mapping>mapping.xml</Mapping>
                    </RFC>
```

```
                </RFCInputTransporterParameters>
            </Parameters>
        </Module>

        <Module type="espconnector">
            <InstanceName>InStream_Publisher</InstanceName>
            <Name>EspPublisher</Name>
            <Parameters>
                <EspPublisherParameters>
                    <ProjectName>EspProject1</ProjectName>
                    <StreamName>MyInStream</StreamName>
                </EspPublisherParameters>
            </Parameters>
        </Module>
    </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject1</Name>
            <Uri>esp://localhost:19011/w1/p1</Uri>
            <!--can specify multiple uri's -->
            <Security>
                <User>sybase</User>
                <Password encrypted="false">sybase</Password>
                <AuthType>user_password</AuthType>
                <!--
                <RSAKeyStore>/keystore/keystore.jks</RSAKeyStore>
                <RSAKeyStorePassword>Sybase123</RSAKeyStorePassword>
                -->
                <!--
                <KerberosKDC>KDC</KerberosKDC>
                <KerberosRealm>REALM</KerberosRealm>
                <KerberosService>service/instance</KerberosService>
                <KerberosTicketCache>/tmp/krb5cc_user</
KerberosTicketCache>
                -->
                <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Here is a sample configuration file for an input adapter running in read table mode.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>RFC Input Adapter</Name>
   <Description>This input adapter retrieves data from an SAP system
and publishes it to an ESP stream</Description>
    <ControlPort>19051</ControlPort>
    <Modules>
        <Module type="transporter">
            <InstanceName>RFCPollingInputTransporter</InstanceName>
            <Name>RFCInputTransporter</Name>
            <Polling>
```

```
                <Enabled>true</Enabled>
                <TimeInterval>2000</TimeInterval>
            </Polling>
            <Next>InStream_Publisher</Next>
            <Parameters>
                <RFCInputTransporterParameters>
                    <Host>hostname</Host>
                    <SystemNumber>00</SystemNumber>
                    <Client>000</Client>
                    <SAPUser>user</SAPUser>
                    <SAPPassword encrypted="false">password</
SAPPassword>
                    <Keystore>keystore.jks</Keystore>
                    <KeystorePassword>password</KeystorePassword>
                    <ReadTable>
                        <Table>ZC010</Table>
                        <Mapping>mapping.xml</Mapping>
                        <RowCount>20</RowCount>
                        <Fields>
                            <Field>LAND1</Field>
                            <Field>NOM</Field>
                        </Fields>
                        <Options>
                            <Option>LAND1 = 'FR'</Option>
                            <Option>AND FLAG = 'X'</Option>
                        </Options>
                    </ReadTable>
                </RFCInputTransporterParameters>
            </Parameters>
        </Module>

        <Module type="espconnector">
            <InstanceName>InStream_Publisher</InstanceName>
            <Name>EspPublisher</Name>
            <Parameters>
                <EspPublisherParameters>
                    <ProjectName>EspProject1</ProjectName>
                    <StreamName>MyInStream</StreamName>
                </EspPublisherParameters>
            </Parameters>
        </Module>
    </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject1</Name>
            <Uri>esp://localhost:19011/w1/p1</Uri>
            <!--can specify multiple uri's -->
            <Security>
                <User>sybase</User>
                <Password encrypted="false">sybase</Password>
                <AuthType>user_password</AuthType>
                <!--
                <RSAKeyStore>/keystore/keystore.jks</RSAKeyStore>
                <RSAKeyStorePassword>Sybase123</RSAKeyStorePassword>
                -->
```

```
                <!--
                <KerberosKDC>KDC</KerberosKDC>
                <KerberosRealm>REALM</KerberosRealm>
                <KerberosService>service/instance</KerberosService>
                <KerberosTicketCache>/tmp/krb5cc_user</
KerberosTicketCache>
                -->
                <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Here is a sample configuration file for an input adapter running in BW mode.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>RFC Input Adapter</Name>
   <Description>This input adapter retrieves data from an SAP system
and publishes it to an ESP stream</Description>
    <ControlPort>19051</ControlPort>
    <Modules>
        <Module type="transporter">
            <InstanceName>RFCPollingInputTransporter</InstanceName>
            <Name>RFCInputTransporter</Name>
            <Polling>
                <Enabled>true</Enabled>
                <TimeInterval>2000</TimeInterval>
            </Polling>
            <Next>InStream_Publisher</Next>
            <Parameters>
                <RFCInputTransporterParameters>
                    <Host>hostname</Host>
                    <SystemNumber>00</SystemNumber>
                    <Client>000</Client>
                    <SAPUser>user</SAPUser>
                    <SAPPassword encrypted="false">password</
SAPPassword>
                    <Keystore>keystore.jks</Keystore>
                    <KeystorePassword>password</KeystorePassword>
                    <BW>
                        <Query>Z_BOBJ/QRY_VAR_M_MDT</Query>
                        <MDX>
                            <Line>SELECT</Line>
                            <Line>{[Measures].
[BAN585JW5ZM9KK6V1T6O36S0C]} ON COLUMNS,</Line>
                       <Line>NON EMPTY [Z_REGION].[LEVEL01].MEMBERS
ON ROWS</Line>
                            <Line>FROM Z_BOBJ/QRY_VAR_M_MDT</Line>
                        </MDX>
                        <Mapping>mapping.xml</Mapping>
                    </BW>
                </RFCInputTransporterParameters>
            </Parameters>
        </Module>
```

```
        <Module type="espconnector">
            <InstanceName>InStream_Publisher</InstanceName>
            <Name>EspPublisher</Name>
            <Parameters>
                <EspPublisherParameters>
                    <ProjectName>EspProject1</ProjectName>
                    <StreamName>MyInStream</StreamName>
                </EspPublisherParameters>
            </Parameters>
        </Module>
    </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject1</Name>
            <Uri>esp://localhost:19011/w1/p1</Uri>
            <!--can specify multiple uri's -->
            <Security>
                <User>sybase</User>
                <Password encrypted="false">sybase</Password>
                <AuthType>user_password</AuthType>
                <!--
                <RSAKeyStore>/keystore/keystore.jks</RSAKeyStore>
                <RSAKeyStorePassword>Sybase123</RSAKeyStorePassword>
                -->
                <!--
                <KerberosKDC>KDC</KerberosKDC>
                <KerberosRealm>REALM</KerberosRealm>
                <KerberosService>service/instance</KerberosService>
                <KerberosTicketCache>/tmp/krb5cc_user</
KerberosTicketCache>
                -->
                <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

SAP RFC Output Adapter Configuration

Configure the SAP RFC Output adapter by specifying values for the RFC Output transporter, the ESP connector (EspSubscriber), and Event Stream Processor.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

ESPConnector Module: ESP Subscriber

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, espconnector.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **EspSubscriberParameters** parameter.
EspSubscriberParameters	(Required) Section containing parameters for the ESP subscriber.

Parameter	Description
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Specifies the unique project tag of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream from which the adapter subscribes to data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
GDMode	Type: `boolean` (Advanced) If set to true, the adapter runs in guaranteed delivery (GD) mode and all GD-related parameters become required. Default value is false.
GDKeyColumnName	Type: `string` (Advanced) Specifies column name in the Flex operator holding the GD key. The GD key is a constantly increasing value that uniquely identifies every event regardless of the opcode in the stream of interest. No default value.

Parameter	Description
GDOpcodeColumnName	Type: `string` (Advanced) Specifies name of column in Flex operator holding opcode. The opcode is the operation code (for example, inserts, update, or delete) of the event occurring in the stream of interest. No default value.
GDBatchSize	Type: `integer` (Advanced) Specifies number of records after which the control window must be updated with the latest GD key. Default value is 3.
GDPurgeInterval	Type: `integer` (Advanced) Specifies number of records after which to purge the Flex operator. Default value is 1000.
GDControlStream	Type: `string` (Advanced) Specifies name of the control window in the GD setup. The control window is a source stream that informs the Flex operator of which data has been processed by the adapter and can be safely deleted. No default value.

Transporter Module: RFC Output Transporter

Parameter	Description
Module	(Required) Section containing all information for the transporter module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.

Parameter	Description
Parameters	(Required) Section containing the **RFCOutput-TraansporterParameters** parameter.
RFCOutputTraansporterParameters	(Required) Section containing the parameters for the SAP RFC Output transporter.
Host	Type: `string` (Required) The host name of the SAP system to which the adapter is connected.
SystemNumber	Type: `string` (Required) The system number of the SAP system to which the adapter is connected.
Client	Type: `string` (Required) The SAP client number for connecting to an SAP server.
SAPUser	Type: `string` (Required) The user name of the SAP system to which the adapter is connected.
SAPPassword	Type: `string` (Required) The password for the SAP system to which the adapter is connected. Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. If set to true, the password value is decrypted using the **Keystore** and **KeystorePassword** parameters. Default value is false.
Keystore	Type: `string` (Optional) Specifies the location of a Java keystore file which contains the key used to encrypt or decrypt the password set in the **Password** parameter.
KeystorePassword	Type: `string` (Optional) Stores the password to the keystore file specified in the **KeystorePassword** parameter.

Parameter	Description
BatchSize	Type: `nonNegativeInteger` (Required) The number of ESP rows that the adapter would batch up for submission on one RFC invocation.
RFC	(Required) Section containing the **Function** and **Mapping** parameters.
Function	Type: `string` (Required) Specify an RFC you wish the adapter to execute.
Mapping	Type: `string` (Required for adapter operation and schema discovery) Specify the name of the mapping file.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the SAP RFC Output adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.

Parameter	Description
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.

Parameter	Description
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

SAP RFC Output Adapter Studio Properties

Adapter type: rfcoutplugin. Set these properties for the SAP RFC Output adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Adapter Configuration File	Property ID: **config** Type: `filename` (Required for adapter operation and schema discovery) Specify the path to the adapter configuration file.
Adapter Mapping File	Property ID: **mapFilePath** Type: `filename` (Optional for adapter operation; required for schema discovery) Specify the path to and the name of the adapter mapping (xml) file. This filename must match the name of the mapping file specified in the adapter configuration file.
SAP Host	Property ID: **host** Type: `string` (Optional for adapter operation; required for schema discovery) Specify the host name or IP address of the SAP server to which you are connecting. Default value is localhost.
SAP System Number	Property ID: **systemNumber** Type: `string` (Optional for adapter operation; required for schema discovery) Specify the system number or instance number of the SAP server to which you are connecting. Default value is 00.
SAP Client	Property ID: **client** Type: `string` (Optional for adapter operation; required for schema discovery) Specify the client number for connecting to the SAP server to which you are connecting. Default value is 000.

Property Label	Description
Username	Property ID: **user** Type: `string` (Optional for adapter operation; required for schema discovery) Specify the SAP system user. Default value is user.
Password	Property ID: **password** Type: `password` (Optional for adapter operation; required for schema discovery) Specify the password for the SAP system to which you are connecting. No default value.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration Files for the RFC Output Adapter

Sample configuration files for the RFC Output adapter are located in the `examples` directory.

Here is a sample configuration file for a simple output adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>RFC Output Adapter</Name>
    <Description>This output adapter retrieves data from an ESP
stream and sends it to an SAP system through RFC</Description>
    <ControlPort>19051</ControlPort>
    <Modules>
        <Module type="espconnector">
            <InstanceName>OutStream_Subscriber</InstanceName>
            <Name>EspSubscriber</Name>
            <Next>RFCOutputTransporter</Next>
```

```
            <Parameters>
                <EspSubscriberParameters>
                    <ProjectName>EspProject2</ProjectName>
                    <StreamName>MyInStream</StreamName>
<!--
                    <GDMode>false</GDMode>
                    <GDKeyColumnName>gdKey</GDKeyColumnName>
                    <GDOpodeColumnName>gdOpcode</GDOpodeColumnName>
                    <GDBatchSize>5</GDBatchSize>
                    <GDPurgeInterval>300</GDPurgeInterval>
                    <GDControlStream>W1_truncate</GDControlStream>
-->
                </EspSubscriberParameters>
            </Parameters>
        </Module>

        <Module type="transporter">
            <InstanceName>RFCOutputTransporter</InstanceName>
            <Name>RFCOutputTransporter</Name>
            <Parameters>
                <RFCOutputTransporterParameters>
                    <Host>hostname</Host>
                    <SystemNumber>00</SystemNumber>
                    <Client>000</Client>
                    <SAPUser>user</SAPUser>
                    <SAPPassword encrypted="false">password</
SAPPassword>
                    <Keystore>keystore.jks</Keystore>
                    <KeystorePassword>password</KeystorePassword>
                    <BatchSize>0</BatchSize>
                    <RFC>
                        <Function>ESP_DEV_INSERT_TABLE</Function>
                        <Mapping>mapping.xml</Mapping>
                    </RFC>
                </RFCOutputTransporterParameters>
            </Parameters>
        </Module>
    </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject2</Name>
            <Uri>esp://localhost:19011/w1/p1</Uri>
            <!--can specify multiple uri's -->
            <Security>
                <User>sybase</User>
                <Password encrypted="false">sybase</Password>
                <AuthType>user_password</AuthType>
                <!--
                <RSAKeyStore>/keystore/keystore.jks</RSAKeyStore>
                <RSAKeyStorePassword>Sybase123</RSAKeyStorePassword>
                -->
                <!--
                <KerberosKDC>KDC</KerberosKDC>
                <KerberosRealm>REALM</KerberosRealm>
                <KerberosService>service/instance</KerberosService>
```

```
                <KerberosTicketCache>/tmp/krb5cc_user</
KerberosTicketCache>
                -->
                <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Here is a sample configuration file for an output adapter with guaranteed delivery enabled.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>RFC Output Adapter</Name>
    <Description>This output adapter retrieves data from an ESP
stream and sends it to an SAP system through RFC</Description>
    <ControlPort>19051</ControlPort>
    <Modules>
        <Module type="espconnector">
            <InstanceName>OutStream_Subscriber</InstanceName>
            <Name>EspSubscriber</Name>
            <Next>RFCOutputTransporter</Next>
            <Parameters>
                <EspSubscriberParameters>
                    <ProjectName>EspProject2</ProjectName>
                    <StreamName>W1_log</StreamName>
                    <GDMode>true</GDMode>
                    <GDKeyColumnName>gdKey</GDKeyColumnName>
                    <GDOpodeColumnName>gdOpcode</GDOpodeColumnName>
                    <GDBatchSize>3</GDBatchSize>
                    <GDPurgeInterval>300</GDPurgeInterval>
                    <GDControlStream>W1_truncate</GDControlStream>
                </EspSubscriberParameters>
            </Parameters>
        </Module>

        <Module type="transporter">
            <InstanceName>RFCOutputTransporter</InstanceName>
            <Name>RFCOutputTransporter</Name>
            <Parameters>
                <RFCOutputTransporterParameters>
                    <Host>hostname</Host>
                    <SystemNumber>00</SystemNumber>
                    <Client>000</Client>
                    <SAPUser>user</SAPUser>
                    <SAPPassword encrypted="false">password</
SAPPassword>
                    <Keystore>keystore.jks</Keystore>
                    <KeystorePassword>password</KeystorePassword>
                    <BatchSize>0</BatchSize>
                    <RFC>
                        <Function>ESP_DEV_INSERT_TABLE</Function>
                        <Mapping>mapping.xml</Mapping>
                    </RFC>
                </RFCOutputTransporterParameters>
```

```
            </Parameters>
        </Module>
    </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject2</Name>
            <Uri>esp://localhost:19011/w1/p1</Uri>
            <!--can specify multiple uri's -->
            <Security>
                <User>sybase</User>
                <Password encrypted="false">sybase</Password>
                <AuthType>user_password</AuthType>
                <!--
                <RSAKeyStore>/keystore/keystore.jks</RSAKeyStore>
                <RSAKeyStorePassword>Sybase123</RSAKeyStorePassword>
                -->
                <!--
                <KerberosKDC>KDC</KerberosKDC>
                <KerberosRealm>REALM</KerberosRealm>
                <KerberosService>service/instance</KerberosService>
                <KerberosTicketCache>/tmp/krb5cc_user</
KerberosTicketCache>
                -->
                <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Mapping File for Generic RFC Mode

The mapping file for the RFC Input adapter running in generic RFC mode specifies mapping definitions for RFCs. The mapping file for the RFC Output adapter running in generic RFC mode specifies how ESP stream columns map to RFC parameters.

You can also use the schema discovery functionality in SAP Sybase Event Stream Processor Studio to automatically create a mapping file. See *Discovering Schema and Creating a Mapping File for the SAP RFC Adapter* in the *Studio Users Guide* for detailed instructions on creating a mapping file in SAP Sybase Event Stream Processor Studio. If you use Studio to create the mapping file, edit the generated mapping file by adding the `<variables>` and `<input>` elements as they are not included.

Input Adapter

In this mode, a mapping definition for the RFC Input adapter starts with an `<rfc>` element which includes a name attribute that indicates the RFC the definition is for. A sample mapping file for the input adapter is included in the `$ESP_HOME/adapters/rfc/examples/polling_input` directory. For example:

```
<rfc name="ESP_RFC_1">
    <variables>
        <variable name="var1">100</variable>
```

```
    </variables>
    <input>
        <import>
            <parameter name="IMPORT_TAB" valueType="table">
                <parameter name="I">99999</parameter>
                <parameter name="C">X</parameter>
               <parameter name="STR">Test Table String 1</parameter>
            </parameter>
            <parameter name="IMPORT_TAB" valueType="table">
                <parameter name="I">88888</parameter>
                <parameter name="C">Y</parameter>
               <parameter name="STR">Test Table String 2</parameter>
            </parameter>
        </import>
        <changing>
                <parameter name="IMPORTSTRUCT.I">54321</parameter>
                <parameter name="IMPORTSTRUCT.C">Z</parameter>
                <parameter name="IMPORTSTRUCT.STR">Test String</
parameter>
              <parameter name=”COUNTER” valueType=”variable”>var1</
parameter>
        </changing>
    </input>
    <mapping>
        <export>
            <fieldMapping name="RESPTEXT" espField="Column1"/>
            <fieldMapping name="EXPORT_TAB.I" espField="Column2"/>
            <fieldMapping name="EXPORT_TAB.STR" espField="Column3"/>
        </export>
        <changing>
            <fieldMapping name=”COUNTER” espField=”Column4”
saveToVariable=”var1”/>
        </changing>
    </mapping>
</rfc>
```

The mapping definition contains three sections: variables, input, and mapping. In generic RFC mode, the adapter can store output values from an RFC into temporary storage called variables. These variables can be subsequently used as input values for invoking RFCs.

Each variable definition consists of a `<variable>` element which contains a name attribute and the content of that element. The content value for the variable definition is considered the initial value of the variable. This value can be overwritten after an RFC is invoked if any of the RFC output parameters are mapped to this variable. The variable can be used as input for a parameter by specifying the valueType attribute of the parameter to variable and the value to the name of the variable.You cannot save RFC structure or table parameters to a variable.

New values are stored to the variable definitions when RFC output is mapped with a saveToVariable attribute set to the variable name on the `<fieldMapping>` entry. In the example, a variable named “var1” is defined, with the initial value set to 100. Its value is used as input for the changing parameter COUNTER. After the RFC is executed, the updated value of COUNTER is stored to var1, so in the next invocation of the RFC, the new value in var1 is used as input to COUNTER.

Input for the RFC is listed under the <input> section. Valid parameter types for this section are <import>, <changing>, and <table>. Under each parameter type element, include a <parameter> element to specify input for the RFC. Each <parameter> element contains a name attribute which indicates the name of the RFC parameter that should be populated with the content of the <parameter> element.

This mode also supports structure parameters, which are a collection of fields and parameters packed in a particular order, and table parameters, which can contain multiple structure parameters. Structures can be nested. To reference nested or subfields of a structure or table, use a dot notation to build the full name of the parameter. For example, the parameter name IMPORTSTRUCT.I references the field “I” of a structure named “IMPORTSTRUCT”. Fields “I”, “C” and “STR” of the structure parameter “IMPORTSTRUCT” are set with values 54321, Z, and Test String.

```
<changing>
                 <parameter name="IMPORTSTRUCT.I">54321</parameter>
                 <parameter name="IMPORTSTRUCT.C">Z</parameter>
                 <parameter name="IMPORTSTRUCT.STR">Test String</
parameter>
               <parameter name=”COUNTER” valueType=”variable”>var1</
parameter>
        </changing>
```

For table parameters, use the dot notation to reference up to the name of the table parameter itself, and set the attribute named valueType on the parameter to “table”. This indicates that that particular <parameter> entry represents a row of a table parameter, and all the child <parameter> elements are considered subfield values for that table entry. Multiple table entries can be defined for the same table parameter to provide more than one entry. In the example, two rows are set to the table parameter named “IMPORT_TAB”, with values of {99999, X, Test Table String 1} and {88888, Y, Test Table String 2}.

Under the <mapping> section, use the <fieldMapping> element specify the ESP stream column to which RFC output is published. Each <fieldMapping> entry represents a mapping of a parameter field to a stream column, with the name attribute indicating the RFC parameter name and the espField attribute the ESP stream column name respectively.

Similar to the <input> section, field mapping entries are grouped according to parameter type. The <mapping> section includes the <export>, <changing>, and <tables> parameter types. In the example above, the export RFC parameter RESPTEXT is mapping to Column1 in the ESP stream, and EXPORT_TAB subfield I and STR are maping to Column2 and Column3.

When a field of a table parameter is being mapped to an ESP field, multiple ESP rows can be returned as a result since the table parameter can contain multiple rows. The result is a product of all the rows in the table parameters. For example, an RFC returns the following output, with PARAM2 and PARAM3 being tables with two subfields each (PARAM21, PARAM22, PARAM31 and PARAM32):

PARAM1	PARAM2	PARAM3
1	{1, “string 1”}	{10, “string 10”}
	{2, “string 2”}	{20, “string 20”}

Both of the PARAM2 and PARAM3 tables contains two rows. If PARAM1, PARAM21, and PARAM32 map to a stream, the following four rows will be the result:

```
{1, 1, “string 10”}
{1, 1, “string 20”}
{1, 2, “string 10”}
{1, 2, “string 20”}
```

Output Adapter

Here is an example of a mapping entry for an RFC Output adapter running in generic RFC mode:

```
<rfc name="ESP_DEV_INSERT_TABLE">
    <variables>
        <variable name="var1">1</variable>
    </variables>
    <input>
        <import>
            <parameter name="PI_INT1" valueType="variable">var1</
parameter>
            <parameter name=”PI_STR2”>Static Text</parameter>
        </import>
    </input>
    <mapping>
        <import>
            <fieldMapping name="PI_INT2" espField="Column2"/>
            <fieldMapping name="PI_STR2" espField="Column1"/>
        </import>
        <export>
            <fieldMapping name="PE_INT" saveToVariable="var1"/>
        </export>
    </mapping>
</rfc>
```

Define static data under the `<input>` element in the same way as the input for the RFC Input adapter. Valid parameter types are `<import>`, `<changing>`, and `<table>`.

Define dynamic data coming from an ESP stream under the `<mapping>` element using `<fieldMapping>` entries. The format of the `<mapping>` element is similar to the RFC Input adapter `<mapping>` element. The `<fieldMapping>` entries are grouped under the parameter types of `<import>`, `<changing>`, and `<tables>` as these are the parameter types from which an RFC obtains input values. The name attribute indicates the RFC parameter to which that the value should be set, and the espField attribute indicates which ESP column the data value is from.

You can also save an output RFC parameter to a variable. In this case, the `<fieldMapping>` entry would have the name of the RFC parameter and the variable to which the value should be

saved. In the example above, the adapter maps data from ESP column Column1 to RFC import parameter PI_STR2, and from ESP column Column2 to RFC import parameter PI_INT2. It saves off the output value from export parameter PE_INT to variable var1.

Mapping File for Read Table Mode

The mapping file for an RFC output adapter running in read table mode contains `fieldMapping` entries that specify how SAP table columns map to ESP stream columns.

You can also use the schema discovery functionality in SAP Sybase Event Stream Processor Studio to automatically create a mapping file. See *Discovering Schema and Creating a Mapping File for the SAP RFC Adapter* in the *Studio Users Guide* for detailed instructions on creating a mapping file in SAP Sybase Event Stream Processor Studio.

A sample mapping file for read table mode is located in the `$ESP_HOME/adapters/rfc/examples/read_table` directory.

Each mapping entry is defined by the `<table>` element which includes a name attribute that corresponds to the SAP table that you are querying. You can define multiple `<fieldMapping>` elements within the `<mapping>` element to indicate which SAP table column maps to which ESP stream column.

Here is an example of a mapping entry for a table named ZC010. This entry maps table columns FLAG and NOM to the ESP stream column Column1 and Column2 respectively:

```
<table name="ZC010">
    <mapping>
        <fieldMapping name="FLAG" espField="Column1"/>
        <fieldMapping name="NOM" espField="Column2"/>
    </mapping>
</table>
```

There is no `<input>` section for a read table mapping entry, and the `<mapping>` section does not require the parameter type elements `<export>`, `<changing>`, or `<table>`.

Mapping File for BW Mode

The mapping file for an RFC input adapter running in BW mode contains entries for specifying how columns of the flattened data result set map to ESP streams.

You can also use the schema discovery functionality in SAP Sybase Event Stream Processor Studio to automatically create a mapping file. See *Discovering Schema and Creating a Mapping File for the SAP RFC Adapter* in the *Studio Users Guide* for detailed instructions on creating a mapping file in SAP Sybase Event Stream Processor Studio. If you use Studio to create the mapping file, edit the generated mapping file by adding the `<variables>` element as this is not included.

A sample mapping file for BW mode is located in the `$ESP_HOME/adapters/rfc/examples/bw` directory.

In this mode, a mapping entry starts with a <bwQuery> element and contains a name attribute that indicates the name of the BW query executed against. The mapping definition is listed underneath the <mapping> element, with each <fieldMapping> element representing a mapping between a result set table column and an ESP column. The attribute name of the <fieldMapping> element is the name of the result set table name column (the unique name of the dimension) or CELL_VALUE for the measure/cell value, and an espField attribute. For example:

```
<bwQuery name="Z_BOBJ/QRY_VAR_M_MDT">
    <!-- sign values: including, excluding -->
    <!-- options values: eq, gt, ge, lt, le, ne -->
    <variables>
        <variable name="[Z_VAR004]" sign="including"
options=”eq”>[Z_COUNTRY].[00000000000000000000000000000015]</
variable>
        <variable name="[Z_VAR004]" sign="including"
options=”eq”>[Z_COUNTRY].[00000000000000000000000000000067]</
variable>
    </variables>
    <!-- field mapping name is the dimension unique name -->
    <!-- for measure/cell value, use name "CELL_VALUE" -->
    <mapping>
        <fieldMapping name="[Measures]" espField="Column1"/>
        <fieldMapping name="[Z_REGION]" espField="Column3"/>
        <fieldMapping name="CELL_VALUE" espField="Column4"/>
    </mapping>
</bwQuery>
```

The usage of the <variable> definition in BW mode is different from its usage in generic RFC mode. In BW mode, use the <variable> element to define all the SAP variables declared on the BW query, along with their details and values. The name and sign attribute are mandatory. The name attribute specifies the name of the SAP variable, and the sign attribute can either have a value of “including” or “excluding” based on the usage of the variable. The options attribute can have any of the following values: “eq”, “gt”, “ge”, “lt”, “le”, “ne”, and correspond to the operators allowed in the SAP variable clause (=, >, >=, <, <=, <>).

When queries are created in BW, you can define variables as part of the query and use them in data evaluation. You can provide values to these variables at runtime when MDX statements are being executed against the query. This is similar to running a parameterized SQL query in a relational database.

The adapter uses these <variable> definitions to create the appropriate SAP variable clause and append it to the MDX statement that you specify in the adapter configuration file.

Logging

The SAP RFC Input and Output adapter use the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing the logging configuration is part of the SAP RFC Input and Output adapter distribution.

Specify the location of the logging file you wish to use in the **Log4jProperty** parameter within the adapter configuration file. You can modify the logging levels within this file or the `$ESP_HOME/adapters/framework/config/log4j.properties`, which is used by default. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
```

```
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Operation

Start and stop the adapter from the command line.

Starting the SAP RFC Adapter

To start the RFC adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

   ```
   cd %ESP_HOME%\cluster\nodes\node1
     %ESP_HOME%\bin\esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   %ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

```
%ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
%ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

UNIX:

a. Start the example cluster.

```
cd $ESP_HOME/cluster/nodes/node1
  $ESP_HOME/bin/esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
$ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

2. Start the adapter.

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/rfc/bin`
	`./start.sh <adapter configuration file name>`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/rfc/bin`
	`start.bat <adapter configuration file name>`

Stopping the RFC Adapter

To stop the RFC adapter from the command line, execute the **stop** or **immediatestop** command. Use **immediatestop** if the output queue is not empty.

When you are running the adapter from the command line, stop the adapter first before stopping the project.

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/rfc/bin`
	`./stop.sh <adapter configuration file name>`
	or
	`./immediatestop.sh <adapter configuration file name>`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/rfc/bin`
	`stop.bat <adapter configuration file name>`
	or
	`immediatestop.bat <adapter configuration file name>`

SAP Sybase IQ Output Adapter

Adapter type: sybase_iq_out. The SAP Sybase IQ Output adapter reads data from Event Stream Processor and loads it into the SAP Sybase IQ database.

Prerequisite

Before running the adapters, create a directory for the primary file location and one for the overflow file location.

To be able to load data using the SAP Sybase IQ Output adapter, you need to:

- Enable the allow_read_client_file option on the database into which the data is being loaded.
- Grant READCLIENTFILE privileges to the user that the adapter uses to connect to the SAP Sybase IQ database.
- Source the SAP Sybase IQ shell script (`IQ-15_4.sh` or `IQ-15_4.csh`) from your SAP Sybase IQ installation or SAP Sybase IQ client install before running the ESP Server. Then source the Event Stream Processor shell script (`$ESP_HOME/../SYBASE.sh` or `$ESP_HOME/../SYBASE.csh`).
- On UNIX and Windows systems, install an SAP Sybase IQ ODBC driver using an SAP Sybase IQ client installation process.

- On UNIX systems, you also need to install an ODBC driver manager. You can use the ODBC driver manager that is shipped with SAP Sybase IQ. See the SAP Sybase IQ documentation for more information.
- On UNIX systems, Event Stream Processor expects your ODBC driver manager library to be called `libodbc.so.1`. Ensure that your driver manager library has this name, or create a symbolic link from `libodbc.so.1` to your ODBC driver manager library.
- On UNIX systems, SAP recommends that you upgrade to version 2.3.0 or later of unixODBC. If you are using a version earlier than 2.3.0, set a parameter for the driver that instructs the database manager not to synchronize database access. To do this, add a line that says "Threading = 0" for your driver in the `odbcinst.ini` file.

The adapter writes data to SAP Sybase IQ load files in the native SAP Sybase IQ binary format, and loads these files in sequence into the database. The adapter supports persisting Event Stream Processor insert, update, and delete records. To improve performance, the data warehousing mode allows you to configure the adapter to either ignore updates and deletes, or to treat updates as inserts and ignore deletes.

The adapter creates files in the primary or overflow file locations as rows are received from Event Stream Processor. Once a file is successfully loaded into SAP Sybase IQ, the data is visible within the database and the file is removed from the file system. If an error occurs while loading a file, an error is logged to the ESP Server log, but the file remains on the file system. Once the problem preventing the load is resolved, you can manually attempt to reload the file using the SQL statement provided in the Event Stream Processor logs.

You can track the progress of this adapter using Sybase Control Center for Event Stream Processor. The file activity report shows each of the files processed by this adapter and lists its current state. To view the file activity report, you must add a special table to the database into which the adapter is loading. See *Enabling File Activity Monitoring for the SAP Sybase IQ Adapter*. See *Viewing File Activity for the SAP Sybase IQ Output Adapter* in the SCC for Event Stream Processor online help for additional information on the file activity report.

When the adapter receives a shutdown request because the project is stopping, the ESP Server is shutting down, or the adapter itself is stopped, it continues processing data until certain conditions are satisfied. Any data the adapter receives before the shutdown request is written into the currently writing file. This file remains on the file system until you run the adapter again, at which time it is loaded into SAP Sybase IQ. If a file is being loaded into SAP Sybase IQ when the adapter receives the shutdown request, that file continues to load until it has completed; but no further files are loaded. When the adapter restarts, it resumes processing any files created, but not loaded, by a previous instance. The same process also occurs if the ESP Server or adapter terminates unexpectedly while a file is being loaded.

Note: Because the adapter continues to process data after it receives the shutdown signal, it may appear to be slow shutting down.

The adapter also supports schema discovery and permutations. Permutations allow mapping between a compatible Event Stream Processor schema and a database schema when the two schemas are not identical. If a permutation does not provide a mapping for a database column,

that database column must be nullable. If the column is nullable, the adapter inserts NULL for each row into this column.

The default character set of the SAP Sybase IQ database must be either ASCII (if no international characters are to be loaded) or UTF-8.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Here is an example of a service entry for the SAP Sybase IQ Output adapter to connect to the SAP Sybase IQ database using an ODBC connection:

```
<Service  Name="IQ"  Type="DB">
     <Parameter Name="DriverLibrary">esp_db_odbc_lib</Parameter>
     <Parameter Name="DSN"><DataSourceName></Parameter>
     <Parameter Name="User">DBA</Parameter>
     <Parameter Name="Password">sql</Parameter>
    </Service>
```

Property Label	Description
DB Service Name	Property ID: **service** Type: `string` (Required for adapter operation and schema discovery) The name of the database service that represents the IQ database into which information will be loaded. Specify service entries in the Event Stream Processor `service.xml` file. You must specify an ODBC service for the database service. See the *Administrators Guide*. No default value.
Target DB Table Name	Property ID: **table** Type: `string` (Required for adapter operation; optional only if you are using schema discovery) A string value representing the name of the table in SAP Sybase IQ into which you wish to load data. No default value.
Include Base Content	Property ID: **outputBase** Type: `boolean` (Optional) Determines whether to process the base data of the window or stream to which the adapter is connected. The adapter processes the base data of the window or stream to which it is attached. Default value is false.

Property Label	Description
Primary File Location	Property ID: **primaryFileLocation** Type: `directory` (Required for adapter operation; optional only if you are using schema discovery) Specify the directory in which SAP Sybase IQ load files are stored temporarily until they are loaded into the system. If this directory does not exist, create it before running the adapter. Because there is sensitive information being loaded into the SAP Sybase IQ database and anyone with access to the load files can change the data being loaded, you must secure this directory as you do your database. SAP recommends that only users with access to the database have access to this directory. No default value.
Overflow File Location	Property ID: **overflowFileLocation** Type: `directory` (Optional) Specify the directory for overflow files. If this directory does not exist, create it before running the adapter. If the primary file location does not contain enough space to write out load files, the files are created in the overflow location. Because there is sensitive information being loaded into the SAP Sybase IQ database and anyone with access to these files can change the data being loaded, you must secure this directory as you do your database. SAP recommends that only users with access to the database have access to this directory. No default value.

Property Label	Description
Field Mapping	Property ID: **permutation** Type: `permutation` (Advanced) The permutation list maps the Event Stream Processor column names to the database schema column names in the specified table. If you do not specify a permutation, ensure that the Event Stream Processor stream or window columns exactly match the database schema of the destination table. For example, both must have the same order, same number of columns, and compatible datatypes. If the Data Warehouse Mode property is OFF, and you want to specify a permutation, include a mapping for at least one column of the primary key of the ESP window attached to the adapter. Without mapping at least one primary key column, the adapter fails to start. The format for this property is: <esp_columnname>=<database_columnname>:<esp_columnname>=<database_columnname> Use a colon to separate mappings. No default value.
Only Base Content	Property ID: **onlyBase** Type: `boolean` (Advanced) If true, the adapter processes only the base data of the window or stream to which it is attached. No further message flow is processed. Default value is false.
Recover Only	Property ID: **recoverOnly** Type: `boolean` (Advanced) If true, the adapter processes any files that have beencreated, but not loaded, by a previous instance of the adapter. After these files are processed, the adapter does not process further message flow. Default value is false.

Property Label	Description
Data Warehouse Mode	Property ID: **dataWarehouseMode** Type: `choice` (Advanced) Specifies the type of data warehousing mode the adapter uses. Valid values are: • • **ON –** updates are converted to inserts, and deletes are ignored. • • **INSERTONLY –** only inserts are processed, and updates and deletes are ignored. • • **OFF –** (default) all inserts, updates and deletes are processed as such. If you want to specify a field mapping, or permutation, map at least one column of the primary key of the ESP window attached to the adapter. Without mapping of at least one primary key column, the adapter fails to start.
Timestamp Column Name	Property ID: **timestampColumnName** Type: `string` (Advanced) If a column name is provided, the time at which the record is written to the load file is stored in that column of the database record. If this property is empty, no timestamp is stored. The timestamp is always provided in UTC. No default value.
Target File Size	Property ID: **targetFileSize** Type: `uint` (in MB) (Advanced) Specifies the maximum size for an SAP Sybase IQ load file before it is loaded into SAP Sybase IQ. SAP Sybase IQ load performance is better for larger files, but latency increases with file size. Default value is 2000. Each SAP Sybase IQ load file is not necessarily this size. If the adapter is not in data warehousing mode, any updates or deletes received reduce the file size. The Idle Buffer Write Delay and Buffer Age Limit parameters may also cause smaller files to be generated.

Property Label	Description
Idle Buffer Write Delay	Property ID: **idleBufferWriteDelay** Type: `uint` (in seconds) (Advanced) If the adapter has not received a row during the number of seconds specified, the previously created file is loaded. This provides latency control in situations where message flow may be sporadic. Default value is 10.
Buffer Age Limit	Property ID: **bufferAgeLimit** Type: `uint` (Advanced) The maximum amount of time (in seconds) between the first and last record in an I/O buffer. This ensures that consistent but low volume message flow does not create very high latency. Default value is 600.
Disable Referential Integrity	Property ID: **disableReferentialIntegrity** Type: `boolean` (Advanced) Disables the referential integrity constraints on the table before the loading operation is performed. This can significantly improve performance of the load, but causes problems if the source does not maintain data integrity. Default value is false.
I/O Buffer Size	Property ID: **ioBufferSizeMB** Type: `uint` (in MB) (Advanced) Determines how much data is buffered in memory before it is written to disk. Increasing this value may increase write performance, but may also increase latency. Determine the amount of memory required by the adapter to buffer by multiplying this number by the number of I/O buffers. The target file size should be a multiple of this number. Default value is 20.
Number of I/O Buffers	Property ID: **numIOBuffers** Type: `uint` (in MB) (Advanced) The number of data buffers to maintain. This number should be sufficiently high so that a buffer is always available to write into. The slower the file system, the more I/O buffers required. Default value is 5.

Property Label	Description
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are re-usable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.
Timezone For Statistics	Property ID: **timezoneForStats** Type: `string` (Advanced) Controls the time zone in which midnight statistics for the adapter roll over. For example, if the adapter is left at its default value of GMT, midnight statistics reset at midnight GMT. This setting does not affect any other dates processed or returned by this adapter. Default value is GMT.
Delay between Reconnection Attempts	Property ID: **reconnectAttemptDelayMSec** Type: `uint` (in milliseconds) (Advanced) Controls the time to delay before a new reconnection attempt. A higher value increases the time between reconnection attempts, which may impact performance. Default value is 1000.
Maximum Number of Reconnection Attempts	Property ID: **maxReconnectAttempts** Type: `uint` (Advanced) The maximum number of times the adapter attempts to reconnect to the database. To retry indefinitely, set this value to -1. Default value is 1.

Custom Statistics

The SAP Sybase IQ Output adapter maintains custom statistics to show its status and to track its loading activities. Enable the time granularity option in the project configuration (`ccr`) file to get these statistics reported by the _ESP_Adapter_Statistics metadata stream. This option helps determine how often the ESP Server gathers statistics.

Rows	Bytes	Average Rows Received/Loaded
• Received in the last hour • Received since midnight • Loaded in the last hour • Loaded since midnight • Skippd* in the last hour • Skipped* since midnight	• Received since midnight • Loaded since midnight • Skipped since midnight	• Over the last minute • Over the last hour

* A row is skipped when the opcode does not match the adapter's current warehousing mode. For example, if the data warehousing mode is INSERTONLY, and the adapter passes in a delete or an update, this results in a skipped row.

The "since midnight" statistics reset after midnight in the time zone specified by the **timezoneForStats** adapter property. If no value is set, the default behavior is reset at midnight GMT.

See also

- *Adapter Support for Schema Discovery* on page 1005

Datatype Mapping for the SAP Sybase IQ Output Adapter

Event Stream Processor datatypes map to SAP Sybase IQ datatypes.

Any SAP Sybase IQ column datatypes that are marked with an asterisk have data dependent limitations on the mapping. Due to the performance implications of the runtime checks on these mappings, SAP recommends that you do not using them unless modifyinh your database schema is not an option.

Event Stream Processor Datatypes	SAP Sybase IQ Column Datatypes	Notes

`integer`	`integer, bigint,`
	`money, decimal(p,s),`
	`numeric(p,s)`
	`tinyint*, smallint*`
		If a decimal column is used, p,s >= 10.
		If the column type in the database is `tinyint` or `smallint`, the number is checked during message flow to prevent overflow.

Event Stream Processor Datatypes	SAP Sybase IQ Column Datatypes	Notes

`long`	`bigint, decimal(p,s),`
	`numeric(p,s),`
	`unsigned int*, unsigned bigint *`
		If a decimal column is used, p-s >= 19.
		If the column type in the database is `unsigned int` or `unsigned bigint`, the number is checked during message flow to prevent overflow.

`float`	`double, decimal(p,s),` `numeric(p,s)`	For the `double` type, you must specify a precision of at least 16. Float to decimal conversions are not checked and may result in incorrect data being stored if the precision and scale for the decimal is not large enough to store the float.
`string`	`varchar, char`	If the destination column is not large enough to store the `string` value, the value is truncated to fit the column and a warning is written in the ESP Server logs.
`money`	`decimal(p,s)*, numeric(p,s)*,` `money, smallmoney*`	For `decimal` and `numeric` datatypes, s >= 4. If you set the precision to less than 19, the number is checked at runtime to ensure it fits. If the destination is `small money`, the value is checked during message flow to prevent overflow.
`bigdatetime`	`timestamp, time`	If the destination column is `time`, the date information is discarded from the incoming data ,and only the time portion is stored.
`timestamp`	`timestamp`	

Event Stream Processor Datatypes	SAP Sybase IQ Column Datatypes	Notes
`date`	`timestamp, date`	If the destination column is `date`, the time information is discarded from the incoming data, and only the data portion is stored.
`interval`	`bigint`	
`binary`	`binary`	If the `binary` object in the Event Stream Processor stream is larger than the destination column in the database, an error is logged in the ESP Server log and the row is discarded.
`boolean`	`varchar(5), bit,` `integer, bigint,` `unsigned integer, unsigned bigint,` `smallint, tinyint`	If the column is `varchar`, the adapter stores "true" for true, and "false" for false. If the column is a `bit` or other integer type, the adapter stores 1 for true or, 0 for false.

Error Handling for the SAP Sybase IQ Output Adapter

Various scenarios that write error messages in the ESP Server log.

Scenario	Result
On start-up, the adapter verifies that the service name provided belongs to an ODBC service entry, and that the specified table exists. The adapter then checks whether a permutation exists, or it assumes a default mapping from the stream or window to the indicated database table and verifies that the mapping is valid.	If any of these checks fail, the adapter logs an error message to the Server log and does not start.
There is a column in the database that is not nullable, and there is no column mapping in the permutation for that column.	The adapter fails to initialize, and an error is logged to the Server log indicating that a mapping for the required column was not provided.

Scenario	Result
Certain column mappings are valid in data-dependent situations. For example, an integer may be stored in a `tinyint` column only if its value is between 0 and 255. Refer to *Datatype Mapping for the SAP Sybase IQ Output Adapter*.	The mappings that require runtime data checks are identified at adapter start-up, and the adapter verifies only these for each record. Due to the performance impact of runtime checks on each Event Stream Processor record, SAP recommends that you do not use these mappings unless you cannot change your existing schemas . If a runtime check fails for a mapping, the adapter rejects the record. The bad rows statistic is incremented, and the adapter logs a message to the Server log indicating that it discarded a row, and the nature of the error, without providing any information about the specific record (no data values).
An Event Stream Processor `string` is longer than the `varchar` or `char` column into which it is to be stored.	The Event Stream Processor record is stored in the database and the string is truncated to fit in the column. The warning that is written to the log indicates that the string was truncated without providing any specific data from the row.
A load file cannot be loaded into the database due to a database error.	The load file remains on the file system, and an error indicating the location and name of the file, and the nature of the error is logged. You can use the **load table** statement for SAP Sybase IQ that is in the Event Stream Processor log file to reload the file into SAP Sybase IQ. The adapter does not attempt to reload any load files that have failed in the past.You must manually correct any problems and load the file.
Binary values that are too large for the database column into which they are mapped.	The individual row is discarded, the bad rows statistic is incremented, and an error is logged to the Server log.
A batch contains a bad row.	The individual row is discarded, the bad rows statistic is incremented, and an error is logged to the Server log.

Enabling File Activity Monitoring for the SAP Sybase IQ Adapter

To use the file activity report in SCC for Event Stream Processor, create a database user and table in the database into which the SAP Sybase IQ Output adapter is loading data.

1. As a user with permissions to create a user, run the `$ESP_HOME/adapters/iqoutput/enableFileActivity.sql` script on your SAP Sybase IQ database

and create tables for that user. You must run this script before the file activity report begins collecting data.

The script adds a user called RAP_USER, and a table owned by this user called `RAP_WORK_FILE`. It also grants the RAP_USER permission to perform client-side loads. RAP_USER can be used as the user through which the SAP Sybase IQ Output adapter connects to the SAP Sybase IQ database. If another user connects the adapter to SAP Sybase IQ, that user must have read and write permissions on the RAP_WORK_FILE table owned by RAP_USER. See the SAP Sybase IQ documentation for further details on these permissions.

While the adapter is running, RAP_WORK_FILE collects information about every file that is created and loaded by the adapter. This allows the file activity report to function as an archive for adapter activity.

2. As a database administrator, you may want to periodically remove information from RAP_WORK_FILE to prevent it from becoming unmanageable in size. Before removing information, ensure that the adapter is not running.
3. Then delete rows from the table using a standard SQL **delete** statement.

   If you remove rows that represent files in the process of being written or loaded, an error is written in the Event Stream Processor server logs on the next start-up of the adapter, and that file is not tracked in the file activity report. SAP recommends that you only delete rows that represent files that were completed or in error (columns for which the LOAD_STATUS is C or E respectively).
4. You must run the script before the file activity report will begin collecting data.

SMTP Output Adapter

Adapter type: smtp_out. The SMTP Output adapter sends an e-mail containing stream records.

For each record, the e-mail body contains:

- Stream name
- Columns names and values

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
SMTP Host	Property ID: **smtpHost** Type: `string` (Required) Name or IP address of the e-mail server. No default value.

Property Label	Description
fromAddress	Property ID: **from** Type: `string` (Required) E-mail address of the sender. No default value.
Importance Column	Property ID: **importanceColumn** Type: `string` (Required) Name of the stream column where the e-mail importance is stored. Valid values are: high, normal, and low. The default value is normal. The values are case-sensitive.
Address Column	Property ID: **addressColumn** Type: `string` (Required) Name of the column where a semicolon-delimited list of recipient e-mail addresses is stored. No default value.
Subject Column	Property ID: **subjectColumn** Type: `string` (Required) Name of the stream column where the e-mail subject is stored. No default value.
SMTP Port	Property ID: **smtpPort** Type: `uint` (Optional) Port used by the SMTP server. Default value is 25.
SMTP Username	Property ID: **smtpUsername** Type: `string` (Optional) Once you configure this property, the SMTP authentication requires the SMTP password to be set also.
SMTP Password	Property ID: **smtpPassword** Type: `string` (Optional) Set this property if you have set a SMTP user name.

Property Label	Description
Use SSL	Property ID: **useSSL** Type: `boolean` (Optional) If you want to enable this option, import the security certificate into the Java keystore file (located under `install/lib/jre/lib/security/cacerts`). Once the security certificate is configured, you can enable this option. Default value is false. **Important:** Enable the SMTP server for SSL before configuring this option.
Use TLS	Property ID: **useTLS** Type: `boolean` (Optional) If you want to enable this option, import the security certificate into the Java keystore file (located under `install/lib/jre/lib/security/cacerts`). Once the security certificate is configured, you can enable this option. Default value is false. **Important:** Enable the SMTP server for TLS before configuring this option.
cc Column	Property ID: **ccColumn** Type: `string` (Advanced) Name of the column where a semicolon-delimited list of recipient cc addresses is stored. By default, no cc e-mails are sent.
bcc Column	Property ID: **bccColumn** Type: `string` (Advanced) Name of the column where a semicolon-delimited list of recipient bcc addresses is stored. By default, no bcc e-mails are sent.

Property Label	Description
Column Names	Property ID: **columnNames** Type: `string` (Advanced) Colon-delimited names of stream columns whose values are included in the e-mail. By default, the e-mail contains values of all columns in the stream.
Show Column Names	Property ID: **showColumnNames** Type: `boolean` (Advanced) If true, the adapter includes the column names in the e-mail along with their values. If false, it includes only the values. Default value is true.
Number of Resend Attempts	Property ID: **resendAttempts** Type: `integer` (Advanced) The number of times to retry sending an e-mail if the initial attempt to send it fails. Default is 0, no attempt is made to resend e-mails. Choose a moderate value (0 - 10) for this property. Requiring a large number of attempts to resend the e-mail may lead to excessive memory consumption, particularly if aggravated by network problems and a high volume of records waiting to be e-mailed.
Log Alert	Property ID: **logAlert** Type: `boolean` (Advanced) If true, logs an alert at debug level 1 each time the e-mail sending has been successful or failed. Default value is true.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Date format. Default value is `%Y-%m-%dT %H:%M:%S`.

Property Label	Description
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Timestamp format. Default value is `%Y-%m-%dT%H:%M:%S`.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Known limitations:

- If you are a Microsoft® Outlook user, disable the feature that removes extra line breaks:
  1. Open Outlook, go to **Tools > Options**.
  2. On the **Preferences** tab, select **E-mail Options**.
  3. Click to clear the **Remove extra line breaks in plain text messages** check box. Click **OK** twice.

Socket CSV Input and Output Adapter

The Socket CSV Input adapter obtains CSV string data from a Socket server and publishes it to Event Stream Processor. The Socket CSV Output adapter takes data from Event Stream Processor, formats it into CSV format, and outputs it to a Socket server.

Socket CSV Input Adapter Configuration

Configure the Socket CSV Input adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

Transporter Module: Socket Input Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **SocketInputTransporterParameters** element.
SocketInputTransporterParameters	(Required) Section containing parameters for the Socket Input transporter.

Parameter	Description
Host	Type: `string` (Required if **EpFile** is set to null) If the transporter is acting as a socket client, specify the socket server name. If the transporter is acting as a socket server, do not set this parameter. No default value.
Port	Type: `integer` (Required if **EpFile** is set to null) Specify the socket server port. If you set this to -1, the adapter reads from the ephemeral port file which is specified in the **EpFile** parameter. The default value is 12345.
EpFile	Type: `string` (Required if **Host** and **Port** are set to null) Specify the file that contains the socket server name/IP and port number. No default value.
Retryperiod	Type: `integer` (Advanced) When the transporter is acting as a socket server, this parameter designates the length of time to wait for the first incoming connection before switching to the continuous state. When the transporter is acting as a socket client, this parameter designates the time period for attempting to re-establish an outgoing connection, in seconds. The default value is 0.
BlockSize	Type: `integer` (Advanced) Define the size of the data block when transporting data from the socket server to the socket client. The default value is 1024.

Parameter	Description
KeepAlive	Type: `boolean` (Advanced) If set to true, the adapter disconnects from the socket server if there are no data transports for the duration of time specified in your router configuration. For example, if you set your router configuration to two hours and there are no messages during that time, the adapter disconnects from the socket server. The default value is false.

Formatter Module: Stream to String Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **StreamToStringFormatterParameters** parameter.
StreamToStringFormatterParameters	(Required) Section containing the Stream to String formatter parameters.

Parameter	Description
Delimiter	Type: `string` (Required) Specify the symbol used to separate columns. The default value is "\n".
IncludeDelimiter	Type: `boolean` (Required) If set to true, the delimiter is part of current row. If set to false, the delimiter is not part of the current row. The default value is false.
AppendString	Type: `string` (Required if **IncludeDelimiter** is set to true) If set to true, specify the string to append to the end of each result row. No default value.
AppendPosition	Type: `string` (Required if **IncludeDelimiter** is set to true) Specify the position to which the **AppendString** parameter takes effect. There are two valid values: front and end. The default value is front.
IgnoreSpace	Type: `boolean` (Required) Specify whether to trim the space char. The default value is true.

Formatter Module: CSV String to ESP Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **CsvString-ToEspFormatterParameters** parameter.
CsvStringToEspFormatterParameters	(Required) Section containing the CSV String to ESP formatter parameters.
ExpectStreamNameOpcode	Type: `boolean` (Required) If set to true, the adapter interprets the first two fields of the incoming CSV line as stream name and opcode respectively. The adapter discards messages with unmatched stream names. The accepted opcodes are: • i or I: INSERT • d or D: DELETE • u or U: UPDATE • p or P: UPSERT • s or S: SAFEDELETE The default value is false.
Delimiter	Type: `string` (Advanced) The symbols used to separate the column. The default value is a comma (,).

Parameter	Description
HasHeader	Type: `boolean` (Advanced) Determines whether the first line of the file contains the description of the fields. Default value is false.
DateFormat	Type: `string` (Advanced) The format string for parsing date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.
TimestampFormat	Type: `string` (Advanced) Format string for parsing timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

ESP Connector Module: ESP Publisher

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **EspPublisherParameters** parameter.
EspPublisherParameters	(Required) Section containing parameters for the ESP publisher.

Parameter	Description
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream to which the adapter publishes data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
MaxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.

Parameter	Description
MaxPubPoolTime	Type: `positive integer` (Optional) Specifies the maximum period of time, in milliseconds, for which records are pooled before being published. If not set, pooling time is unlimited and the pooling strategy is governed by **maxPubPoolSize**. No default value.
UseTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.
SafeOps	Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT, and converts DELETE to SAFEDELETE. The default value is false.
SkipDels	Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. The default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the Socket CSV Input adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.

Parameter	Description
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.

Parameter	Description
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

Socket CSV Input Adapter Studio Properties

Adapter type: toolkit_socket_csv_input. Set these properties for the Socket CSV Input adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Server	Property ID: **host** Type: `string` (Required) Server host name.
Port	Propterty ID: **port** Type: `int` (Required) Server port. To read from the Ephemeral Port File, set to -1.
Ephemeral Port File	Property ID: **epFile** Type: `filename` (Advanced) File that will contain the server port number if Port is -1.
Has Header	Property ID: **csvHasHeader** Type: `boolean` (Advanced) Indicate whether the first line of the file contains the description of the CSV fields.
Retry Period (seconds)	Property ID: **retryperiod** Type: `uint` (Advanced) Period for trying to re-establish an outgoing connection.
Buffer Size	Property ID: **inputBufferSize** Type: `uint` (Advanced) The buffer size of the socket connection (bytes).

Property Label	Description
Enable TCP keepalive	Property ID: **keepAlive** Type: `boolean` (Advanced) Enable TCP keepalive on the socket connection.
Charset Name	Property ID: **charsetName** Type: `string` (Advanced) Specify the name of a supported charset.
Column Delimiter	Property ID: **csvDelimiter** Type: `string` (Advanced) Specify the symbol used to separate the columns.
Date Format	Propterty ID: **csvDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **csvTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.
Has Header	Property ID: **csvHasHeader** Type: `boolean` (Advanced) Indicate whether the first line of the file contains the description of the CSV fields.
Steam name, opcode expected	Property ID: **csvExpectStreamNameOpcode** Type: `boolean` (Optional) If set to true, the adapter interprets the first two fields of the incoming CSV line as stream name and opcode. The adapter discards messages with unmatched values.

Property Label	Description
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: Socket CSV Input Adapter

Sample adapter configuration file for the Socket CSV Input adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
  <Name>socket_csv_input</Name>
  <Description>An adapter which gets csv string from socket server,
transforms to ESP data format, and publishes to ESP stream.</
Description>
  <Log4jProperty>./log4j.properties</Log4jProperty>
  <Modules>
    <Module type="transporter">
      <InstanceName>MyExampleSocketInTransporter</InstanceName>
      <Name>SocketInputTransporter</Name>
      <Next>MyStreamingInputFormatter</Next>
      <Parameters>
        <SocketInputTransporterParameters>
          <Host>localhost</Host>
          <Port>9998</Port>
          <EpFile></EpFile>
          <Retryperiod>60</Retryperiod>
          <Blocksize>512</Blocksize>
          <KeepAlive>true</KeepAlive>
        </SocketInputTransporterParameters>
      </Parameters>
    </Module>

     <Module type="formatter">
      <InstanceName>MyStreamingInputFormatter</InstanceName>
      <Name>StreamToStringFormatter</Name>
      <Next>MyCSVInputFormatter</Next>
      <Parameters>
        <StreamToStringFormatterParameters>
          <Delimiter>\n</Delimiter>
          <IncludeDelimiter>false</IncludeDelimiter>
```

```
          <AppendString></AppendString>
          <AppendPosition>front</AppendPosition>
          <IgnoreSpace>true</IgnoreSpace>
       </StreamToStringFormatterParameters>
      </Parameters>
    </Module>

    <Module type="formatter">
        <InstanceName>MyCSVInputFormatter</InstanceName>
        <Name>CsvStringToEspFormatter</Name>
        <Next>MyInStream_Publisher</Next>
        <Parallel>true</Parallel>
        <Parameters>
        </Parameters>
    </Module>

    <Module type="espconnector">
      <InstanceName>MyInStream_Publisher</InstanceName>
      <Name>EspPublisher</Name>
      <Parameters>
        <EspPublisherParameters>
          <ProjectName>EspProject1</ProjectName>
          <StreamName>BaseInput</StreamName>
          <MaxPubPoolSize>1</MaxPubPoolSize>
          <UseTransactions>false</UseTransactions>
          <SafeOps>true</SafeOps>
          <SkipDels>true</SkipDels>
        </EspPublisherParameters>
      </Parameters>
      <BufferMaxSize>10240</BufferMaxSize>
    </Module>

 </Modules>

  <EspProjects>
    <EspProject>
      <Name>EspProject1</Name>
     <Uri>esp://localhost:19011/sample_workspace/socket_csv_input</
Uri>
      <Security>
        <User></User>
        <Password encrypted="false"></Password>
        <AuthType>user_password</AuthType>
        <!--
          <RSAKeyStore>/keystore/keystore.jks</RSAKeyStore>
          <RSAKeyStorePassword>Sybase123</RSAKeyStorePassword>
        -->
        <!--
         <KerberosKDC>KDC</KerberosKDC>
         <KerberosRealm>REALM</KerberosRealm>
          <KerberosService>service/instance</KerberosService>
        <KerberosTicketCache>/tmp/krb5cc_user</KerberosTicketCache>
        -->
        <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
      </Security>
    </EspProject>
```

```
  </EspProjects>
  <GlobalParameters></GlobalParameters>
</Adapter>
```

Socket CSV Output Adapter Configuration

Configure the Socket CSV Output adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

ESPConnector Module: ESP Subscriber

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, espconnector.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	Section containing the **EspSubscriberParameters** parameter.

Parameter	Description
EspSubscriberParameters	(Required) Section containing parameters for the ESP subscriber.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Specifies the unique project tag of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream from which the adapter subscribes to data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.

Formatter Module: ESP to CSV String Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.

Parameter	Description
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **EspToCsvStringFormatterParameters** parameter.
EspToCsvStringFormatterParameters	(Required) Section containing parameters for the ESP to CSV String formatter parameters.
PrependStreamNameOpcode	Type: `boolean` (Optional) If set to true, the adapter prepends the stream name and the opcode in each row of data that is generated. The default value is false.
Delimiter	Type: `string` (Advanced) The symbols used to separate the column. The default value is a comma (,).
HasHeader	Type: `boolean` (Advanced) Determines whether the first line of the file contains the description of the fields. The default value is false.
DateFormat	Type: `string` (Advanced) The format string for date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.

Parameter	Description
TimestampFormat	Type: `string` (Advanced) Format string for timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

Formatter Module: String to Stream Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **StringToStreamFormatterParameters** parameter.
StringToStreamFormatterParameters	(Required) Section containing parameters for the String to Stream formatter parameters.
Delimiter	Type: `string` (Required) Specify the symbol used to separate columns. The default value is "\n".

Parameter	Description
IncludeDelimiter	Type: `boolean` (Required) If set to true, the delimiter is part of current row. If set to false, the delimiter is not part of the current row. The default value is false.
AppendString	Type: `string` (Required if **IncludeDelimiter** is set to true) If set to true, specify the string to append to the end of each result row. No default value.
AppendPosition	Type: `string` (Required if **IncludeDelimiter** is set to true) Specify the position to which the **AppendString** parameter takes effect. There are two valid values: front and end. The default value is front.
IgnoreSpace	Type: `boolean` (Required) Specify whether to ignore the space char. The default value is false.
CharsetName	Type: `string` (Advanced) Specify the name of a supported charset. The default value is US-ASCII.

Transporter Module: Socket Output Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **SocketOutTransporterParameters** parameter.
SocketOutputTransporterParameters	(Required) Section containing the Socket Output transporter parameters.
Host	Type: `string` (Required if **EpFile** is set to null) If the transporter is acting as a socket client, specify the socket server name. If the transporter is acting as a socket server, do not set this parameter. No default value.
Port	Type: `integer` (Required if **EpFile** is set to null) Specify the socket server port. If you set this to -1, the adapter reads from the ephemeral port file which is specified in the **EpFile** parameter. The default value is 12345.
EpFile	Type: `string` (Required if **Host** and **Port** are set to null) Specify the file that contains the socket server name/IP and port number. No default value.
Retryperiod	Type: `integer` (Advanced) When the transporter is acting as a socket server, this parameter designates the length of time to wait for the first incoming connection before switching to the continuous state. When the transporter is acting as a socket client, this parameter designates the time period for attempting to re-establish an outgoing connection, in seconds. The default value is 0.

Parameter	Description
KeepAlive	Type: `boolean` (Advanced) If set to true, the adapter disconnects from the socket server if there are no data transports for the duration of time specified in your router configuration. For example, if you set your router configuration to two hours and there are no messages during that time, the adapter disconnects from the socket server. The default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the Socket CSV Output adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.

Parameter	Description
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

Socket CSV Output Adapter Studio Properties

Adapter type: toolkit_socket_csv_output. Set these properties for the Socket CSV Output adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Column Delimiter	Property ID: **csvDelimiter** Type: `string` (Advanced) Specify the symbol used to separate the columns.
Date Format	Propterty ID: **csvDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.

Property Label	Description
Timestamp Format	Property ID: **csvTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.
Has Header	Property ID: **csvHasHeader** Type: `boolean` (Advanced) Indicate whether the first line of the file contains the description of the CSV fields.
Prepend stream name, opcode	Property ID: **csvPrependStreamNameOpcode** Type: `boolean` (Optional) If set to true, the adapter prepends the stream name and the opcode in each row of data that is generated.
Charset Name	Property ID: **charsetName** Type: `string` (Advanced) Specify the name of a supported charset.
Server	Property ID: **host** Type: `string` (Required) Server host name.
Port	Propterty ID: **port** Type: `int` (Required) Server port. To read from the Ephemeral Port File, set to -1.
Ephemeral Port File	Property ID: **epFile** Type: `filename` (Advanced) File that will contain the server port number if Port is -1.

Property Label	Description
Retry Period (seconds)	Property ID: **retryperiod** Type: `uint` (Advanced) Period for trying to re-establish an outgoing connection.
Enable TCP keepalive	Property ID: **keepAlive** Type: `boolean` (Advanced) Enable TCP keepalive on the socket connection.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: Socket CSV Output Adapter

Sample adapter configuration file for the Socket CSV Output adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
  <Name>socket_csv_output</Name>
  <Description>An adapter which transforms ESP data to csv format,
and output the data to socket server.</Description>
  <Log4jProperty>./log4j.properties</Log4jProperty>
  <Modules>
      <Module type="espconnector">
        <InstanceName>MyOutStream_Subscriber</InstanceName>
        <Name>EspSubscriber</Name>
        <Next>MyCSVOutputFormatter</Next>
        <Parameters>
            <EspSubscriberParameters>
                <ProjectName>EspProject2</ProjectName>
                <StreamName>BaseOutput</StreamName>
            </EspSubscriberParameters>
        </Parameters>
    </Module>
```

```
    <Module type="formatter">
        <InstanceName>MyCSVOutputFormatter</InstanceName>
        <Name>EspToCsvStringFormatter</Name>
        <Next>MyStreamingOutputFormatter</Next>
        <Parallel>true</Parallel>
        <Parameters>
        </Parameters>
    </Module>

    <Module type="formatter">
      <InstanceName>MyStreamingOutputFormatter</InstanceName>
      <Name>StringToStreamFormatter</Name>
      <Next>MySocketOutTransporter</Next>
      <Parameters>
        <StringToStreamFormatterParameters>
          <Delimiter>\n</Delimiter>
          <IncludeDelimiter>true</IncludeDelimiter>
          <AppendString>\n</AppendString>
          <AppendPosition>end</AppendPosition>
          <IgnoreSpace>true</IgnoreSpace>
         <CharsetName>US-ASCII</CharsetName>
       </StringToStreamFormatterParameters>
      </Parameters>
    </Module>

    <Module type="transporter">
      <InstanceName>MySocketOutTransporter</InstanceName>
      <Name>SocketOutputTransporter</Name>
      <Parameters>
        <SocketOutputTransporterParameters>
          <Host>localhost</Host>
          <Port>9996</Port>
          <EpFile></EpFile>
          <Retryperiod>10</Retryperiod>
          <KeepAlive>true</KeepAlive>
        </SocketOutputTransporterParameters>
      </Parameters>
    </Module>
  </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject2</Name>
            <Uri>esp://localhost:19011/sample_workspace/
socket_csv_output</Uri>
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Adapter Controller Parameters

The Adapter Controller port listens for commands. The `controller.xml` file is located in `%ESP_HOME%/adapters/framework/config` directory. This file is shared among all the adapter within the `%ESP_HOME%/adapters/framework/instances` directory.

Parameter	Description
ControlPort	(Required) Section containing the **MinPort** and **MaxPort** parameters.
MinPort	Type: `int` (Required) The minimum port number that can act as a control port number. The adapter framework allocates this number automatically. A recommended value is 19082.
MaxPort	Type: `int` (Optional) The maximum port number that can act as a control port number. The adapter framework allocates this number automatically. The default value is 65535.
Statistics_Update_Interval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework publishes statistics data to the ESP project to which the adapter is connected. A recommended value is 5000.
MonitorInterval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework checks the status of internal resource allocation. The framework logs a message in the log file if a resource is about to be exhausted. A recommended value is 3000.

Logging

The Socket CSV Input and Output adapter use the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing

the logging configuration is located in the `%ESP_HOME%\adapters\framework\config` directory.

Specify the location of the logging file you wish to use in the **Log4jProperty** parameter within the adapter configuration file. You can modify the logging levels within this file or the `$ESP_HOME/adapters/framework/config/log4j.properties`, which is used by default. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
```

```
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Starting the Socket CSV Adapter

To start the Socket CSV adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

   ```
   cd %ESP_HOME%\cluster\nodes\node1
     %ESP_HOME%\bin\esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   %ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

   UNIX:

a. Start the example cluster.

```
cd $ESP_HOME/cluster/nodes/node1
  $ESP_HOME/bin/esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
$ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

2. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: For the Socket CSV Input adapter: `cd $ESP_HOME/adapters/framework/instances/socket_csv_input` For the Socket CSV Output adapter: `cd $ESP_HOME/adapters/framework/instances/socket_csv_output` `./start_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the Socket CSV Input adapter:`cd %ESP_HOME%/adapters/framework/instances/socket_csv_input` For the Socket CSV Output adapter:`cd %ESP_HOME%/adapters/framework/instances/socket_csv_output` `start_adapter.bat <adapter configuration file name>`

Stopping the Socket CSV Adapter

To stop the Socket CSV adapter from the command line, execute the **stop** command.

When you are running the adapter from the command line, stop the adapter first before stopping the project.

Operating System	Step
UNIX	Open a terminal window and enter: For the Socket CSV Input adapter: `cd $ESP_HOME/adapters/framework/instances/socket_csv_input` For the Socket CSV Output adapter: `cd $ESP_HOME/adapters/framework/instances/socket_csv_output` `./stop_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the Socket CSV Input adapter: `cd %ESP_HOME%/adapters/framework/instances/socket_csv_input` For the Socket CSV Output adapter: `cd %ESP_HOME%/adapters/framework/instances/socket_csv_output` `stop_adapter.bat <adapter configuration file name>`

Socket FIX Input Adapter

Adapter type: fixsocket_in. The Socket FIX Input adapter reads FIX messages from a TCP server socket and writes them as stream records.

Each stream hosts FIX messages of a certain type. The adapter discards messages of any other FIX type, and stores FIX fields in the same order in stream columns. The following fields are exceptions to this:

- BeginString
- BodyLength
- MsgType
- CheckSum

Ensure that the names of the stream columns correspond to the FIX protocol specification.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
FIX Version	Property ID: **fixVersion** Type: `choice` (Required) Version of the FIX protocol. Default value is 4.2.
FIX Message Type	Property ID: **fixMessageType** Type: `string` (Required) Type of messages hosted by the stream. No default value.
Source Host	Property ID: **fixHost** Type: `string` (Required) Name or IP address of source server for FIX messages. Default value is localhost.
Source Port	Property ID: **fixPort** Type: `uint` (Required) Port on which FIX messages are available on. Default value is 12345.
Reconnect Interval	Property ID: **reconnectInterval** Type: `uint` (Required) Reconnect interval, in seconds. If zero, makes no attempt to reconnect. Default value is 10.
Maximum Reconnect Attempts	Property ID: **maxReconnectAttempts** Type: `uint` (Required) Maximum number of reconnect attempts. Default value is zero.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Date format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.

Property Label	Description
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Timestamp format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Known limitations:

- This adapter is not a full FIX Engine.
- Supports only FIX versions 4.2, 4.3, 4.4, and 5.0.
- Does not support repeating groups and components.
- Supports only INSERT opcode.

See also

- *FIX Input Adapter* on page 162

Datatype Mapping for the Socket FIX Input Adapter

Event Stream Processor datatypes map to FIX datatypes.

Event Stream Processor Datatype	QuickFix Datatype
`integer`	`boolean`
`string`	`byte[]`
`string`	`char`
`string`	`string`
`date`	`date`

Event Stream Processor Datatype	QuickFix Datatype
`float`	`float`
`integer`	`integer`
`date` or `timestamp`	`UTCDateOnly`
`date` or `timestamp`	`UTCTimeOnly`
`date` or `timestamp`	`UTCTimeStamp`

Socket FIX Output Adapter

Adapter type: fixsocket_out. The Socket FIX Output adapter writes stream data as FIX messages to a TCP server socket.

Each stream hosts FIX messages of a certain type. The adapter sends messages contiguously, with no line feeds. It generates the following FIX fields:

- BeginString
- BodyLength
- MsgType
- CheckSum

Ensure that the rest of the fields are stored in the appropriate order in stream columns, and that the names of the stream columns correspond to the FIX protocol specification.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
FIX Version	Property ID: **fixVersion** Type: `choice` (Required) Version of the FIX protocol. Default value is 4.2.
FIX Message Type	Property ID: **fixMessageType** Type: `string` (Required) Type of messages hosted by the stream. No default value.

Property Label	Description
Destination Host	Property ID: **fixHost** Type: `string` (Required) Name or IP address of destination server for FIX messages. Default value is localhost.
Destination Port	Property ID: **fixPort** Type: `uint` (Required) Port on which the destination server socket is listening to FIX messages. Default value is 12346.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Date format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Timestamp format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Known limitations:

- This adapter is not a full FIX Engine.
- Supports only FIX versions 4.2, 4.3, 4.4, and 5.0.
- Does not support repeating groups and components.
- Does not attempt to reconnect if the connection to the FIX server is lost.
- Supports only INSERT opcode.

See also

- *FIX Input Adapter* on page 162

Datatype Mapping for the Socket FIX Output Adapter

Event Stream Processor datatypes map to FIX datatypes.

Event Stream Processor Datatype	QuickFix Datatype
`integer`	`boolean`
`string`	`byte[]`
`string`	`char`
`string`	`string`
`date`	`date`
`float`	`float`
`integer`	`integer`
`date` or `timestamp`	`UTCDateOnly`
`date` or `timestamp`	`UTCTimeOnly`
`date` or `timestamp`	`UTCTimeStamp`

Socket JSON Input and Output Adapter

The Socket JSON Input adapter obtains streaming data from the socket server, formats data into JSON format and inputs it into Event Stream Processor. The Socket JSON Output adapter takes JSON data from Event Stream Processor, formats it to bytebuffer, and transports it to the socket server in streaming format.

Socket JSON Input Adapter Configuration

Configure the Socket JSON Input adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

Transporter Module: Socket Input Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **SocketInputTransporterParameters** element.
SocketInputTransporterParameters	(Required) Section containing parameters for the Socket Input transporter.

Parameter	Description
Host	Type: `string` (Required if **EpFile** is set to null) If the transporter is acting as a socket client, specify the socket server name. If the transporter is acting as a socket server, do not set this parameter. No default value.
Port	Type: `integer` (Required if **EpFile** is set to null) Specify the socket server port. If you set this to -1, the adapter reads from the ephemeral port file which is specified in the **EpFile** parameter. The default value is 12345.
EpFile	Type: `string` (Required if **Host** and **Port** are set to null) Specify the file that contains the socket server name/IP and port number. No default value.
Retryperiod	Type: `integer` (Advanced) When the transporter is acting as a socket server, this parameter designates the length of time to wait for the first incoming connection before switching to the continuous state. When the transporter is acting as a socket client, this parameter designates the time period for attempting to re-establish an outgoing connection, in seconds. The default value is 0.
BlockSize	Type: `integer` (Advanced) Define the size of the data block when transporting data from the socket server to the socket client. The default value is 1024.

Parameter	Description
KeepAlive	Type: `boolean` (Advanced) If set to true, the adapter disconnects from the socket server if there are no data transports for the duration of time specified in your router configuration. For example, if you set your router configuration to two hours and there are no messages during that time, the adapter disconnects from the socket server. The default value is false.

Formatter Module: Streaming JSON to JSON String Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **JsonStreamToJsonStringFormatterParameters** parameter.
JsonStreamToJsonStringFormatterParameters	(Required) Section containing parameters for the Streaming JSON to JSON String formatter.

Parameter	Description
CharsetName	Type: `string` (Optional) Specify the name of a supported charset. The default value is US-ASCII.

Formatter Module: JSON String to ESP Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **JsonStringToEspFormatterParameters** parameter.
JsonStringToEspFormatterParameters	(Required) Section containing the JSON String to ESP formatter parameters.
ColumnMappings	(Required) Section containing the **ColsMapping** parameter.
ColsMapping	(Required) Section containing the **Column** parameter.

Parameter	Description
Column	Type: `string` (Required) Specify a value for the JSON data that you wish to map to ESP columns. This value is matched by a pattern path expression. For example, [<Column>JSONPath expression</Column>]. The first <Column/> is mapped to the first column of an ESP row, the second <Column/> is mapped to the second column of an ESP row, and so on. This parameter has two attributes: • **streamname –** Specify which stream to publish. • **rootpath –** Specify a rootpath for the JSON data. You can use one rootpath to publish more than one ESP row from one JSON data record.
DateFormat	Type: `string` (Advanced) The format string for parsing date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.
TimestampFormat	Type: `string` (Advanced) Format string for parsing timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

ESP Connector Module: ESP Publisher

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.

Parameter	Description
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **EspPublisherParameters** parameter.
EspPublisherParameters	(Required) Section containing parameters for the ESP publisher.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.

Parameter	Description
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream to which the adapter publishes data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
MaxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.
MaxPubPoolTime	Type: `positive integer` (Optional) Specifies the maximum period of time, in milliseconds, for which records are pooled before being published. If not set, pooling time is unlimited and the pooling strategy is governed by **maxPubPoolSize**. No default value.
UseTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.
SafeOps	Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT, and converts DELETE to SAFEDELETE. The default value is false.

Parameter	Description
SkipDels	Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. The default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the Socket JSON Input adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.

Parameter	Description
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

Socket JSON Input Adapter Studio Properties

Adapter type: toolkit_socket_json_input. Set these properties for the Socket JSON Input adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
JSON Column Mapping List	Property ID: **jsonColsMappingList** Type: `string` (Required) The JSON expression list is separated by commas. The first separated part is mapped to the first column of an ESP row, the second separated part is mapped to the second column of an ESP row, and so on.

Property Label	Description
JSON Root Path	Property ID: **jsonRootpath** Type: `string` (Required) Specify a root path for the JSON data.
Date Format	Propterty ID: **jsonDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **jsonTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.
Server	Property ID: **host** Type: `string` (Required) Server host name.
Port	Propterty ID: **port** Type: `int` (Required) Server port. To read from the Ephemeral Port File, set to -1.
Ephemeral Port File	Property ID: **epFile** Type: `filename` (Advanced) File that will contain the server port number if Port is -1.
Retry Period (seconds)	Property ID: **retryperiod** Type: `uint` (Advanced) Period for trying to re-establish an outgoing connection.
Buffer Size	Property ID: **inputBufferSize** Type: `uint` (Advanced) The buffer size of the socket connection (bytes).

Property Label	Description
Enable TCP keepalive	Property ID: **keepAlive** Type: `boolean` (Advanced) Enable TCP keepalive on the socket connection.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: Socket JSON Input Adapter

Sample adapter configuration file for the Socket JSON Input adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
    <Name>socket_json_input</Name>
    <Description>An adapter which receives JSON message from socket
server,transforms to ESP data format, and publishes to ESP stream.</
Description>
    <Log4jProperty>./log4j.properties</Log4jProperty>
    <Modules>
        <Module type="transporter">
           <InstanceName>MyExampleSocketInTransporter</InstanceName>
            <Name>SocketInputTransporter</Name>
            <Next>MyJsonStreamToJsonStringFormatter</Next>
            <Parameters>
                <SocketInputTransporterParameters>
                    <Host>localhost</Host>
                    <Port>9998</Port>
                    <EpFile></EpFile>
                    <Retryperiod>60</Retryperiod>
                    <Blocksize>512</Blocksize>
                    <KeepAlive>true</KeepAlive>
                </SocketInputTransporterParameters>
            </Parameters>
        </Module>

        <Module type="formatter">
```

```
            <InstanceName>MyJsonStreamToJsonStringFormatter</
InstanceName>
            <Name>JsonStreamToJsonStringFormatter</Name>
            <Next>MyJsonInFormatter</Next>
            <Parameters />
        </Module>

        <Module type="formatter">
            <InstanceName>MyJsonInFormatter</InstanceName>
            <Name>JsonStringToEspFormatter</Name>
            <Next>MyInStream_Publisher</Next>
            <Parameters>
                <JsonStringToEspFormatterParameters>
                    <DateFormat>yyyy-MM-dd HH:mm:ss.SSS</DateFormat>
                    <TimestampFormat>yyyy/MM/dd HH:mm:ss</
TimestampFormat>
                    <ColumnMappings>
                        <ColsMapping streamname="EntityStream"
rootpath="entities">
                            <Column>display_text</Column>
                            <Column>domain_role</Column>
                            <Column>offset</Column>
                            <Column>length</Column>
                        </ColsMapping>
                    </ColumnMappings>
                </JsonStringToEspFormatterParameters>
            </Parameters>
        </Module>

        <Module type="espconnector">
            <InstanceName>MyInStream_Publisher</InstanceName>
            <Name>EspPublisher</Name>
            <Parameters>
                <EspPublisherParameters>
                    <ProjectName>EspProject1</ProjectName>
                    <StreamName>EntityStream</StreamName>
                    <MaxPubPoolSize>1</MaxPubPoolSize>
                    <UseTransactions>false</UseTransactions>
                    <SafeOps>true</SafeOps>
                    <SkipDels>true</SkipDels>
                </EspPublisherParameters>
            </Parameters>
        </Module>

    </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject1</Name>
            <Uri>esp://localhost:19011/sample_workspace/
socket_json_input</Uri>
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
             <!-- <RSAKeyStore>/keystore/keystore.jks</RSAKeyStore>
```

```
<RSAKeyStorePassword>Sybase123</RSAKeyStorePassword> -->
                <!-- <KerberosKDC>KDC</KerberosKDC>
<KerberosRealm>REALM</KerberosRealm>
                 <KerberosService>service/instance</KerberosService>
<KerberosTicketCache>/tmp/krb5cc_user</KerberosTicketCache> -->
                <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Socket JSON Output Adapter Configuration

Configure the Socket JSON Output adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

ESPConnector Module: ESP Subscriber

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, espconnector.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.

Parameter	Description
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	Section containing the **EspSubscriberParameters** parameter.
EspSubscriberParameters	(Required) Section containing parameters for the ESP subscriber.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Specifies the unique project tag of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream from which the adapter subscribes to data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.

Formatter Module: ESP to JSON Stream Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **EsptoJsonStreamFormatterParameters** parameter.
EsptoJsonStreamFormatterParameters	(Required) Section containing the ESP to JSON Stream formatter parameters.
ColsMapping	Type: `complextype` (Required) Specify which columns of an ESP row you wish to map to JSON data. These values are matched by a pattern path expression. For example, [<Column>JSONPath expression</Column>]+. The first <Column/> is mapped to the first column of an ESP row, the second <Column/> is mapped to the second column of an ESP row, and so on.
DateFormat	Type: `string` (Advanced) The format string for date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.

Parameter	Description
TimestampFormat	Type: `string` (Advanced) Format string for timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

Transporter Module: Socket Output Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **SocketOutTransporterParameters** parameter.
SocketOutputTransporterParameters	(Required) Section containing the Socket Output transporter parameters.
Host	Type: `string` (Required if **EpFile** is set to null) If the transporter is acting as a socket client, specify the socket server name. If the transporter is acting as a socket server, do not set this parameter. No default value.
Port	Type: `integer` (Required if **EpFile** is set to null) Specify the socket server port. If you set this to -1, the adapter reads from the ephemeral port file which is specified in the **EpFile** parameter. The default value is 12345.

Parameter	Description
EpFile	Type: `string` (Required if **Host** and **Port** are set to null) Specify the file that contains the socket server name/IP and port number. No default value.
Retryperiod	Type: `integer` (Advanced) When the transporter is acting as a socket server, this parameter designates the length of time to wait for the first incoming connection before switching to the continuous state. When the transporter is acting as a socket client, this parameter designates the time period for attempting to re-establish an outgoing connection, in seconds. The default value is 0.
KeepAlive	Type: `boolean` (Advanced) If set to true, the adapter disconnects from the socket server if there are no data transports for the duration of time specified in your router configuration. For example, if you set your router configuration to two hours and there are no messages during that time, the adapter disconnects from the socket server. The default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the Socket JSON Output adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.

Parameter	Description
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.

Parameter	Description
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using key-store • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

Socket JSON Output Adapter Studio Properties

Adapter type: toolkit_socket_json_output. Set these properties for the Socket JSON Output adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
JSON Column Mapping List	Property ID: **jsonColsMappingList** Type: `string` (Required) The JSON expression list is separated by commas. The first separated part is mapped to the first column of an ESP row, the second separated part is mapped to the second column of an ESP row, and so on.
Date Format	Propterty ID: **jsonDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.

Property Label	Description
Timestamp Format	Property ID: **jsonTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.
Charset Name	Property ID: **charsetName** Type: `string` (Advanced) Specify the name of a supported charset.
Server	Property ID: **host** Type: `string` (Required) Server host name.
Port	Propterty ID: **port** Type: `int` (Required) Server port. To read from the Ephemeral Port File, set to -1.
Ephemeral Port File	Property ID: **epFile** Type: `filename` (Advanced) File that will contain the server port number if Port is -1.
Retry Period (seconds)	Property ID: **retryperiod** Type: `uint` (Advanced) Period for trying to re-establish an outgoing connection.
Enable TCP keepalive	Property ID: **keepAlive** Type: `boolean` (Advanced) Enable TCP keepalive on the socket connection.

Property Label	Description
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: Socket JSON Output Adapter

Sample adapter configuration file for the Socket JSON Output adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
  <Name>socket_json_output</Name>
  <Description>An adapter which transforms ESP data to JSON format
and sends out the data to socket interface.</Description>
  <Log4jProperty>./log4j.properties</Log4jProperty>
  <Modules>
    <Module type="espconnector">
      <InstanceName>MyOutStream_Subscriber</InstanceName>
      <Name>EspSubscriber</Name>
      <Next>MyJsonOutFormatter</Next>
      <Parameters>
        <EspSubscriberParameters>
          <ProjectName>EspProject1</ProjectName>
          <StreamName>EntityStream</StreamName>
        </EspSubscriberParameters>
      </Parameters>
    </Module>

    <Module type="formatter">
      <InstanceName>MyJsonOutFormatter</InstanceName>
      <Name>EspToJsonStringFormatter</Name>
      <Next>MyStringToStreamFormatter</Next>
      <Parameters>
        <EspToJsonStringFormatterParameters>
          <DateFormat>yyyy-MM-dd HH:mm:ss.SSS</DateFormat>
          <TimestampFormat>yyyy/MM/dd HH:mm:ss</TimestampFormat>
          <ColsMapping>
            <Column>published_at</Column>
            <Column>title</Column>
            <Column>lang</Column>
          </ColsMapping>
```

```
        </EspToJsonStringFormatterParameters>
      </Parameters>
    </Module>

     <Module type="formatter">
      <InstanceName>MyStringToStreamFormatter</InstanceName>
      <Name>StringToStreamFormatter</Name>
      <Next>MySocketOutTransporter</Next>
      <Parameters>
        <StringToStreamFormatterParameters>
          <Delimiter>\n</Delimiter>
          <IncludeDelimiter>true</IncludeDelimiter>
          <AppendString>\n</AppendString>
          <AppendPosition>end</AppendPosition>
          <IgnoreSpace>true</IgnoreSpace>
         <CharsetName>US-ASCII</CharsetName>
       </StringToStreamFormatterParameters>
      </Parameters>
    </Module>

    <Module type="transporter">
      <InstanceName>MySocketOutTransporter</InstanceName>
      <Name>SocketOutputTransporter</Name>
      <Parameters>
        <SocketOutputTransporterParameters>
          <Host>localhost</Host>
          <Port>9996</Port>
          <EpFile></EpFile>
          <Retryperiod>10</Retryperiod>
          <KeepAlive>true</KeepAlive>
        </SocketOutputTransporterParameters>
      </Parameters>
    </Module>

  </Modules>

  <EspProjects>
    <EspProject>
      <Name>EspProject1</Name>
      <Uri>esp://localhost:19011/sample_workspace/
socket_json_output</Uri>
      <Security>
        <User></User>
        <Password encrypted="false"></Password>
        <AuthType>user_password</AuthType>
        <!--
          <RSAKeyStore>/keystore/keystore.jks</RSAKeyStore>
          <RSAKeyStorePassword>Sybase123</RSAKeyStorePassword>
        -->
        <!--
         <KerberosKDC>KDC</KerberosKDC>
         <KerberosRealm>REALM</KerberosRealm>
          <KerberosService>service/instance</KerberosService>
        <KerberosTicketCache>/tmp/krb5cc_user</KerberosTicketCache>
        -->
        <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
```

```
      </Security>
    </EspProject>
  </EspProjects>
  <GlobalParameters></GlobalParameters>
</Adapter>
```

Adapter Controller Parameters

The Adapter Controller port listens for commands. The `controller.xml` file is located in `%ESP_HOME%/adapters/framework/config` directory. This file is shared among all the adapter within the `%ESP_HOME%/adapters/framework/instances` directory.

Parameter	Description
ControlPort	(Required) Section containing the **MinPort** and **MaxPort** parameters.
MinPort	Type: `int` (Required) The minimum port number that can act as a control port number. The adapter framework allocates this number automatically. A recommended value is 19082.
MaxPort	Type: `int` (Optional) The maximum port number that can act as a control port number. The adapter framework allocates this number automatically. The default value is 65535.
Statistics_Update_Interval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework publishes statistics data to the ESP project to which the adapter is connected. A recommended value is 5000.
MonitorInterval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework checks the status of internal resource allocation. The framework logs a message in the log file if a resource is about to be exhausted. A recommended value is 3000.

Logging

The Socket JSON Input and Output adapter use the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing the logging configuration is located in the `%ESP_HOME%\adapters\framework\config` directory.

Specify the location of the logging file you wish to use in the **Log4jProperty** parameter within the adapter configuration file. You can modify the logging levels within this file or the `$ESP_HOME/adapters/framework/config/log4j.properties`, which is used by default. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
```

```
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Starting the Socket JSON Adapter

To start the Socket JSON adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

   ```
   cd %ESP_HOME%\cluster\nodes\node1
     %ESP_HOME%\bin\esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   %ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

```
%ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

UNIX:

a. Start the example cluster.

```
cd $ESP_HOME/cluster/nodes/node1
  $ESP_HOME/bin/esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
$ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

2. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: For the Socket JSON Input adapter: `cd $ESP_HOME/adapters/framework/instances/socket_json_input` For the Socket JSON Output adapter: `cd $ESP_HOME/adapters/framework/instances/socket_json_output` `./start_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the Socket JSON Input adapter:`cd %ESP_HOME%/adapters/framework/instances/socket_json_input` For the Socket JSON Output adapter:`cd %ESP_HOME%/adapters/framework/instances/socket_json_output` `start_adapter.bat <adapter configuration file name>`

Stopping the Socket JSON Adapter

To stop the Socket JSON adapter from the command line, execute the **stop** command.

When you are running the adapter from the command line, stop the adapter first before stopping the project.

Operating System	Step
UNIX	Open a terminal window and enter: For the Socket JSON Input adapter: `cd $ESP_HOME/adapters/framework/instances/socket_json_input` For the Socket JSON Output adapter: `cd $ESP_HOME/adapters/framework/instances/socket_json_output` `./stop_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the Socket JSON Input adapter: `cd %ESP_HOME%/adapters/framework/instances/socket_json_input` For the Socket JSON Output adapter: `cd %ESP_HOME%/adapters/framework/instances/socket_json_output` `stop_adapter.bat <adapter configuration file name>`

Socket XML Input and Output Adapter

The Socket XML Input adapter obtains XML data from a Socket server and publishes it to Event Stream Processor. The Socket XML Output adapter takes data from Event Stream Processor, formats it to XML list format, and outputs it to a Socket server.

Socket XML Input Adapter Configuration

Configure the Socket XML Input adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

Transporter Module: Socket Input Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **SocketInputTransporterParameters** element.
SocketInputTransporterParameters	(Required) Section containing parameters for the Socket Input transporter.

Parameter	Description
Host	Type: `string` (Required if **EpFile** is set to null) If the transporter is acting as a socket client, specify the socket server name. If the transporter is acting as a socket server, do not set this parameter. No default value.
Port	Type: `integer` (Required if **EpFile** is set to null) Specify the socket server port. If you set this to -1, the adapter reads from the ephemeral port file which is specified in the **EpFile** parameter. The default value is 12345.
EpFile	Type: `string` (Required if **Host** and **Port** are set to null) Specify the file that contains the socket server name/IP and port number. No default value.
Retryperiod	Type: `integer` (Advanced) When the transporter is acting as a socket server, this parameter designates the length of time to wait for the first incoming connection before switching to the continuous state. When the transporter is acting as a socket client, this parameter designates the time period for attempting to re-establish an outgoing connection, in seconds. The default value is 0.
BlockSize	Type: `integer` (Advanced) Define the size of the data block when transporting data from the socket server to the socket client. The default value is 1024.

Parameter	Description
KeepAlive	Type: `boolean` (Advanced) If set to true, the adapter disconnects from the socket server if there are no data transports for the duration of time specified in your router configuration. For example, if you set your router configuration to two hours and there are no messages during that time, the adapter disconnects from the socket server. The default value is false.

Formatter Module: Stream to String Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **StreamToStringFormatterParameters** parameter.
StreamToStringFormatterParameters	(Required) Section containing the Stream to String formatter parameters.

Parameter	Description
Delimiter	Type: `string` (Required) Specify the symbol used to separate columns. The default value is "\n".
IncludeDelimiter	Type: `boolean` (Required) If set to true, the delimiter is part of current row. If set to false, the delimiter is not part of the current row. The default value is false.
AppendString	Type: `string` (Required if **IncludeDelimiter** is set to true) If set to true, specify the string to append to the end of each result row. No default value.
AppendPosition	Type: `string` (Required if **IncludeDelimiter** is set to true) Specify the position to which the **AppendString** parameter takes effect. There are two valid values: front and end. The default value is front.
IgnoreSpace	Type: `boolean` (Required) Specify whether to trim the space char. The default value is true.

Formatter Module: XML String to ESP Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.

Parameter	Description
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **XmlString-ToEspFormatterParameters** parameter.
XmlStringToEspFormatterParameters	(Required) Section containing the XML String to ESP formatter parameters.
DateFormat	Type: `string` (Optional) The format string for parsing date values. For example, `yyyy-MM-dd'T'HH:mm:ss`.
TimestampFormat	Type: `string` (Optional) Format string for parsing timestamp values. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

ESP Connector Module: ESP Publisher

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.

Parameter	Description
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **EspPublisherParameters** parameter.
EspPublisherParameters	(Required) Section containing parameters for the ESP publisher.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.

Parameter	Description
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream to which the adapter publishes data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
MaxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.
MaxPubPoolTime	Type: `positive integer` (Optional) Specifies the maximum period of time, in milliseconds, for which records are pooled before being published. If not set, pooling time is unlimited and the pooling strategy is governed by **maxPubPoolSize**. No default value.
UseTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.
SafeOps	Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT, and converts DELETE to SAFEDELETE. The default value is false.

Parameter	Description
SkipDels	Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. The default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the Socket XML Input adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.

Parameter	Description
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

Socket XML Input Adapter Studio Properties

Adapter type: toolkit_socket_xmllist_input. Set these properties for the Socket XML Input adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Date Format	Propterty ID: **xmllistDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **xmllistTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.

Property Label	Description
Charset Name	Property ID: **charsetName** Type: `string` (Advanced) Specify the name of a supported charset.
Server	Property ID: **host** Type: `string` (Required) Server host name.
Port	Propterty ID: **port** Type: `int` (Required) Server port. To read from the Ephemeral Port File, set to -1.
Ephemeral Port File	Property ID: **epFile** Type: `filename` (Advanced) File that will contain the server port number if Port is -1.
Retry Period (seconds)	Property ID: **retryperiod** Type: `uint` (Advanced) Period for trying to re-establish an outgoing connection.
Buffer Size	Property ID: **inputBufferSize** Type: `uint` (Advanced) The buffer size of the socket connection (bytes).
Enable TCP keepalive	Property ID: **keepAlive** Type: `boolean` (Advanced) Enable TCP keepalive on the socket connection.

Property Label	Description
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: Socket XML Input Adapter

Sample adapter configuration file for the Socket XML Input adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
  <Name>socket_xmllist_input</Name>
  <Description>An adapter which gets xml list data from socket
server, transforms to ESP data format, and publishes to ESP stream.</
Description>
  <Log4jProperty>./log4j.properties</Log4jProperty>
  <Modules>
    <Module type="transporter">
      <InstanceName>MyExampleSocketInTransporter</InstanceName>
      <Name>SocketInputTransporter</Name>
      <Next>MyStreamingInputFormatter</Next>
      <Parameters>
        <SocketInputTransporterParameters>
          <Host>localhost</Host>
          <Port>9998</Port>
          <EpFile></EpFile>
          <Retryperiod>60</Retryperiod>
          <Blocksize>512</Blocksize>
          <KeepAlive>true</KeepAlive>
        </SocketInputTransporterParameters>
      </Parameters>
    </Module>

     <Module type="formatter">
      <InstanceName>MyStreamingInputFormatter</InstanceName>
      <Name>StreamToStringFormatter</Name>
      <Next>MyXmlListInputFormatter</Next>
      <Parameters>
        <StreamToStringFormatterParameters>
          <Delimiter><![CDATA[<BaseInput]]></Delimiter>
          <IncludeDelimiter>true</IncludeDelimiter>
```

```
          <AppendString><![CDATA[<BaseInput]]></AppendString>
          <AppendPosition>front</AppendPosition>
          <IgnoreSpace>true</IgnoreSpace>
       </StreamToStringFormatterParameters>
      </Parameters>
    </Module>

    <Module type="formatter">
      <InstanceName>MyXmlListInputFormatter</InstanceName>
      <Name>XmlStringToEspFormatter</Name>
      <Next>MyInStream_Publisher</Next>
      <Parallel>true</Parallel>
      <Parameters>
      </Parameters>
    </Module>


    <Module type="espconnector">
      <InstanceName>MyInStream_Publisher</InstanceName>
      <Name>EspPublisher</Name>
      <Parameters>
        <EspPublisherParameters>
          <ProjectName>EspProject1</ProjectName>
          <StreamName>BaseInput</StreamName>
          <MaxPubPoolSize>1</MaxPubPoolSize>
          <UseTransactions>false</UseTransactions>
          <SafeOps>true</SafeOps>
          <SkipDels>true</SkipDels>
        </EspPublisherParameters>
      </Parameters>
      <BufferMaxSize>10240</BufferMaxSize>
    </Module>

 </Modules>

  <EspProjects>
    <EspProject>
      <Name>EspProject1</Name>
      <Uri>esp://localhost:19011/sample_workspace/
socket_xmllist_input</Uri>
      <Security>
        <User></User>
        <Password encrypted="false"></Password>
        <AuthType>user_password</AuthType>
        <!--
          <RSAKeyStore>/keystore/keystore.jks</RSAKeyStore>
          <RSAKeyStorePassword>Sybase123</RSAKeyStorePassword>
        -->
        <!--
         <KerberosKDC>KDC</KerberosKDC>
         <KerberosRealm>REALM</KerberosRealm>
          <KerberosService>service/instance</KerberosService>
         <KerberosTicketCache>/tmp/krb5cc_user</KerberosTicketCache>
        -->
        <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
      </Security>
```

```
    </EspProject>
  </EspProjects>
  <GlobalParameters></GlobalParameters>
</Adapter>
```

Socket XML Output Adapter Configuration

Configure the Socket XML Output adapter by specifying values for the ESP connector, formatter, and transporter modules in the adapter configuration file.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

ESPConnector Module: ESP Subscriber

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, espconnector.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.

Parameter	Description
Parameters	Section containing the **EspSubscriberParameters** parameter.
EspSubscriberParameters	(Required) Section containing parameters for the ESP subscriber.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Specifies the unique project tag of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream from which the adapter subscribes to data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.

Formatter Module: ESP to XML String Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.

Parameter	Description
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parallel	Type: `boolean` (Optional) If set to true, the module runs as a separated thread. The default value is true.
Parameters	(Required) Section containing the **EspToXmlStringFormatterParameters** parameter.
EspToXmlStringFormatterParameters	(Required) Section containing the ESP to XML String formatter parameters.
DateFormat	Type: `string` (Optional) The format string for date values from an ESP project. For example, `yyyy-MM-dd'T'HH:mm:ss.`
TimestampFormat	Type: `string` (Optional) Format string for timestamp values from an ESP project. For example, `yyyy-MM-dd'T'HH:mm:ss.SSS`.

Formatter Module: String to Stream Formatter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **StringToStreamFormatterParameters** parameter.
StringToStreamFormatterParameters	(Required) Section containing parameters for the String to Stream formatter parameters.
Delimiter	Type: `string` (Required) Specify the symbol used to separate columns. The default value is "\n".
IncludeDelimiter	Type: `boolean` (Required) If set to true, the delimiter is part of current row. If set to false, the delimiter is not part of the current row. The default value is false.
AppendString	Type: `string` (Required if **IncludeDelimiter** is set to true) If set to true, specify the string to append to the end of each result row. No default value.

Parameter	Description
AppendPosition	Type: `string` (Required if **IncludeDelimiter** is set to true) Specify the position to which the **AppendString** parameter takes effect. There are two valid values: front and end. The default value is front.
IgnoreSpace	Type: `boolean` (Required) Specify whether to ignore the space char. The default value is false.
CharsetName	Type: `string` (Advanced) Specify the name of a supported charset. The default value is US-ASCII.

Transporter Module: Socket Output Transporter

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **SocketOutputTransporterParameters** parameter.
SocketOutputTransporterParameters	(Required) Section containing the Socket Output transporter parameters.

Parameter	Description
Host	Type: `string` (Required if **EpFile** is set to null) If the transporter is acting as a socket client, specify the socket server name. If the transporter is acting as a socket server, do not set this parameter. No default value.
Port	Type: `integer` (Required if **EpFile** is set to null) Specify the socket server port. If you set this to -1, the adapter reads from the ephemeral port file which is specified in the **EpFile** parameter. The default value is 12345.
EpFile	Type: `string` (Required if **Host** and **Port** are set to null) Specify the file that contains the socket server name/IP and port number. No default value.
Retryperiod	Type: `integer` (Advanced) When the transporter is acting as a socket server, this parameter designates the length of time to wait for the first incoming connection before switching to the continuous state. When the transporter is acting as a socket client, this parameter designates the time period for attempting to re-establish an outgoing connection, in seconds. The default value is 0.
KeepAlive	Type: `boolean` (Advanced) If set to true, the adapter disconnects from the socket server if there are no data transports for the duration of time specified in your router configuration. For example, if you set your router configuration to two hours and there are no messages during that time, the adapter disconnects from the socket server. The default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the Socket XML Output adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.

Parameter	Description
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using key-store • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

Socket XML Output Adapter Studio Properties

Adapter type: toolkit_socket_xmllist_output. Set these properties for the Socket XML Output adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Date Format	Propterty ID: **xmllistDateFormat** Type: `string` (Advanced) Specify the format for parsing date values.
Timestamp Format	Property ID: **xmllistTimestampFormat** Type: `string` (Advanced) Specify the format for parsing timestamp values.
Charset Name	Property ID: **charsetName** Type: `string` (Advanced) Specify the name of a supported charset.

Property Label	Description
Server	Property ID: **host** Type: `string` (Required) Server host name.
Port	Propterty ID: **port** Type: `int` (Required) Server port. To read from the Ephemeral Port File, set to -1.
Ephemeral Port File	Property ID: **epFile** Type: `filename` (Advanced) File that will contain the server port number if Port is -1.
Retry Period (seconds)	Property ID: **retryperiod** Type: `uint` (Advanced) Period for trying to re-establish an outgoing connection.
Enable TCP keepalive	Property ID: **keepAlive** Type: `boolean` (Advanced) Enable TCP keepalive on the socket connection.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Sample Configuration File: Socket XML Output Adapter

Sample adapter configuration file for the Socket XML Output adapter.

```
<?xml version="1.0" encoding="utf-8"?>
<Adapter>
  <Name>socket_xmllist_output</Name>
  <Description>An adapter which transforms ESP data to xml list
format, and output the data to socket server.</Description>
  <Log4jProperty>./log4j.properties</Log4jProperty>
  <Modules>
      <Module type="espconnector">
        <InstanceName>MyOutStream_Subscriber</InstanceName>
        <Name>EspSubscriber</Name>
        <Next>XmlListOutputFormatter</Next>
        <Parameters>
            <EspSubscriberParameters>
                <ProjectName>EspProject2</ProjectName>
                <StreamName>BaseOutput</StreamName>
            </EspSubscriberParameters>
        </Parameters>
    </Module>

      <Module type="formatter">
        <InstanceName>XmlListOutputFormatter</InstanceName>
        <Name>EspToXmlStringFormatter</Name>
        <Next>StreamingOutputFormatter</Next>
        <Parallel>true</Parallel>
        <Parameters>
        </Parameters>
    </Module>

    <Module type="formatter">
      <InstanceName>StreamingOutputFormatter</InstanceName>
      <Name>StringToStreamFormatter</Name>
      <Next>MySocketOutTransporter</Next>
      <Parameters>
        <StringToStreamFormatterParameters>
          <Delimiter>\n</Delimiter>
          <IncludeDelimiter>true</IncludeDelimiter>
          <AppendString>\n</AppendString>
          <AppendPosition>end</AppendPosition>
          <IgnoreSpace>true</IgnoreSpace>
         <CharsetName>US-ASCII</CharsetName>
       </StringToStreamFormatterParameters>
      </Parameters>
    </Module>

    <Module type="transporter">
      <InstanceName>MySocketOutTransporter</InstanceName>
      <Name>SocketOutputTransporter</Name>
      <Parameters>
        <SocketOutputTransporterParameters>
          <Host>localhost</Host>
          <Port>9996</Port>
```

```
          <EpFile></EpFile>
          <Retryperiod>10</Retryperiod>
          <KeepAlive>true</KeepAlive>
        </SocketOutputTransporterParameters>
      </Parameters>
    </Module>
  </Modules>

    <EspProjects>
        <EspProject>
            <Name>EspProject2</Name>
            <Uri>esp://localhost:19011/sample_workspace/
socket_xmllist_output</Uri>
            <Security>
                <User></User>
                <Password encrypted="false"></Password>
                <AuthType>user_password</AuthType>
            </Security>
        </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Adapter Controller Parameters

The Adapter Controller port listens for commands. The `controller.xml` file is located in `%ESP_HOME%/adapters/framework/config` directory. This file is shared among all the adapter within the `%ESP_HOME%/adapters/framework/instances` directory.

Parameter	Description
ControlPort	(Required) Section containing the **MinPort** and **MaxPort** parameters.
MinPort	Type: `int` (Required) The minimum port number that can act as a control port number. The adapter framework allocates this number automatically. A recommended value is 19082.
MaxPort	Type: `int` (Optional) The maximum port number that can act as a control port number. The adapter framework allocates this number automatically. The default value is 65535.

Parameter	Description
Statistics_Update_Interval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework publishes statistics data to the ESP project to which the adapter is connected. A recommended value is 5000.
MonitorInterval	Type: `int` (Required) The interval, in milliseconds, that the adapter framework checks the status of internal resource allocation. The framework logs a message in the log file if a resource is about to be exhausted. A recommended value is 3000.

Logging

The Socket XML Input and Output adapter use the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing the logging configuration is located in the `%ESP_HOME%\adapters\framework\config` directory.

Specify the location of the logging file you wish to use in the **Log4jProperty** parameter within the adapter configuration file. You can modify the logging levels within this file or the `$ESP_HOME/adapters/framework/config/log4j.properties`, which is used by default. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.

Level	Description
TRACE	Logs fine-grained debug messages that capture the flow of the application.
ALL	Logs all events.

Note: Setting the log level to DEBUG or ALL may result in large log files. The default value is INFO.

Here is a sample log4j.properties file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The log4j.rootLogger option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the log4j.rootLogger to any value more verbose than info may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose

than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Starting the Socket XML Adapter

To start the Socket XML adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

   ```
   cd %ESP_HOME%\cluster\nodes\node1
     %ESP_HOME%\bin\esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   %ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

   UNIX:

   a. Start the example cluster.

   ```
   cd $ESP_HOME/cluster/nodes/node1
     $ESP_HOME/bin/esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   $ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   $ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   $ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

2. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: For the Socket XML Input adapter: `cd $ESP_HOME/adapters/framework/instances/socket_xmllist_input` For the Socket XML Output adapter: `cd $ESP_HOME/adapters/framework/instances/socket_xmllist_output` `./start_adapter.sh <adapter configuration file name>`
Windows	Open a command window and enter: For the Socket XML Input adapter:`cd %ESP_HOME%/adapters/framework/instances/socket_xmllist_input` For the Socket XML Output adapter:`cd %ESP_HOME%/adapters/framework/instances/socket_xmllist_output` `start_adapter.bat <adapter configuration file name>`

Stopping the Socket XML Adapter

To stop the Socket XML adapter from the command line, execute the **stop** command.

When you are running the adapter from the command line, stop the adapter first before stopping the project.

Operating System	Step
UNIX	Open a terminal window and enter: For the Socket XML Input adapter: `cd $ESP_HOME/adapters/framework/instances/socket_xmllist_input` For the Socket XML Output adapter: `cd $ESP_HOME/adapters/framework/instances/socket_xmllist_output` `./stop_adapter.sh <adapter configuration file name>`

Operating System	Step
Windows	Open a command window and enter: For the Socket XML Input adapter: `cd %ESP_HOME%/adapters/framework/instances/socket_xmllist_input` For the Socket XML Output adapter: `cd %ESP_HOME%/adapters/framework/instances/socket_xmllist_output` `stop_adapter.bat <adapter configuration file name>`

TIBCO Rendezvous Adapter

Adapter type: tibcorvplugin. The SAP Sybase Event Stream Processor TIBCO Rendezvous adapter subscribes to and publishes data from Event Stream Processor to the Rendezvous server.

The Rendezvous adapter:

- Connects to a Rendezvous server, opens sessions, subscribes and unsubscribes to subjects
- Translates Rendezvous messages into Event Stream Processor records and vice-versa
- Receives messages from and publishes messages to the Rendezvous server

See also

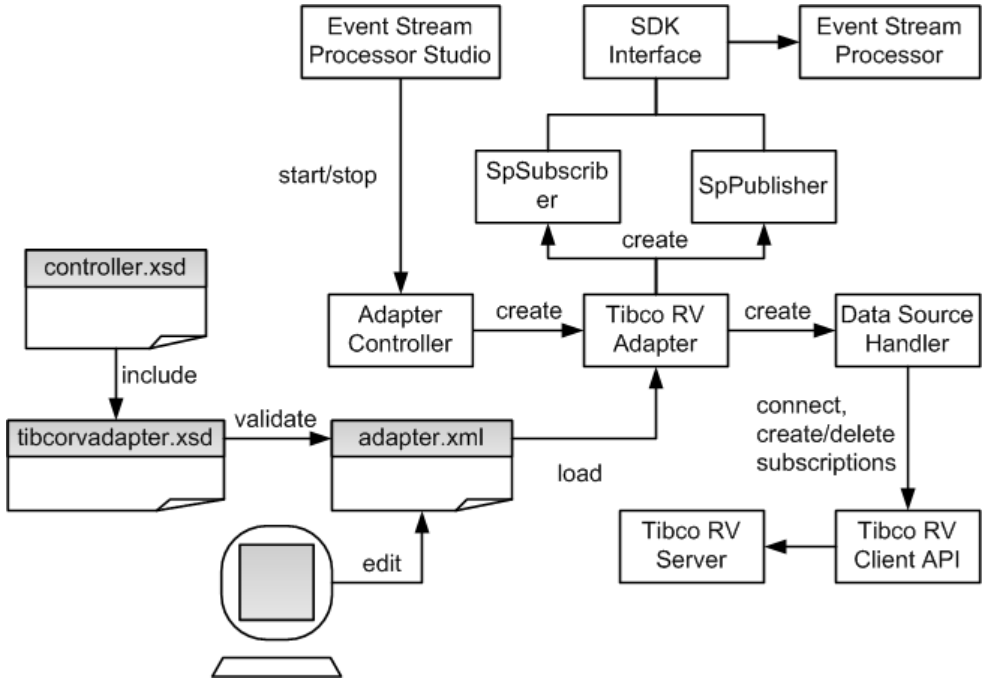
- *Chapter 4, Guaranteed Delivery* on page 1013

Control Flow

The adapter loads its configuration from a file (for example, `adapter.xml`), and validates it against the adapter schema (`tibcorvadapter.xsd`), which includes the API-wide controller schema (`controller.xsd`).

You cannot edit schemas.

Figure 13: TIBCO Adapter Control Flow



The Adapter Controller creates an instance of the adapter which then receives and executes user commands. The Adapter Controller can execute **start**, **stop**, and **status** commands.

Start Command

The **start** command configures and starts the adapter command and control interface.

The Data Source Handler connects to and initiates a session with the Rendezvous server using the Rendezvous Client API. The SpSubscriber and SpPublisher components connect to Event Stream Processor using the Pub/Sub interface. SpSubscriber starts listening to the outbound streams and SpPublisher is ready to publish data to inbound streams.

If the **start** command is executed when there is a running instance of the adapter, it is ignored and a warning is sent that the adapter is already running.

See also

- *Starting the TIBCO Rendezvous Adapter* on page 937

Stop Command

The **stop** command disconnects the SpPublisher and SpSubscriber from Event Stream Processor, causes the Data Source Handler to close the session and disconnect from the

datasource, causes the Adapter Controller to stop listening to user commands, and terminates the adapter process.

If the **stop** command is executed when there is no instance of a running adapter, the command is ignored and a warning is sent.

See also

- *Stopping the TIBCO Rendezvous Adapter* on page 939

Status Command

The **status** command reports the adapter status, and the Adapter Controller prints out its status: either running or stopped.

See also

- *Checking the TIBCO Rendezvous Adapter Status* on page 938

Data Streams

The adapter stores each Rendezvous message in a stream record.

A single stream may store messages on different subjects. The subject is stored in a mandatory column Subject. The rest of the columns correspond to fields in Rendezvous messages.

Ensure the names of the stream columns are identical to the names of the corresponding scalar fields in Rendezvous messages. In case of message-type fields, the columns adhere to the following naming convention:

<message type field name>_<field name>

The adapter supports embedded messages of arbitrary depth. Columns unrelated to Rendezvous messages are not allowed, and fields of array type are not supported.

The Client and Date columns correspond to scalar fields. Trade is an embedded message which contains two fields: Price and Volume.

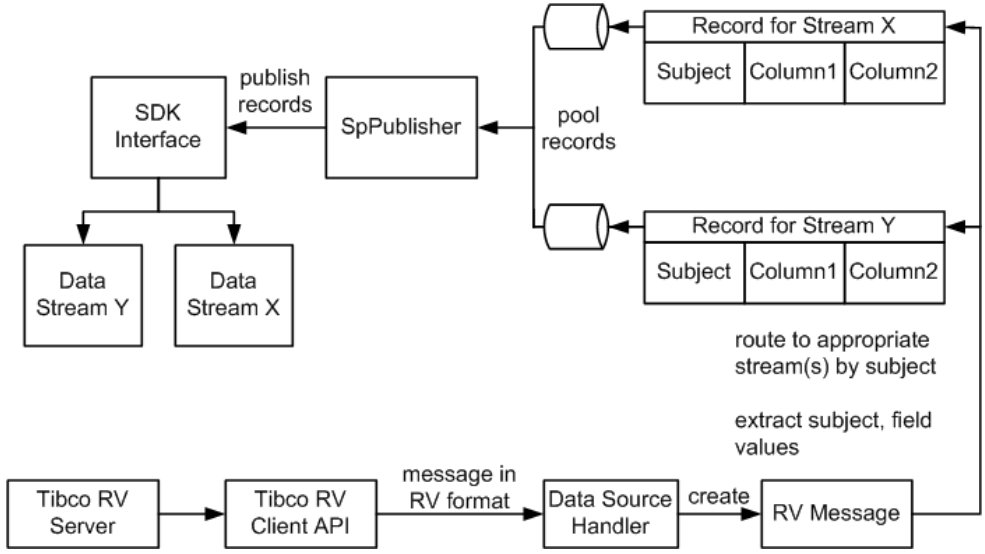
Table 13. Sample data strean

Sub-ject	Client	Date	Trade_Price	Trade_Volume
MySub-ject	UBS	2008-03-13T08:19:30	34.7	6000

Message Flow

The start command initiates the message flow through the adapter.

This figure shows the inbound message flow.



On start-up, the adapter subscribes to all subjects listed in the inbound section of the configuration file. Wildcard subscriptions are supported. Inbound messages are received via client API callbacks. The messages are passed on to the Message Distributor.

The Message Distributor converts each Rendezvous message into a record targeting one or more data streams. The records are now ready to be published to Event Stream Processor, but they are not published immediately. Records are queued, then picked up by the SpPublisher object on separate threads, one thread for each record queue. You can configure the queue capacity. A larger queue is less likely to overflow in the event of a message burst. When the queue becomes three-quarters full, a warning is logged. Another warning is logged when the queue returns to three-quarters empty. If the queue is full, the adapter waits until room becomes available before placing the next record.

Records are published asynchronously. The adapter receives no feedback from Event Stream Processor. In the event of a failover, the Pub/Sub API switches, as configured to the spare Event Stream Processor instance without message loss.

Note: For outbound records, opcodes (values for ESP_OPS) are not communicated to the TIBCO Rendezvous server. For inbound records, all records have "p" (upsert) set as the opcode before publishing to the Server.

Datatype Mapping for the TIBCO Rendezous Adapter

Event Stream Processor datatypes map to TIBCO Rendezous datatypes.

Event Stream Processor Datatype	TIBCO Datatype
`boolean`	TibcorvMsg.Bool
`integer`	TibcorvMsg.i32
long	TibcorvMsg.i64
float/money/money1-money15	TibcorvMsg.f64
string	TibcorvMsg.string
date/timestamp	TibcorvMsg.datetime
bigdatetime	TibcorvMsg.i64
interval	TibcorvMsg.i64
binary	TibcorvMsg.string

Setting the JAVA_HOME Environment Variable

Set the JAVA_HOME environment variable to point to the Java directory.

Prerequisites

- Install Java Runtime Environment version 1.7.0_1 or higher. To see if you have a suitable version of Java installed, go to *http://www.java.com/en/download/installed.jsp*.
- Copy the TIBCO Rendezvous binary libraries from your TIBCO installation to the adapter's host machine. Create a new directory under `$ESP_HOME/adapters/tibco_rv` called `lib/tibco/<platform type>`, where `<platform_type>` is retrieved by the **arch** command. Then, copy the TIBCO Rendezvous libraries to this directory.
- Copy the TIBCO Rendezvous binary libraries from your TIBCO installation to the adapter's host machine under `%ESP_HOME%\adapters\tibco_rv\lib\tibco\<platform_type>`, where `<platform_type>` is either win32 or win64.

Task

Set the JAVA_HOME environment variable to the directory path where Java Runtime Environment 1.7.0_1 or higher is installed.

Next

- Verify that the ESP_HOME environment variable is set correctly.

Configuration

Configuration information for the TIBCO Rendezvous adapter.

TIBCO Rendezvous Adapter Directory

The adapter directory contains all files, such as configuration files, templates, examples, and JAR files, relating to the adapter.

```
README.txt  Quick Guide
ReleaseNotes.txt  Release Notes

bin/
    start_adapter.bat  Standalone adapter startup script
    start_adapter.sh      Standalone adapter startup script
    adapter-plugin.bat Plug-in connector startup script
    adapter-plugin.sh  Plug-in connector startup script

config/
    controller.xsd     Controller schema
    log4j.properties   Sample logging configuration
    tibcorvadapter.xsd  Adapter schema
    login.config  Authentication configuration

examples/        Working example
    GD       Working example for guaranteed delivery

libj/

    commons-codec-1.3.jar Required by SDK API
    commons-collections-3.2.1.jar
    commons-configuration-1.6.jar
    commons-lang-2.6.jar
    commons-logging-1.1.jar Logging library
    esp_adapter_api.jar Adapter API code
    esp_adapter_tibrb.jar  Tibco Rendezvous adapter library
    esp_i18n.jar
    esp_license.jar
    esp_sdk.jar ESP SDK library
    log4j-1.2.16.jar Logging library
    postgresql.jar
    sylapi.jar
    ws-commons-util-1.0.2.jar Required by ESP SDK
    xerces-impl-2.9.1.jar XML parser library
    xmlrpc-client-3.1.3.jar Required by ESP SDK
    xmlrpc-common-3.1.3.jar Required by ESP SDK
```

In addition to the files listed here, copy the TIBCO Rendezvous binary libraries from your TIBCO installation to the adapter's host machine. For Linux and Solaris, create a new directory under `$ESP_HOME/adapters/tibco_rv` called `lib/tibco/<platform_type>`, where `<platform_type>` is retrieved by the **arch** command. For

Windows, the `lib\tibco\<platform type>` directory already exists, where `<platform_type>` is either win32 or win64. For all platforms, copy the TIBCO libraries over to the adapter's `<platform_type>` directory.

Schema and Configuration File

The adapter configuration loads from a file and validates against the adapter schema.

The example folder contains a sample adapter configuration file.

Provide a valid configuration file, and ensure the adapter configuration specified in the file validates against the adapter schema.

Adapter Controller Parameters

The adapter **controllerPort** parameter specifies the adapter command and control port.

This parameter is defined in the `controller.xsd` file located in the `config` directory.

Parameter Name	Description
controllerPort	Type: `positive integer` (Required) Specifies the adapter command and control port. User commands are sent to this port on localhost.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the TIBCO Rendezvous adapter.

These parameters are defined in the `controller.xsd` file in the `config` directory.

Parameter Name	Description
espAuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using keystore • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **espAuthType** is overridden by the **Authentication Mode** Studio start-up parameter.

Parameter Name	Description
espUser	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **espAuthType**). No default value.
espPassword	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **espPassword** value is encrypted. Default value is false. If set to true, the password value is decrypted using **espRSAKeyStore** and **espRSAKeyStorePassword**.
espProjectUri	Type: `string` (Required) Specifies the total project URI to connect to Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
pulseInterval	Type: `non-negative integer` (Optional) Specifies the time interval, in seconds, during which outbound record changes are coalesced by Event Stream Processor, then received by the adapter as a single event. If not set or set to 0, record changes are received individually as they occur.
espHeartbeatPeriod	Type: `positive integer` (Optional) Specifies the length of time, in seconds, that adapter waits before sending the next heartbeat to Event Stream Processor. If Event Stream Processor fails to receive two consecutive heartbeats, all records published by the adapter are marked stale. The default value is 10.
recordQueueCapacity	Type: `positive integer` (Optional) Specifies capacity of the record queues. Default value is 4096.

Parameter Name	Description
maxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.
maxPubPoolTime	Type: `positive integer` (Optional) Specifies the maximum period of time, in milliseconds, for which records are pooled before being published. If not set, pooling time is unlimited and the pooling strategy is governed by **maxPubPoolSize**. No default value.
useTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.
espRSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **espAuthType** is set to server_rsa, or the encrypted attribute for **espPassword** is set to true, or both.
espRSAKeyStorePassword	Type: `string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **espAuthType** is set to server_rsa, or the encrypted attribute for **espPassword** is set to true, or both.
espKerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **espAuthType** is set to kerberos.
espKerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **espAuthType** is set to kerberos.

Parameter Name	Description
espKerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **espAuthType** is set to kerberos.
espKerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **espAuthType** is set to kerberos.
espEncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **espPassword** is set to true. If left blank, RSA is used as default.

Stream Configuration

Use the streams section in the configuration file to map message subjects to data streams.

Input Stream Parameters

The **name** and **subjects** parameters defined within the **inboundStreamType** parameter specify the name of the data stream and message subjects.

These parameters are defined in the `tibcorvadapter.xsd` file located in the `config` folder.

Property ID	Description
name	Type: `string` (Required) Specifies the name of the data stream.
subjects	Type: `subjectsType` (Required) Specifies 0 or more message subjects. On start-up, the adapter subscribes to each of these subjects. Wildcard subscriptions are supported. If an inbound message arrives on any of the specified subjects, it is published to this data stream. Several data streams can specify the same subjects.

Output Stream Parameters

The output stream parameters are defined within the **outboundStreamType** parameter.

These parameters are defined in the `tibcorvadapter.xsd` file in the `config` folder.

Parameter ID	Description
name	Type: `string` (Required) Specifies the name of the data stream.
gdmode	Type: `boolean` (Advanced) If set to true, the adapter runs in guaranteed delivery (GD) mode and all GD-related parameters become required. Default value is false.
gdkeycolumnname	Type: `string` (Advanced) Specifies column name in the Flex operator holding the GD key. The GD key is a constantly increasing value that uniquely identifies every event regardless of the opcode in the stream of interest. No default value.
gdopcodecolumnname	Type: `string` (Advanced) Specifies name of column in Flex operator holding op-code. The opcode is the operation code (for example, inserts, update, or delete) of the event occurring in the stream of interest. No default value.
gdcontrolstream	Type: `string` (Advanced) Specifies name of the control window in the GD setup. The control window is a source stream that informs the Flex operator of which data has been processed by the adapter and can be safely deleted. No default value.
gdbatchsize	Type: `integer` (Advanced) Specifies number of records after which the control window must be updated with the latest GD key. Default value is 1000.
gdpurgeinterval	Type: `integer` (Advanced) Specifies number of records after which to purge the Flex operator. Default value is 1000.

See also

- *Chapter 4, Guaranteed Delivery* on page 1013

Rendezvous Server Settings

Configure the Rendezvous Server settings using the **rvDaemon**, **rvService**, **rvNetwork**, **cmmode**, **cmname**, and **cmledgerfile** parameters.

These parameters are defined in the `tibcorvadapter.xsd` file in the `config` folder.

Parameter Name	Description
rvDaemon	Type: `string` (Required) Specifies the colon-separated host name (or IP address) and port on which the Rendezvous server daemon runs.
rvService	Type: `string` (Optional) Specifies the name of the Rendezvous service.
rvNetwork	Type: `string` (Optional) Specifies the name of the Rendezvous network.
cmmode	Type: `boolean` (Required) Enables Certified Message (CM) transport mode for both input and output streams. Default value is false.
cmname	Type: `string` (Optional) Is a reusable name that identifies the CM transport to other CM transports. If its value is not specified in CM mode, then it defaults to 'sesp'. Its value is considered only if **cmmode** is set to true.
cmledgerfile	Type: `string` (Optional) Specify a valid file location to run the CM transport in file-based ledger mode. Otherwise, it runs in process-based ledger mode. Its value is considered only if **cmmode** is set to true.

Sample TIBCO Rendezvous Configuration File

Sample configuration file (`adapter.xml`) for the TIBCO Rendezvous adapter.

This file is in the `example` folder.

```
<adapter>

<!-- Adapter controller -->
 <controller>
  <controllerPort>13579</controllerPort>
 </controller>

<!-- Sybase Stream processor settings -->
 <esp>
   <espConnection>
```

```
        <espProjectUri>esp://localhost:19011/w1/p1</espProjectUri>
   </espConnection>

   <espSecurity>
        <espUser>espuser</espUser>
        <espPassword encrypted="false">espuser</espPassword>
        <espAuthType>none</espAuthType>
<!--
        <espRSAKeyStore>/keystore/keystore.jks</espRSAKeyStore>
       <espRSAKeyStorePassword>Sybase123</espRSAKeyStorePassword> --
>
        <espEncryptionAlgorithm>RSA</espEncryptionAlgorithm>
   </espSecurity>
  <maxPubPoolSize>1</maxPubPoolSize>
 </esp>

<!-- Stream to subject mapping -->
 <streams>
  <inbound>
   <stream>
    <name>MyInStream</name>
    <subjects>
     <subject>MySubject</subject>
    </subjects>
   </stream>
  </inbound>
  <outbound>
   <stream>
    <name>MyOutStream</name>
        <gdmode>false</gdmode>
        <gdkeycolumnname>gdkey</gdkeycolumnname>
        <gdopcodecolumnname>gdopcode</gdopcodecolumnname>
        <gdcontrolstream>W1_truncate</gdcontrolstream>
        <gdbatchsize>2</gdbatchsize>
        <gdpurgeinterval>4</gdpurgeinterval>
   </stream>
  </outbound>
 </streams>

<!-- Rendezvous settings -->
 <rvSettings>
  <rvDaemon>localhost:7500</rvDaemon>
  <cmmode>false</cmmode>
  <cmname>sesp</cmname>
  <cmledgerfile>C:\ledger.txt</cmledgerfile>
 </rvSettings>

</adapter>
```

TIBCO Rendezvous Adapter

The TIBCO Rendezvous adapter publishes stream data to and from a Rendezvous subject.

The authentication method is set to that of Event Stream Processor: rsa, kerberos, or Native OS (user name/password).

Install TIBCO Rendezvous Adapter version 1.0 or later to use this adapter.

Property Label	Description
Connector Directory Path	Property ID: **baseDir** Type: `directory` (Required) Specify the path to the adapter installation directory. This property is ignored if the **Connector Remote Directory Path** property is supplied. No default value. Use a forward slash for both UNIX and Windows paths.
Configuration File Path	Property ID: **configFilePath** Type: `configFilename` (Required) Specify the absolute path to the adapter configuration file. This property is ignored if the **Remote Configuration File Path** property is supplied. No default value. Use a forward slash for both UNIX and Windows paths.
Connector Remote Directory Path	Property ID: **remoteBaseDir** Type: `string` (Advanced) Specify the path to the adapter remote base directory (for remote execution only). If this property is supplied, the **Connector Directory Path** property is ignored. No default value. Use a forward slash for both UNIX and Windows paths.
Remote Configuration File Path	Property ID: **remoteConfigFilePath** Type: `string` (Advanced) Specify the absolute path to the adapter remote configuration file (for remote execution only). If this property is supplied, the **Configuration File Path** property is ignored. No default value. Use a forward slash for both UNIX and Windows paths.

Property Label	Description
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Logging

The TIBCO Rendezvous adapter uses the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample `log4j.properties` file containing the logging configuration is part of the TIBCO Rendezvous adapter distribution.

You can modify the logging levels of the `log4j.properties` configuration file which is located in the `%ESP_HOME%\adapters\<adapter-name>\config` directory. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Operation

Operate the adapter from the command line.

Ensure the **rvDaemon** and, optionally, **rvService** and **rvNetwork** parameters in the configuration are consistent with the Rendezvous server settings.

Ensure that the project that processes events contains inbound and outbound streams. Set the desired logging level in the `log4j.properties` file.

Starting the TIBCO Rendezvous Adapter

To start the TIBCO adapter from the command line, start Event Stream Processor and execute the **start** command.

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

   ```
   cd %ESP_HOME%\cluster\nodes\node1
     %ESP_HOME%\bin\esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   %ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   %ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

   UNIX:

   a. Start the example cluster.

   ```
   cd $ESP_HOME/cluster/nodes/node1
     $ESP_HOME/bin/esp_server --cluster-node node1.xml
   ```

   b. Compile CCL to create CCX.

   ```
   $ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
   ```

   c. Deploy the project on the cluster.

   ```
   $ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --add_project --workspace-
   name=w1 --project-name=p1 --ccx=model.ccx
   ```

   d. Start the deployed project on the cluster.

   ```
   $ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
   --username=sybase --password=sybase --start_project --
   workspace-name=w1 --project-name=p1
   ```

2. Start the adapter.

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/tibco_rv/bin`
	`./adapter.sh <configuration file path> start`

Windows	Open a command window and enter:
	`cd %ESP_HOME%\adapters\tibco_rv\bin`
	`adapter.bat <configuration file path> start`

Note: Use the **esp_subscribe** utility to ensure that inbound Rendezvous messages are successfully published to Event Stream Processor.

See also

- *Start Command* on page 922

Checking the TIBCO Rendezvous Adapter Status

To check the TIBCO adapter status from the command line, execute the **status** command.

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/tibco_rv/bin`
	`./adapter.sh <configuration file path> status`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/tibco_rv/bin`
	`adapter.bat <configuration file path> status`

You see the adapter status: running or stopped.

See also

- *Status Command* on page 923

Stopping the TIBCO Rendezvous Adapter

To stop the TIBCO adapter from the command line, execute the **stop** command.

Prerequisites

When you are running the adapter from the command line, stop the adapter first before stopping the project.

Task

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/tibco_rv/bin`
	`./adapter.sh <configuration file path> stop`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/tibco_rv/bin`
	`adapter.bat <configuration file path> stop`

See also

- *Stop Command* on page 922

Example: Subscribing and Publishing

Subscribe to a subject, upload an outbound record on that subject to Event Stream Processor, send the message to the Rendezvous server, and then receive and publish it as an inbound record.

Prerequisites

You have a network connection to a running instance of the TIBCO Rendezvous server. Operate this example from the command line.

Task

1. Set the username and password for the adapter:
   a) Edit `adapter.xml`.
   b) In the <User> and <Password> elements, enter `sybase`.
2. Set the username and password in the example environment:

Operating System	Step
UNIX	**1.** Edit the `set_example_env.sh` script **2.** Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.
Windows	**1.** Edit the `set_example_env.bat` script **2.** Set the ADAPTER_EXAMPLE_USERNAME and ADAPTER_EXAMPLE_PASSWORD variables to `sybase`.

3. Edit the `start_adapter.sh` script.
4. Set the JAVA_HOME environment variable to the directory where the Java Runtime Environment (JRE) is installed.
5. Start Event Stream Processor.

Operating System	Step

UNIX	Open a terminal window:
	1. Start the example cluster:
	`start_node.sh`
	2. Start the project on the cluster:
	`start_project.sh`

Windows	Open a command window:
	1. Start the example cluster:
	`start_node.bat`
	2. Add project to the cluster, and start it on the cluster:
	`start_project.bat`

6. Start the adapter.

Operating System	Step
UNIX	Open a terminal window and enter: `start_adapter.sh`
Windows	Open a command window and enter: `start_adapter.bat`

7. Wait five to ten seconds for the adapter to initialize.
8. Upload the outbound record.

Operating System	Step
UNIX	Open a terminal window and enter: `./upload.sh`
Windows	Open a command window and enter: `upload.bat`

9. Start the Event Stream Processor subscriber utility to view data stream content.

Operating System	Step
UNIX	Open a terminal window and enter: `./esp-subscribe.sh`
Windows	Open a command window and enter: `esp-subscribe.bat`

10. Note the inbound record published to Event Stream Processor.

Web Services (SOAP) Input and Output Adapter

The Web Services (SOAP) Input and Output adapters are SOAP Web services clients which connect to a Web service to obtain data to feed into Event Stream Processor and deliver output from Event Stream Processor to a Web service.

Both adapters include an adapter configuration file and a mapping file. Use the adapter configuration file to set up the SOAP Input and Output Transporter modules, the ESP Publisher and Subscriber modules, as well as establish a connection to Event Stream Processor. Use the mapping file to map Web service WSDL to ESP columns.

You can either manually create a mapping file or use the schema discovery functionality in ESP Studio to automatically create one. See *Discovering Schema and Creating a Mapping File for the Web Services (SOAP) Adapter* in the *Studio Users Guide* for detailed instructions on creating a mapping file in SAP Sybase Event Stream Processor Studio.

Note: The adapter does not support schema discovery if you are using HTTP Basic Access Authentication.

The Web Services (SOAP) Input adapter:

- Calls or polls a Web service
- Parses and converts the response into Event Stream Processor data format
- Feeds data into ESP streams

The Web Services (SOAP) Output adapter:

- Connects to a Web service
- Converts Event Stream Processor data into Web services request format (for example, XML)
- Outputs Event Stream Processor data into a Web service

The Web Services (SOAP) Input and Output adapters report custom statistics. Enable the time granularity option in the project configuration (ccr) file to get these statistics reported by the _ESP_Adapter_Statistics metadata stream.

The Web Services (SOAP) Input adapter reports these statistics:

- AdapterRunningTime
- TotalInputRowsNumber
- SuccessInputRowsNumber
- ErrorInputRowsNumber
- InputLatency

The Web Services (SOAP) Output adapter reports these statistics:

- AdapterRunningTime
- TotalOutputRowsNumber
- SuccessOutputRowsNumber
- ErrorOutputRowsNumber
- OutputLatency

See also

- *Chapter 4, Guaranteed Delivery* on page 1013
- *Adapter Support for Schema Discovery* on page 1005

Control Flow

The Web Services (SOAP) adapter loads its configuration from a file (for example, `adapter_config`) and validates it against the adapter schema `(framework.xsd)`.

The Adapter Controller creates an instance of the Web Services (SOAP) adapter, and receives and executes **start** and **stop** commands.

Figure 14: Web Services (SOAP) Input Adapter Control Flow

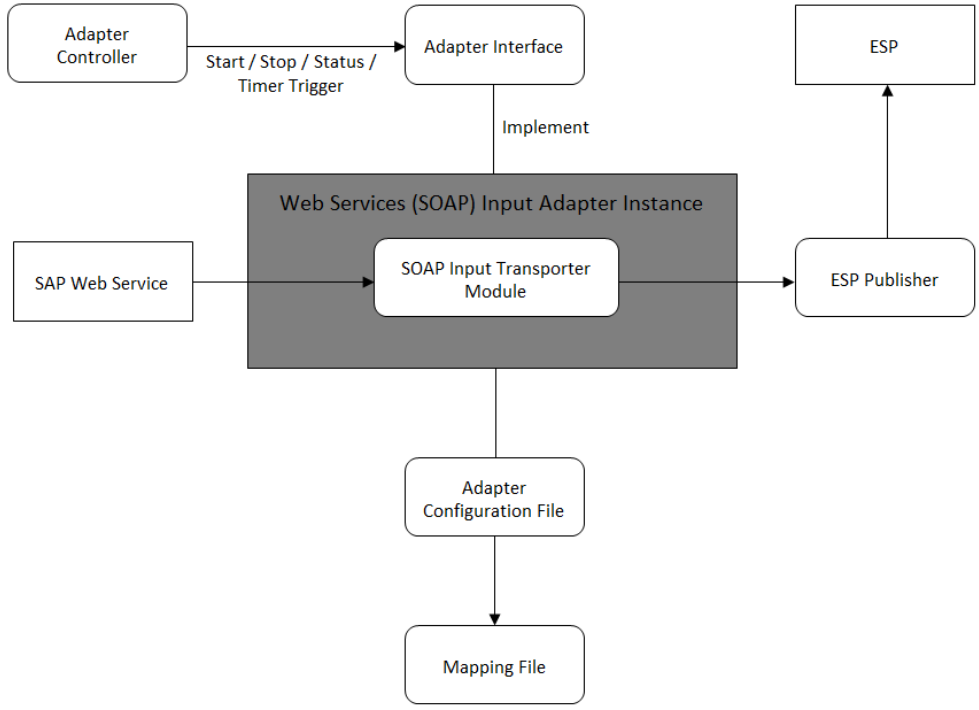
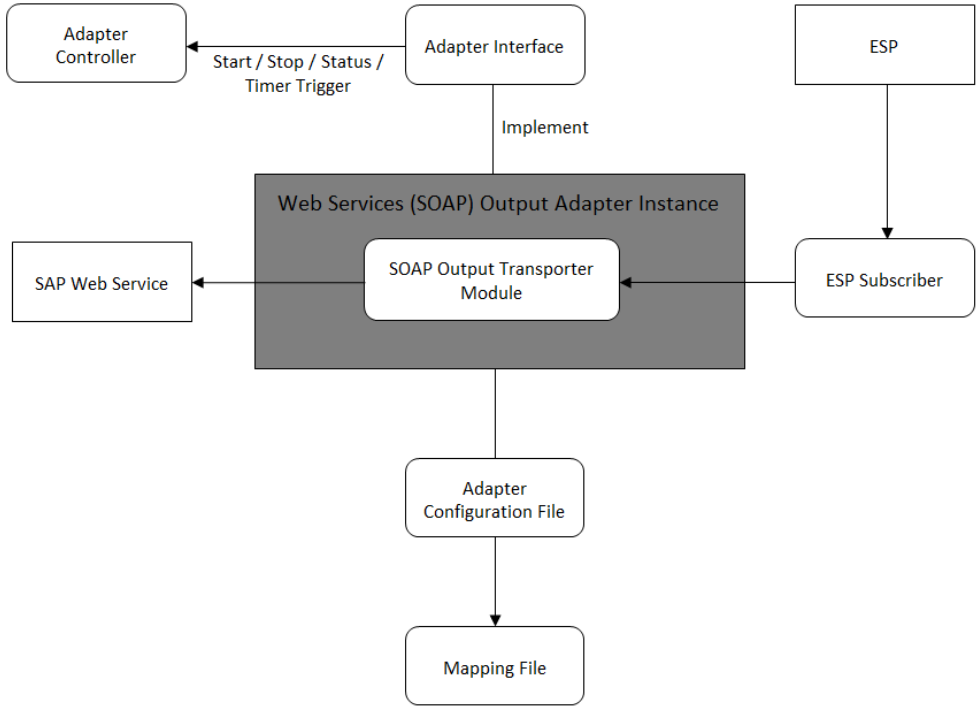


Figure 15: Web Services (SOAP) Output Adapter Control Flow



Start Command

When the **start** command is executed, it configures and starts the command and control interface, starts the adapter, discovers the Web service using its WSDL, connects the ESPPublisher and ESPPSubscriber components to Event Stream Processor, and then connects to the Web service using the endpoints specified in either the adapter configuration file or the WSDL.

The adapter ignores the **start** command if it is executed when there is a running instance of the adapter, and sends a warning.

Stop Command

When the **stop** command is executed, it disconnects the ESPPublisher and ESPSubscriber components from Event Stream Proecssor, disconnects from the Web service, and terminates the adapter process.

If the **stop** command is executed when there is no instance of a running adapter, the command is ignored and a warning is sent.

Message Flow

When the Web Services (SOAP) adapter is started, it establishes a connection with a Web service.

For the Web Services (SOAP) Input adapter, message flow begins as the adapter calls or polls a Web service and then converts and publishes data into ESP streams and windows to which the adapter is attached.

For the Web Services (SOAP) Output adapter, message flow begins as the adapter submits data from ESP streams and windows to a Web service.

Datatype Mapping for the Web Services (SOAP) Input Adapter

SOAP datatypes map to Event Stream Processor datatypes.

SOAP Datatype	Event Stream Processor Datatype
`string`	`string`
`boolean`	`boolean`
`float`	`double`
`double`	`double`
`decimal`	`double`
`binary`	`binary`
`integer`	`integer`
`nonPositiveInteger`	`integer`
`negativeInteger`	`integer`
`long`	`long`
`int`	`integer`
`short`	`integer`
`byte`	`integer`
`nonNegativeInteger`	`integer`
`unsignedLong`	`long`
`unsignedInt`	`long`
`unsignedShort`	`integer`

SOAP Datatype	Event Stream Processor Datatype
unsignedByte	integer
positiveInteger	integer
date	date
time	timestamp

Datatype Mapping for the Web Services (SOAP) Output Adapter

Event Stream Processor datatypes map to SOAP datatypes.

Event Stream Processor datatype	SOAP Datatype
binary	binary
boolean	boolean
date	date
money	double
double	double
integer	integer
interval	long
long	long
string	string
bigdatetime	time
timestamp	time

Web Services (SOAP) Adapter Directory

The adapter directory contains all files, such as configuration files, templates, examples relating to the adapter.

```
bin\ (contains startup scripts and discovery scripts for the input
and output adapters)

config\ (contains xml and xsd files used by the adapter and external
framework)

examples\ (contains example configuration files for Web Service
(SOAP) input and output adapters)
```

```
    bin\ (contains commands that you can run on the adapter)

    input\ (contains sample configuration files for the input
adapter, also contains a META-INF directory which includes
services.xml and .wsdl for the example)

    input_messageUT\ (contains sample configuration for an input
adapter using WS policy driven username/token security and ssl)

    input_transportUT\ (contains sample configuration for an input
adapter using transport level username/token and ssl)

    output\ (contains sample configuration files for the outputput
adapter, also contains a META-INF directory which includes
services.xml and .wsdl for the example)

    service\ (contains three .aar service archives; only deploy one
of these at a time)
        StockTraderService_noSec.aar – used with the input & output
examples
        StockTraderService_messageUT.aar - used for the WS policy
example
        StockTraderService_transportUT.aar - used for the transport
level example
        src\(contains source code for example services, including
security and extensible security classes used by WS security)

i18n\ (contains internationalization files)

libj\ (contains the jar files required by the adapter)

templates\ (contains helper scripts for starting and stopping nodes
and projects, and so on)

set_adapter_env (used to set the required environment variable for
the adapter)
```

Configuration

Configuration information for the Web Services (SOAP) Input and Output adapter.

Configuring the Web Services (SOAP) Input and Output adapter involves specifying values for modules and configuring the adapter to communicate with Event Stream Processor.

Adapter Control Port Parameter

Specify the optional **ControlPort** parameter in the adapter configuration file.

Parameter Name	Description
ControlPort	Type: `positive integer` (Required) Specifies the adapter command and control port. User commands are sent to this port on localhost. Default value is 19050.

Web Services (SOAP) Input Adapter Configuration

Configure the Web Services (SOAP) Input adapter by specifying values for the SOAP Input transporter, the ESP connector (EspPublisher), and Event Stream Processor.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

Transporter Module: SOAP Input Transporter

Parameter	Description
Module	(Required) Section containing all information for the SOAP input transporter. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.

Parameter	Description
Polling	Section containing the polling parameters, **Enabled** and **TimeInterval**.
Enabled	Type: `boolean` (Required) Turns on polling for the Web Service (SOAP) Input adapter. This parameter should always be enabled for the input adapter.
TimeInterval	Type: `integer` (milliseconds) (Required) The polling interval for the Web Services (SOAP) Input adapter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **SOAPInputTransportParameters** element.
SOAPInputTransportParameters	(Required) Section containing parameters for the SOAP Input transporter.
webservice	Type: `string` (Required) Specify the Web service to which the adapter connects. Include the **name** attribute. For example, `name="StockTraderService"`. This section contains the **urls**, **wsdlURL**, **serviceURL**, **serviceTimeout**, **serviceRetries**, **request**, **mappingFile**, and **security** parameters.
urls	(Required) Section containing the **wsdlURL** and **serviceURL** parameters.
wsdlURL	Type: `string` (Required) The URL for the Web service's WSDL.

Parameter	Description
serviceURL	Type: `string` (Dependent required) Required only if you want to call the Web service at a different endpoint than what is specified in the **wsdlURL** parameter. For example, for testing purposes or tcpMon.
serviceTimeout	Type: `integer` (Required) Specifies the amount of time the adapter waits before timing out the connection with the Web service.
serviceRetries	Type: `integer` (Required) Specifies the number of times the adapter retries each service endpoint if it fails to connect.
request	Type: `string` (Required) Section containing the **param** parameter. Specifies the Web service request. Includes the **action** attribute which specifies the action performed by the Web service operation that is being called.
param	Type: `string` (Required) Specify the input parameters to the Web service. You can specify multiple **param** parameters. Includes these attributes: • • **name –** (Required) the name of the Web service operation that is called. • • **savedVarName –** (Optional) the name of the variable listed in the mapping file. It allows the last value from a previous call to be used as the value for the next call. • • **initValue –** (Required) The initial value that is sent to the Web service. This value is sent every time unless the **savedVarName** attribute is specified, in which case the last value of the associated field is used, as specified in the mapping file.

Parameter	Description
mappingFile	Type: `string` (Required for adapter operation and schema discovery) Specify a valid path to the mapping file for the adapter.
security	(Required) Section containing the adapter security configuration. The adapter supports HTTPS protocol, HTTP basic access authentication, WS policy security, and transport level security. Specify either WS policy security or transport level security. This section contains the **sslTrustStore** and **sslTrustStorePassword** parameters, as well as the configuration parameters for either **WSPolicy** security or **TransportUsernameToken** security. **Note:** You can enable only one of HTTP basic access authentication, WS policy security, or transport level security at a time.
sslTrustStore	Type: `string` (Required only if ESP projects are running in SSL mode) Specify the path to the trust store file containing the server's certificate.

SOAP Input Transporter: HTTP Basic Access Authentication Parameters

BasicAccessAuthentication	(Required only if using HTTP basic access authentication) Stores the credentials used to authenticate against a Web service that is set up with HTTP basic access authentication. Section containing the **credentials**, **User**, **Password**, **RSAKeyStore** , **RSAKeyStorePassword** and **EncryptionAlgorithm** parameters for WS policy security.
credentials	(Optional) Section containing the **User**, **Password**, **RSAKeyStore**, **RSAKeyStorePassword** and **EncryptionAlgorithm** parameters.

User	Type: `string` (Required) Specifies the user name required to connect to the Web service.
Password	Type: `string` (Required) Specifies the password required to connect to the Web service. Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
RSAKeyStore	Type: `string` (Optional) Specify the location of an RSA keystore file which contains the key used to encrypt or decrypt the password set in the **Password** parameter.
RSAKeyStorePassword	Type: `string` (Optional) Stores the password to the RSA keystore file specified in the **Keystore** parameter.
EncryptionAlgorithm	Type: `string` (Optional) Specifies the algorithm that is used to decrypt the password specified in the **Password** parameter, if encrypted. For example, RSA.

SOAP Input Transporter: WS Policy Security Parameters

Parameter	Description
WSPolicy	(Required only if using WS policy security) Section containing the **credentials**, **User**, **Password**, **RSAKeyStore**, **RSAKeyStorePassword**, **policyClass**, and **param** parameters for WS policy security.
credentials	(Optional) Section containing the **User**, **Password**, **RSAKeyStore**, and **RSAKeyStorePassword** parameters.

Parameter	Description
User	Type: `string` (Optional) Specifies the user name required to connect to the Web service.
Password	Type: `string` (Optional) Specifies the password required to connect to the Web service. Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
RSAKeyStore	Type: `string` (Optional) Specify the location of an RSA keystore file which contains the key used to encrypt or decrypt the password set in the **Password** parameter.
RSAKeyStorePassword	Type: `string` (Optional) Stores the password to the RSA keystore file specified in the **Keystore** parameter.
policyClass	Type: `string` (Required only if using WS policy security) Specifies the plugin class to use that provides the policy information to the system. This class must extend from the class com.sap.esp.adapter.ws.security.WSPolicy.
param	Type: `string` (Optional) Specifies the parameters needed by the security mechanism of the WS policy. Includes "name" and "value" attributes.

SOAP Input Transporter: Transport Level Security Parameters

Parameter	Description
TransportUsernameToken	(Required only if using transport level security) Section containing the **credentials**, **User**, **Password**, **RSAKeyStore**, **RSAKeyStorePassword**, and **EncryptionAlgorithm** parameters. This type of security uses a username/token which is signed at the transport level.
credentials	(Required) Section containing the **User**, **Password**, **RSAKeyStore**, **RSAKeyStorePassword**, and **EncryptionAlgorithm** parameters. Specifies credentials for transport level security.
User	Type: `string` (Required) Specifies the user name required for connecting to the Web service.
Password	Type: `string` (Required) Specifies the password required to connect to the Web service. Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
RSAKeyStore	Type: `string` (Optional) Specify the location of an RSA keystore file which contains the key used to encrypt or decrypt the password set in the **Password** parameter.
RSAKeyStorePassword	Type: `string` (Optional) Stores the password to the RSA keystore file specified in the **Keystore** parameter.
EncryptionAlgorithm	Type: `string` (Optional) Specifies the algorithm that is used to decrypt the password specified in the **Password** parameter, if encrypted. For example, RSA.

SOAP Input Transporter: Working Directory Parameters

Parameter	Description
workingDir	Type: `string` (Required) Section containing the **proxy**, **host**, **port**, and **nonProxyHost** parameters. The directory where temporary adapter files are written. These files are cleaned up when the adapter shuts down.
proxy	(Optional) Section containing the **host**, **port**, and **nonProxyHosts** parameters. These properties specify a proxy server address through which the HTTP traffic is routed. This is often required in corporate networks, such as SAP, as individual machines do not connect directly to the external internet.
host	Type: `string` (Optional) If the adapter needs to connect through a proxy, specify the host of the proxy through which the adapter is connecting.
port	Type: `integer` (Optional) If the adapter needs to connect through a proxy, specify the host of the proxy through which the adapter is connecting.
nonProxyHosts	Type: `string` (Optional) Specify the host names which should not be routed through the proxy host. These are addresses located within the intranet that can be reached directly.

ESP Connector Module: ESP Publisher

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, formatter.

Parameter	Description
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Parameters	(Required) Section containing the **EspPublisherParameters** parameter.
EspPublisherParameters	(Required) Section containing parameters for the ESP publisher.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.

Parameter	Description
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream to which the adapter publishes data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
MaxPubPoolSize	Type: `positive integer` (Optional) Specifies the maximum size of the record pool. Record pooling, also referred to as block or batch publishing, allows for faster publication since there is less overall resource cost in publishing multiple records together compared to publishing records individually. Block publishing (record pooling or batch publishing) is disabled if this value is set to 1. The default value is 256.
UseTransactions	Type: `boolean` (Optional) If set to true, pooled messages are published to Event Stream Processor in transactions. If set to false, they are published in envelopes. Default value is false.
SafeOps	Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT, and converts DELETE to SAFEDELETE. The default value is false.
SkipDels	Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. The default value is false.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the Web Services (SOAP) Input adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.

Parameter	Description
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using key-store • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

Web Services (SOAP) Input Adapter Studio Properties

Adapter type: soapinput. Set these properties for the Web Services (SOAP) Input adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Adapter Configuration File	Property ID: **configFilePath** Type: `filename` (Required for adapter operation and schema discovery) Specify the path to the adapter configuration file.
Adapter Mapping File	Property ID: **mapFilePath** Type: `filename` (Optional for adapter operation; required for schema discovery) Specify the path to and the name of the adapter mapping (xml) file. This filename must match the name of the mapping file specified in the adapter configuration file.

Property Label	Description
JDK Location	Property ID: **jdkHome** Type: `directory` (Required for adapter operation and schema discovery) Specify the path to the JDK that the adapter uses. The JDK bit size should match the bit size of your Event Stream Processor installation. For example, if you installed Event Stream Processor on a 64-bit edition of Windows, install a 64-bit Java JDK. If you installed Event Stream Processor on Linux platform, install a 64-bit Linux Java JDK.
Discovery WSDL URL	Property ID: **discoveryWsdl** Type: `string` (Optional for adapter operation; required for schema discovery only) Specify the URL for the Web service WSDL to discover.
Discovery Working Directory	Property ID: **discoveryWorkingDir** Type: `directory` (Optional for adapter operation; required for schema discovery only) Specify the path to which temporary files are written during schema discovery.
Discovery Service Name	Property ID: **discoveryServiceName** Type: `string` (Optional for adapter operation; required for schema discovery only) Specify the name of the Web service to discover.
Host name of Proxy Server to Use	Property ID: **proxyHost** Type: `string` (Advanced) Specify the host name of the proxy server you wish to use for schema discovery.

Property Label	Description
Port of Proxy Server to Use	Property ID: **proxyPort** Type: `int` (Advanced) Specify the port of the proxy server you wish to use for schema discovery.
Non-proxy Hosts	Property ID: **nonProxyHosts** Type: `string` (Advanced) Specify the names of the hosts that do not use the proxy. Separate these names by |.
SSL TrustStore	Property ID: **trustStore** Type: `filename` (Advanced; optional for schema discovery) Location of SSL trust store to use for schema discovery. Set if you enabled SSL.
SSL TrustStore Password	Property ID: **trustStorePassword** Type: `string` (Advanced; optional for schema discovery) Password for the SSL TrustStore. Set if you enabled SSL.

Sample Configuration File for the Web Services (SOAP) Input Adapter

Sample configuration file for the Web Services (SOAP) Input adapter.

Here is a sample configuration file for a simple input adapter.

```
<?xml version="1.0"?>

<Adapter>

  <Name>Example streaming input adapter</Name>
  <Description>This is an example SOAP input adapter.</Description>
  <Modules>
    <Module type="transporter">
      <InstanceName>StockTraderServiceTransporter</InstanceName>
      <Name>SOAPInputTransporter</Name>
      <Polling>
        <Enabled>true</Enabled>
        <TimeInterval>20000</TimeInterval>
      </Polling>
      <Next>StockTraderServicePublisher</Next>
      <Parameters>
            <SOAPInputTransportParameters>
```

```
                <webservice name="StockTraderService">
                    <urls>
                      <wsdlURL>http://localhost:8080/axis2/services/
StockTraderService?wsdl</wsdlURL>
                  <!--    the serviceURL is only required if you wish
to call the service at a different endpoint than
                        is specified in the wsdl (e.g. for testing or
tcpMon)
                         <serviceURL>http://localhost:8081/axis2/
services/StockTraderService</serviceURL>
                    -->
                    </urls>
                    <serviceTimeout>60000</serviceTimeout>
                    <serviceRetries>2</serviceRetries>
                    <request action="getTransactions">
                     <param name="lastTransactionTime" initValue="1"
savedVarName="lastTime"/>
                    </request>
                  <mappingFile>stockTraderMappings.xml</mappingFile>
                </webservice>
                <workingDir>/tmp/adapter/soap</workingDir>
        <!--
                <proxy>
                    <host></host>
                    <port></port>
                  <nonProxyHosts>localHost|127.0.0.1</nonProxyHosts>
                </proxy>
        -->
            </SOAPInputTransportParameters>
      </Parameters>
    </Module>

    <Module type="espconnector">
      <InstanceName>StockTraderServicePublisher</InstanceName>
      <Name>EspPublisher</Name>
      <Parameters>
        <EspPublisherParameters>
          <ProjectName>StockTraderProject</ProjectName>
          <StreamName>tradesIn</StreamName>
        </EspPublisherParameters>
      </Parameters>
    </Module>
  </Modules>

    <EspProjects>
      <EspProject>
        <Name>StockTraderProject</Name>
        <Uri>esp://localhost:19011/w1/p1</Uri>
        <Security>
            <User></User>
            <Password></Password>
            <AuthType>user_password</AuthType>
<!--
            <RSAKeyStore>/keystore/keystore.jks</espRSAKeyStore>
            <RSAKeyStorePassword></espRSAKeyStorePassword>
-->
```

```
<!--
            <KerberosKDC>KDC</espKerberosKDC>
            <KerberosRealm>REALM</espKerberosRealm>
            <KerberosService>service/instance</espKerberosService>
            <KerberosTicketCache>/tmp/krb5cc_user</
espKerberosTicketCache>
-->
            <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
        </Security>
      </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Here is a sample configuration file for an input adapter that uses Policy driven security and communicates over HTTPS.

```
<?xml version="1.0"?>

<Adapter>

  <Name>Example streaming input adapter</Name>
  <Description>This is an example SOAP input adapter.</Description>
  <Modules>
    <Module type="transporter">
      <InstanceName>StockTraderServiceTransporter</InstanceName>
      <Name>SOAPInputTransporter</Name>
      <Polling>
        <Enabled>true</Enabled>
        <TimeInterval>20000</TimeInterval>
      </Polling>
      <Next>StockTraderServicePublisher</Next>
      <Parameters>
            <SOAPInputTransportParameters>
                <webservice name="StockTraderService">
                    <urls>
                     <wsdlURL>https://localhost:8443/axis2/services/
StockTraderService?wsdl</wsdlURL>
                 <!--    the serviceURL is only required if you wish
to call the service at a
                         different endpoint than is specified in the
wsdl
                             (e.g. for testing or tcpMon)

                        <serviceURL>http://localhost:8081/axis2/
services/StockTraderService?wsdl</serviceURL>
                    -->
                    </urls>
                    <serviceTimeout>60000</serviceTimeout>
                    <serviceRetries>2</serviceRetries>
                    <request action="getTransactions">
                     <param name="lastTransactionTime" initValue="1"
savedVarName="lastTime"/>
                    </request>
                  <mappingFile>stockTraderMappings.xml</mappingFile>
                    <security>
```

```
                        <!-- The keystore is used to establish https
connection with http server -->
                           <sslTrustStore>server.jks</sslTrustStore>
                       <sslTrustStorePassword></sslTrustStorePassword>

                           <WSPolicy>
                             <credentials>
                               <User>client</User>
                      <!-- here the password is the pw for the client
keystore -->
                                 <Password encrypted="false"></Password>
                      <!--
                                    <RSAKeyStore>/keystore/keystore.jks</
espRSAKeyStore>
                                     <RSAKeyStorePassword></
espRSAKeyStorePassword>
                      -->
                                 </credentials>

<policyClass>com.sap.esp.adapter.ws.security.MessageUTPolicy</
policyClass>
                             <param name="policyPath"
value="messageUT_policy.xml"/>
                             <param name="clientKeystore"
value="client.jks"/>
                           </WSPolicy>
                      </security>
                  </webservice>
                  <workingDir>/tmp/adapter/soap</workingDir>
          <!--
                  <proxy>
                      <host></host>
                      <port></port>
                    <nonProxyHosts>localHost|127.0.0.1</nonProxyHosts>
                  </proxy>
          -->
              </SOAPInputTransportParameters>
        </Parameters>
      </Module>

      <Module type="espconnector">
        <InstanceName>StockTraderServicePublisher</InstanceName>
        <Name>EspPublisher</Name>
        <Parameters>
          <EspPublisherParameters>
            <ProjectName>StockTraderProject</ProjectName>
            <StreamName>tradesIn</StreamName>
          </EspPublisherParameters>
        </Parameters>
      </Module>
    </Modules>

      <EspProjects>
        <EspProject>
          <Name>StockTraderProject</Name>
          <Uri>esp://localhost:19011/w1/p1</Uri>
```

```
        <Security>
            <User></User>
            <Password></Password>
            <AuthType>user_password</AuthType>
<!--
            <RSAKeyStore>/keystore/keystore.jks</espRSAKeyStore>
            <RSAKeyStorePassword></espRSAKeyStorePassword>
-->
<!--
            <KerberosKDC>KDC</espKerberosKDC>
            <KerberosRealm>REALM</espKerberosRealm>
            <KerberosService>service/instance</espKerberosService>
            <KerberosTicketCache>/tmp/krb5cc_user</
espKerberosTicketCache>
-->
            <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
        </Security>
      </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Here is a sample configuration file for an input adapter that uses transport level username/token security and communicates over HTTPS.

```
<?xml version="1.0"?>

<Adapter>

  <Name>Example streaming input adapter</Name>
  <Description>This is an example SOAP input adapter.</Description>
  <Modules>
    <Module type="transporter">
      <InstanceName>StockTraderServiceTransporter</InstanceName>
      <Name>SOAPInputTransporter</Name>
      <Polling>
        <Enabled>true</Enabled>
        <TimeInterval>20000</TimeInterval>
      </Polling>
      <Next>StockTraderServicePublisher</Next>
      <Parameters>
            <SOAPInputTransportParameters>
                <webservice name="StockTraderService">
                    <urls>
                     <wsdlURL>https://localhost:8443/axis2/services/
StockTraderService?wsdl</wsdlURL>
                  <!--    the serviceURL is only required if you wish
to call the service at a different endpoint than
                        is specified in the wsdl (e.g. for testing or
tcpMon)
                         <serviceURL>http://localhost:8081/axis2/
services/StockTraderService</serviceURL>
                    -->
                    </urls>
                    <serviceTimeout>60000</serviceTimeout>
                    <serviceRetries>2</serviceRetries>
```

```
                    <request action="getTransactions">
                    <param name="lastTransactionTime" initValue="1"
savedVarName="lastTime"/>
                    </request>
                  <mappingFile>stockTraderMappings.xml</mappingFile>
                    <security>
                       <!-- The keystore is used to establish https
connection with http server -->
                        <sslTrustStore>server.jks</sslTrustStore>
                     <sslTrustStorePassword></sslTrustStorePassword>

                        <TransportUsernameToken>
                          <credentials>
                        <!-- The user value should not be changed in
this adapter example -->
                            <User>sybase</User>
                        <!-- The password value shall match with the
parameter "TransportUTPassword" in service.xml-->
                            <Password encrypted="false"></Password>
                    <!--
                            <RSAKeyStore>/keystore/keystore.jks</
RSAKeyStore>
                         <RSAKeyStorePassword></RSAKeyStorePassword>
                    -->
                            <EncryptionAlgorithm>RSA</
EncryptionAlgorithm>
                          </credentials>
                        </TransportUsernameToken>
                    </security>
                </webservice>
                <workingDir>/tmp/adapter/soap</workingDir>
        <!--
                <proxy>
                    <host></host>
                    <port></port>
                  <nonProxyHosts>localHost|127.0.0.1</nonProxyHosts>
                </proxy>
        -->
            </SOAPInputTransportParameters>
      </Parameters>
    </Module>

    <Module type="espconnector">
      <InstanceName>StockTraderServicePublisher</InstanceName>
      <Name>EspPublisher</Name>
      <Parameters>
        <EspPublisherParameters>
          <ProjectName>StockTraderProject</ProjectName>
          <StreamName>tradesIn</StreamName>
        </EspPublisherParameters>
      </Parameters>
    </Module>
  </Modules>

    <EspProjects>
      <EspProject>
```

```
        <Name>StockTraderProject</Name>
        <Uri>esp://localhost:19011/w1/p1</Uri>
        <Security>
            <User></User>
            <Password></Password>
            <AuthType>user_password</AuthType>
<!--
            <RSAKeyStore>/keystore/keystore.jks</espRSAKeyStore>
            <RSAKeyStorePassword></espRSAKeyStorePassword>
-->
<!--
            <KerberosKDC>KDC</espKerberosKDC>
            <KerberosRealm>REALM</espKerberosRealm>
            <KerberosService>service/instance</espKerberosService>
            <KerberosTicketCache>/tmp/krb5cc_user</
espKerberosTicketCache>
-->
            <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
        </Security>
      </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Web Services (SOAP) Output Adapter Configuration

Configure the Web Services (SOAP) Output adapter by specifying values for the SOAP Output transporter, the ESP connector (EspSubscriber), and Event Stream Processor.

Logging

Parameter	Description
Log4jProperty	Type: `string` (Optional) Specify the path to the `log4j.properties` logging file you wish to use. The default value is `$ESP_HOME/adapters/framework/config/log4j.properties`.

ESPConnector Module: ESP Subscriber

Parameter	Description
Module	(Required) Section containing all information for this module. It contains a type attribute for specifying the module type. For example, espconnector.

Parameter	Description
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInput-Transporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Next	Type: `string` (Required) Specify the instance name of the module that follows this one.
Parameters	(Required) Section containing the **EspSubscriber-Parameters** parameter.
EspSubscriberParameters	(Required) Section containing parameters for the ESP subscriber.
ProjectName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Specifies the unique project tag of the ESP project to which the adapter is connected. For example, EspProject2. This is the same project tag that you specify later in the adapter configuration file in the **Name** parameter of the Event Stream Processor (**EspProjects**) parameters section. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the project name.

Parameter	Description
StreamName	Type: `string` (Required if running adapter in standalone mode; optional if running in managed mode) Name of the ESP stream from which the adapter subscribes to data. If you are starting the adapter with the ESP project to which it is attached (running the adapter in managed mode), you do not need to set this property as the adapter automatically detects the stream name.
GDMode	Type: `boolean` (Advanced) If set to true, the adapter runs in guaranteed delivery (GD) mode and all GD-related parameters become required. Default value is false.
GDKeyColumnName	Type: `string` (Advanced) Specifies column name in the Flex operator holding the GD key. The GD key is a constantly increasing value that uniquely identifies every event regardless of the opcode in the stream of interest. No default value.
GDOpcodeColumnName	Type: `string` (Advanced) Specifies name of column in Flex operator holding opcode. The opcode is the operation code (for example, inserts, update, or delete) of the event occurring in the stream of interest. No default value.
GDBatchSize	Type: `integer` (Advanced) Specifies number of records after which the control window must be updated with the latest GD key. Default value is 3.
GDPurgeInterval	Type: `integer` (Advanced) Specifies number of records after which to purge the Flex operator. Default value is 1000.

Parameter	Description
GDControlStream	Type: `string` (Advanced) Specifies name of the control window in the GD setup. The control window is a source stream that informs the Flex operator of which data has been processed by the adapter and can be safely deleted. No default value.

Transporter Module: SOAP Output Transporter

Parameter	Description
Module	(Required) Section containing all information for the SOAP output transporter module. It contains a type attribute for specifying the module type. For example, transporter.
InstanceName	Type: `string` (Required) Specify the instance name of the specific module you wish to use. For example, MyInputTransporter.
Name	Type: `string` (Required) The name of the module as defined in the `modulesdefine.xml` file. For example, <TransporterType>InputTransporter.
Polling	Section containing the polling parameter **Enabled**.
Enabled	Type: `boolean` (Required) Turns on polling for the Web Service Client Output adapter. This parameter should always be enabled for the output adapter.
Parameters	(Required) Section containing the **SOAPOutputTraansportParameters** parameter.
SOAPOutputTraansportParameters	(Required) Section containing the parameters for the SOAP Output transporter.

Parameter	Description
webservice	Type: `string` (Required) Specify the Web service to which the adapter connects. Include the **name** attribute. For example, `name="StockTraderService"`. This section contains the **urls**, **wsdlURL**, **serviceURL**, **serviceTimeout**, **serviceRetries**, **request**, and **mappingFile** parameters.
urls	(Required) Section containing the **wsdlURL** and **serviceURL** parameters.
wsdlURL	Type: `string` (Required) The URL for the Web service's WSDL.
serviceURL	Type: `string` (Dependent required) Required only if you want to call the Web service at a different endpoint than what is specified in the **wsdlURL** parameter. For example, for testing purposes or tcpMon.
serviceTimeout	Type: `integer` (Required) Specifies the amount of time the adapter waits before timing out the connection with the Web service.
serviceRetries	Type: `integer` (Required) Specifies the number of times the adapter retries each service endpoint if it fails to connect.
request	Type: `string` (Required) Specify the Web service request. Includes the **action** attribute which specifies the action performed by the Web service operation that is being called.

Parameter	Description
mappingFile	Type: `string` (Required for adapter operation and schema discovery) Specifies a valid path to the mapping file you want the adapter to use.
security	(Required) Section containing the adapter security configuration. The adapter supports HTTPS protocol, HTTP basic access authentication, WS policy security, and transport level security. Specify either WS policy security or transport level security. This section contains the **sslTrustStore** and **sslTrustStorePassword** parameters, as well as the configuration parameters for either **WSPolicy** security or **TransportUsernameToken** security. **Note:** You can enable only one of HTTP basic access authentication, WS policy security, or transport level security at a time.
sslTrustStore	Type: `string` (Required only if ESP projects are running in SSL mode) Specify the path to the trust store file containing the server's certificate.
sslTrustStorePassword	Type: `string` (Required only if ESP projects are running in SSL mode) Specify the password to access the trust store.

SOAP Output Transporter: HTTP Basic Access Authentication Parameters

BasicAccessAuthentication	(Required only if using HTTP basic access authentication) Stores the credentials used to authenticate against a Web service that is set up with HTTP basic access authentication. Section containing the **credentials**, **User**, **Password**, **RSAKeyStore** , **RSAKeyStorePassword** and **EncryptionAlgorithm** parameters for WS policy security.

credentials	(Optional) Section containing the **User**, **Password**, **RSAKeyStore**, **RSAKeyStorePassword** and **EncryptionAlgorithm** parameters.
User	Type: `string` (Required) Specifies the user name required to connect to the Web service.
Password	Type: `string` (Required) Specifies the password required to connect to the Web service. Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
RSAKeyStore	Type: `string` (Optional) Specify the location of an RSA keystore file which contains the key used to encrypt or decrypt the password set in the **Password** parameter.
RSAKeyStorePassword	Type: `string` (Optional) Stores the password to the RSA keystore file specified in the **Keystore** parameter.
EncryptionAlgorithm	Type: `string` (Optional) Specifies the algorithm that is used to decrypt the password specified in the **Password** parameter, if encrypted. For example, RSA.

SOAP Output Transporter: WS Policy Security Parameters

Parameter	Description
WSPolicy	(Required only if using WS policy security) Section containing the **credentials**, **User**, **Password**, **RSAKeyStore**, **RSAKeyStorePassword**, **policyClass**, and **param** parameters for WS policy security.

Parameter	Description
credentials	(Optional) Section containing the **User**, **Password**, **RSAKeyStore**, and **RSAKeyStorePassword** parameters.
User	Type: `string` (Optional) Specifies the user name required to connect to the Web service.
Password	Type: `string` (Optional) Specifies the password required to connect to the Web service. Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
RSAKeyStore	Type: `string` (Optional) Specify the location of an RSA keystore file which contains the key used to encrypt or decrypt the password set in the **Password** parameter.
RSAKeyStorePassword	Type: `string` (Optional) Stores the password to the RSA keystore file specified in the **Keystore** parameter.
policyClass	Type: `string` (Required only if using WS policy security) Specifies the plugin class to use that provides the policy information to the system. This class must extend from the class com.sap.esp.adapter.ws.security.WSPolicy.
param	Type: `string` (Optional) Specifies the parameters needed by the security mechanism of the WS policy. Includes "name" and "value" attributes.

SOAP Output Transporter: Transport Level Security Parameters

Parameter	Description
TransportUsernameToken	(Required only if using transport level security) Section containing the **credentials**, **User**, **Password**, **RSAKeyStore**, **RSAKeyStorePassword**, and **EncryptionAlgorithm** parameters. This type of security uses a username/token which is signed at the transport level.
credentials	(Required) Section containing the **User**, **Password**, **RSAKeyStore**, **RSAKeyStorePassword**, and **EncryptionAlgorithm** parameters. Specifies credentials for transport level security.
User	Type: `string` (Required) Specifies the user name required for connecting to the Web service.
Password	Type: `string` (Required) Specifies the password required to connect to the Web service. Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.
RSAKeyStore	Type: `string` (Optional) Specify the location of an RSA keystore file which contains the key used to encrypt or decrypt the password set in the **Password** parameter.
RSAKeyStorePassword	Type: `string` (Optional) Stores the password to the RSA keystore file specified in the **Keystore** parameter.
EncryptionAlgorithm	Type: `string` (Optional) Specifies the algorithm that is used to decrypt the password specified in the **Password** parameter, if encrypted. For example, RSA.

SOAP Output Transporter: Working Directory Parameters

Parameter	Description
workingDir	Type: `string` (Required) Section containing the **proxy**, **host**, **port**, and **nonProxyHost** parameters. The directory where temporary adapter files are written. These files are cleaned up when the adapter shuts down.
proxy	(Optional) Section containing the **host**, **port**, and **nonProxyHosts** parameters. These properties specify a proxy server address through which the HTTP traffic is routed. This is often required in corporate networks, such as SAP, as individual machines do not connect directly to the external internet.
host	Type: `string` (Optional) If the adapter needs to connect through a proxy, specify the host of the proxy through which the adapter is connecting.
port	Type: `integer` (Optional) If the adapter needs to connect through a proxy, specify the host of the proxy through which the adapter is connecting.
nonProxyHosts	Type: `string` (Optional) Specify the host names which should not be routed through the proxy host. These are addresses located within the intranet that can be reached directly.

Event Stream Processor Parameters

Event Stream Processor parameters configure communication between Event Stream Processor and the Web Services (SOAP) Output adapter.

Parameter	Description
EspProjects	Section containing parameters for connecting to Event Stream Processor.

Parameter	Description
EspProject	Section containing the **Name** and **Uri** parameters. Specifies information for the ESP project to which the adapter is connected.
Name	Type: `string` (Required) Specifies the unique project tag of the ESP project which the espconnector (publisher/subscriber) module references.
Uri	Type: `string` (Required) Specifies the total project URI to connect to the Event Stream Processor cluster. For example, `esp://localhost:19011/ws1/p1`.
Security	Section containing all the authentication parameters below. Specifies details for the authentication method used for Event Stream Processor.
User	Type: `string` (Required) Specifies the user name required to log in to Event Stream Processor (see **AuthType**). No default value.
Password	Type: `string` (Required) Specifies the password required to log in to Event Stream Processor (see **espAuthType**). Includes an "encrypted" attribute indicating whether the **Password** value is encrypted. Default value is false. If set to true, the password value is decrypted using **RSAKeyStore** and **RSAKeyStorePassword**.

Parameter	Description
AuthType	Type: `string` (Required) Specifies method used to authenticate to the Event Stream Processor. Valid values are: • **server_rsa –** RSA authentication using key-store • **kerberos –** Kerberos authentication using ticket-based authentication • **user_password –** LDAP, SAP BI, and Native OS (user name/password) authentication If the adapter is operated as a Studio plug-in, **AuthType** is overridden by the **Authentication Mode** Studio start-up parameter.
RSAKeyStore	Type: `string` (Dependent required) Specifies the location of the RSA keystore, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
RSAKeyStorePassword	Type:`string` (Dependent required) Specifies the keystore password, and decrypts the password value. Required if **AuthType** is set to server_rsa, or the encrypted attribute for **Password** is set to true, or both.
KerberosKDC	Type: `string` (Dependent required) Specifies host name of Kerberos key distribution center. Required if **AuthType** is set to kerberos.
KerberosRealm	Type: `string` (Dependent required) Specifies the Kerberos realm setting. Required if **AuthType** is set to kerberos.

Parameter	Description
KerberosService	Type: `string` (Dependent required) Specifies the Kerberos principal name that identifies an Event Stream Processor cluster. Required if **AuthType** is set to kerberos.
KerberosTicketCache	Type: `string` (Dependent required) Specifies the location of the Kerberos ticket cache file. Required if **AuthType** is set to kerberos.
EncryptionAlgorithm	Type: `string` (Optional) Used when the encrypted attribute for **Password** is set to true. If left blank, RSA is used as default.

Web Services (SOAP) Output Adapter Studio Properties

Adapter type: soapoutput. Set these properties for the Web Services (SOAP) Output adapter in the ESP Studio adapter properties dialog.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Adapter Configuration File	Property ID: **configFilePath** Type: `filename` (Required for adapter operation and schema discovery) Specify the path to the adapter configuration file.
Adapter Mapping File	Property ID: **mapFilePath** Type: `filename` (Optional for adapter operation; required for schema discovery) Specify the path to and the name of the adapter mapping (xml) file. This filename must match the name of the mapping file specified in the adapter configuration file.

Property Label	Description
JDK Location	Property ID: **jdkHome** Type: `directory` (Required for adapter operation and schema discovery) Specify the path to the JDK that the adapter uses. The JDK bit size should match the bit size of your Event Stream Processor installation. For example, if you installed Event Stream Processor on a 64-bit edition of Windows, install a 64-bit Java JDK. If you installed Event Stream Processor on Linux platform, install a 64-bit Linux Java JDK.
Discovery WSDL URL	Property ID: **discoveryWsdl** Type: `string` (Optional for adapter operation; required for schema discovery only) Specify the URL for the Web service WSDL to discover.
Discovery Working Directory	Property ID: **discoveryWorkingDir** Type: `directory` (Optional for adapter operation; required for schema discovery only) Specify the path to which temporary files are written during schema discovery.
Discovery Service Name	Property ID: **discoveryServiceName** Type: `string` (Optional for adapter operation; required for schema discovery only) Specify the name of the Web service to discover.
Host name of Proxy Server to Use	Property ID: **proxyHost** Type: `string` (Advanced) Specify the host name of the proxy server you wish to use.

Property Label	Description
Port of Proxy Server to Use	Property ID: **proxyPort** Type: `int` (Advanced) Specify the port of the proxy server you wish to use.
Non-proxy Hosts	Property ID: **nonProxyHosts** Type: `string` (Advanced) Specify the names of the hosts that do not use the proxy. Separate these names by |.
SSL Trust Store	Property ID: **trustStore** Type: `filename` (Advanced; optional for schema discovery) Location of SSL trust store to use for schema discovery. Set if you enabled SSL
SSL TrustStore Password	Property ID: **trustStorePassword** Type: `string` (Advanced; optional for schema discovery) Password for the SSL trust store. Set if you enabled SSL

Sample Configuration File for the Web Services (SOAP) Output Adapter

Sample configuration file for the Web Services (SOAP) Output adapter.

Here is a sample configuration file for a simple output adapter.

```
<?xml version="1.0"?>

<Adapter>

  <Name>Example SOAP webservice output adapter</Name>
  <Description>This is an example SOAP output adapter.</Description>
  <Modules>
    <Module type="espconnector">
        <InstanceName>StockTraderSubscriber</InstanceName>
        <Name>EspSubscriber</Name>
        <Next>StockTraderServiceTransporter</Next>
        <Parameters>
            <EspSubscriberParameters>
                <ProjectName>StockTraderProject</ProjectName>
                <StreamName>tradesOut</StreamName>
<!--
                    <GDMode>false</GDMode>
```

```
                    <GDKeyColumnName>gdKey</GDKeyColumnName>
                    <GDOpodeColumnName>gdOpcode</GDOpodeColumnName>
                    <GDBatchSize>5</GDBatchSize>
                    <GDPurgeInterval>300</GDPurgeInterval>
                    <GDControlStream>W1_truncate</GDControlStream>
-->
            </EspSubscriberParameters>
        </Parameters>
    </Module>


    <Module type="transporter">
      <InstanceName>StockTraderServiceTransporter</InstanceName>
      <Name>SOAPOutputTransporter</Name>
      <Polling>
        <Enabled>true</Enabled>
        <TimeInterval>20000</TimeInterval>
      </Polling>
      <Parameters>
            <SOAPOutputTransportParameters>
                <webservice name="StockTraderService">
                    <urls>
                      <wsdlURL>http://localhost:8080/axis2/services/
StockTraderService?wsdl</wsdlURL>
                 <!--    the serviceURL is only required if you wish
to call the service at a different endpoint than
                        is specified in the wsdl (e.g. for testing or
tcpMon)
                         <serviceURL>http://localhost:8081/axis2/
services/StockTraderService</serviceURL>
                    -->
                    </urls>
                    <serviceTimeout>60000</serviceTimeout>
                    <serviceRetries>2</serviceRetries>
                    <request action="executeTrade"/>
                  <mappingFile>stockTraderMappings.xml</mappingFile>
                </webservice>
                <workingDir>/tmp/adapter/soap</workingDir>
        <!--
                <proxy>
                    <host></host>
                    <port></port>
                  <nonProxyHosts>localHost|127.0.0.1</nonProxyHosts>
                </proxy>
        -->
            </SOAPOutputTransportParameters>
      </Parameters>
    </Module>
    </Modules>

    <EspProjects>
      <EspProject>
        <Name>StockTraderProject</Name>
        <Uri>esp://localhost:19011/w1/p1</Uri>
        <Security>
            <User></User>
```

```
            <Password></Password>
            <AuthType>user_password</AuthType>
<!--
            <RSAKeyStore>/keystore/keystore.jks</espRSAKeyStore>
            <RSAKeyStorePassword></espRSAKeyStorePassword>
-->
<!--
            <KerberosKDC>KDC</espKerberosKDC>
            <KerberosRealm>REALM</espKerberosRealm>
            <KerberosService>service/instance</espKerberosService>
            <KerberosTicketCache>/tmp/krb5cc_user</
espKerberosTicketCache>
-->
            <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
        </Security>
      </EspProject>
    </EspProjects>
    <GlobalParameters></GlobalParameters>
</Adapter>
```

Mapping a Web Service Response to an ESP Column

Create a mapping file to map Web service WSDL to ESP columns.

You can also use SAP Sybase Event Stream Processor Studio to generate a mapping file. See *Discovering Schema and Creating a Mapping File for the Web Services (SOAP) Adapter* in the *Studio Users Guide* for detailed instructions on creating a mapping file in SAP Sybase Event Stream Processor Studio.

If you choose to manually create the mapping file, run discovery using the Web service's WSDL. This generates an .xml file with all the possible fields, as well as the default datatypes. The fields in this discovery file are the names that you use as the `adapterField` values.

1. Create a new file or edit an existing sample mapping file (for example, `weatherMappings2.xml`).
2. Within the `<columnMappings>` section, specify the mappings. For example:

   ```
   <mapping espCol="getCityForecastByZIPResult_state"
   adapterField="GetCityForecastByZIPResult.State"/>
   <mapping espCol="tradeTime" adapterField="transaction.tradeTime"
   saveAs="lastTime"/>
   ```

   The `saveAs` attribute corresponds to the variable name specified in the **savedVarName** attribute of the **request** parameter, and must match exactly. This is applicable to the input adapter only and is ignored if specified for the output adapter.

Logging

The Web Services (SOAP) Input and Output adapter uses the Apache `log4j` API to log errors, warnings, and information and debugging messages. A sample

`log4j.properties` file containing the logging configuration is part of the Web Services (SOAP) Input and Output adapter distribution.

Specify the location of the logging file you wish to use in the **Log4jProperty** parameter within the adapter configuration file. You can modify the logging levels within this file or the `$ESP_HOME/adapters/framework/config/log4j.properties`, which is used by default. Set the ADAPTER_CLASSPATH environment variable to point to the configuration directory of each adapter for which you are configuring logging.

The logging levels in `log4j.properties` are:

Level	Description
`OFF`	Logs no events.
`FATAL`	Logs severe errors that prevent the application from continuing.
`ERROR`	Logs potentially recoverable application errors.
`WARN`	Logs events that possibly lead to errors.
`INFO`	Logs events for informational purposes.
`DEBUG`	Logs general debugging events.
`TRACE`	Logs fine-grained debug messages that capture the flow of the application.
`ALL`	Logs all events.

Note: Setting the log level to `DEBUG` or `ALL` may result in large log files. The default value is `INFO`.

Here is a sample `log4j.properties` file:

```
# Set root logger level to INFO and set appenders to stdout, file and
email
log4j.rootLogger=INFO, stdout, R

# stdout appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.stdout.Threshold=INFO

# file appender
log4j.appender.R=org.apache.log4j.DailyRollingFileAppender
log4j.appender.R.File=logs/rtviewadapter.log
log4j.appender.R.DatePattern='.'yyyy-MM-dd
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d{MM-dd-yyyy
```

```
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.R.Threshold=INFO

# email appender
log4j.appender.email=org.apache.log4j.net.SMTPAppender
log4j.appender.email.To=your.name@yourcompany.com
log4j.appender.email.From=alert.manager@yourcompany.com
log4j.appender.email.SMTPHost=yourmailhost
log4j.appender.email.BufferSize=1
log4j.appender.email.Subject=RTView Adapter Error
log4j.appender.email.layout=org.apache.log4j.PatternLayout
log4j.appender.email.layout.ConversionPattern=%d{MM-dd-yyyy
HH:mm:ss.SSS} %p [%t] (%C{1}.%M) %m%n
log4j.appender.email.Threshold=ERROR

log4j.logger.com.sybase.esp=INFO
```

The `log4j.rootLogger` option sets the default log behavior for all the sub-loggers in the adapter. In addition to the root logger, the adapter contains various sub-loggers that control logging for specific adapter functions.

Setting the `log4j.rootLogger` to any value more verbose than `info` may produce excess information. If you explicitly set the log level for a sub-logger, you overwrite the default setting for that particular logger. In this way, you can make sub-loggers more verbose than the default. The names for Event Stream Processor related loggers contain the string `com.sybase.esp`.

Operation

Start and stop the adapter from the command line.

Starting the Web Service (SOAP) Adapter

To start the Web Service (SOAP) adapter from the command line, start Event Stream Processor and execute the **start** command.

Prerequisites

Have a JDK installed and ready for use. The JDK bit size should match the bit size of your Event Stream Processor installation. For example, if you installed Event Stream Processor on a 64-bit edition of Windows, install a 64-bit Java JDK. If you installed Event Stream Processor on Linux platform, install a 64-bit Linux Java JDK.

Task

1. Start Event Stream Processor.

   Windows:

   a. Start the example cluster.

   ```
   cd %ESP_HOME%\cluster\nodes\node1
     %ESP_HOME%\bin\esp_server --cluster-node node1.xml
   ```

b. Compile CCL to create CCX.

```
%ESP_HOME%\bin\esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
%ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
%ESP_HOME%\bin\esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

UNIX:

a. Start the example cluster.

```
cd $ESP_HOME/cluster/nodes/node1
  $ESP_HOME/bin/esp_server --cluster-node node1.xml
```

b. Compile CCL to create CCX.

```
$ESP_HOME/bin/esp_compiler -i model.ccl -o model.ccx
```

c. Deploy the project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --add_project --workspace-
name=w1 --project-name=p1 --ccx=model.ccx
```

d. Start the deployed project on the cluster.

```
$ESP_HOME/bin/esp_cluster_admin" --uri=esp://localhost:19011
--username=sybase --password=sybase --start_project --
workspace-name=w1 --project-name=p1
```

2. Start the adapter.

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/soap/bin`
	`./adapter.sh <adapter_config_file> <jdk_home> start>`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/soap/bin`
	`adapter.bat <adapter_config_file> <jdk_home> start  >`

Stopping the Web Services (SOAP) Adapter

To stop the Web Services (SOAP) adapter from the command line, execute the **stop** command.

Prerequisites

When you are running the adapter from the command line, stop the adapter first before stopping the project.

Task

Operating System	Step

UNIX	Open a terminal window and enter:
	`cd $ESP_HOME/adapters/soap/bin`
	`./adapter.sh <adapter_config_file> <jdk_home> stop`

Windows	Open a command window and enter:
	`cd %ESP_HOME%/adapters/soap/bin`
	`adapter.bat <adapter_config_file> <jdk_home> stop`

Examples

Use the working examples provided with the adapter to learn how to subscribe and publish data to streams in Event Stream Processor.

Example: Using a Simple Web Services (SOAP) Input Adapter

Set up a basic Web Services (SOAP) Input adapter. This Web service has no security and communicates over HTTP.

1. Install Apache Tomcat.
2. Add Apache Axis2™ to Tomcat. Copy `axis2.war` to `tomcat/webapps`, and start Tomcat.

   Axis2 is automatically unzipped.
3. Ensure that the JDK_HOME environment variable is properly set. If it is not, you can set using `set_example_env.bat` or `set_example_env.sh`.

4. Remove any `StockTraderService_messageUT.aar` or `StockTraderService_transportUT.aar` files from the adapter `examples/service` directory if you previously ran those examples.
5. Copy the `examples/service/StockTraderService_noSec.aar` file to the `tomcat/webapps/axis2/WEB-INF/services` directory under your Web server.
6. Modify the `adapter_config.xml` file by setting `<User>` and `<Password>` to `sybase`:

```
<EspProjects>
      <EspProject>
        <Name>StockTraderProject</Name>
        <Uri>esp://localhost:19011/w1/p1</Uri>
        <Security>
            <User>sybase</User>
            <Password>sybase</Password>
            <AuthType>user_password</AuthType>
```

7. Modify `set_example_env.bat` or `set_example_env.sh` by setting `ADAPTER_EXAMPLE_USERNAME` and `ADAPTER_EXAMPLE_PASSWORD` to `sybase`.
8. Start the Web server.
9. Start the ESP node by running the `start_node.bat` or `start_node.sh` script.
10. Start the ESP project by running the `start_project.bat` or `start_project.sh` script.
11. Subscribe to the stream in the project by running the `subscribe.bat` or `subscribe.sh` script.
12. Start the adapter by running the `start_adapter.sh` or `start_adapter.sh` script.
    Data begins flowing in the subscription window.

Example: Using a Web Services (SOAP) Input Adapter with Policy Driven Security

Set up a Web Services (SOAP) Input adapter that uses Policy driven security and communicates over HTTPS.

The source code for the WSPolicy and MessageUT_Policy classes are located in the adapter `example/src` directory.

The steps below result in the creation of three keystores (`server.jks`, `client.jks`, and `service.jks`). Do not change the username "client" and "service".

1. Install Apache Tomcat.
2. Ensure that the JDK_HOME environment variable is properly set. If it is not, you can set using `set_example_env.bat` or `set_example_env.sh`.
3. Run `create_server_keystore <YOURSTOREPASSWORD>` to create a `server.jks`. Answer "localhost" to "What is your first and last name".

The `server.jks` is created under the current working directory. The Tomcat SSL HTTP connector and Web Services (SOAP) adapter use the `server.jks` to set up the HTTPS connection between them.

4. Run `create_client_service_cert <YOURCLIENTPASSWORD> <YOURSERVICEPASSWORD>` to create the `client.jks` and `service.jks` leystores.

   `<YOURCLIENTPASSWORD>` is the `client.jks` keystore password, and `<YOURSERVICEPASSWORD>` is the `service.jks` keystore password.

5. Add the following to the `tomcat/conf/server.xml` file:

```
<Connector port="8443"
              protocol="org.apache.coyote.http11.Http11Protocol"
               SSLEnabled="true" maxThreads="150"
               scheme="https" secure="true"
               keystoreFile="ESP_INSTALL\adapters\webservices
\examples\input_transportUT\server.jks"
               keystorePass="YOURSTOREPASSWORD"
               clientAuth="false"
               sslProtocol="TLS" />
```

6. Add Apache Axis2™ to Tomcat. Copy `axis2.war` to `tomcat/webapps`, and start Tomcat.

   Axis2 is automatically unzipped.

7. Copy the files in `rampart/modules` to `tomcat/webapps/axis2/WEB-INF/modules`.

8. Copy the files in `rampart/lib` to `tomcat/webapps/axis2/ WEB-INF/lib`.

9. Add the following to the `<Tomcat>\webapps\axis2\WEB-INF\conf\axis2.xml` file:

```
<transportReceiver name="https"
class="org.apache.axis2.transport.http.AxisServletListener">
<parameter name="port">8443</parameter>
</transportReceiver>
```

10. Modify the `adapter_config.xml` file as follows:

```
<security>
            <sslTrustStore>server.jks</sslTrustStore>
            <sslTrustStorePassword>YOURSTOREPASSWORD</
sslTrustStorePassword> <!--Just change the element to the same as
your input-->

            <WSPolicy>
                <credentials>
                          <User>client</User><!--Just change the
element to the same as your input-->
                  <!-- here the password is the pw for the client
keystore -->
                           <Password
encrypted="false">YOURCLIENTPASSWORD</Password> <!--Just change
the element to the same as your input-->
                </credentials>
```

```
<policyClass>com.sap.esp.adapter.ws.security.MessageUTPolicy</
policyClass>
                          <param name="policyPath"
value="messageUT_policy.xml"/>
                          <param name="clientKeystore"
value="client.jks"/>
                </WSPolicy>
    </security>
```

Set `<User>` and `<Password>` to the username and password used by node1 in `$ESP_HOME/cluster/examples`:

```
<EspProjects>
      <EspProject>
        <Name>StockTraderProject</Name>
        <Uri>esp://localhost:19011/w1/p1</Uri>
        <Security>
            <User>sybase</User>
            <Password>sybase</Password>
            <AuthType>user_password</AuthType>
```

11. Modify the `services.xml` file as follows:

```
<parameter name="MessageUTPassword">YOURSERVICEPASSWORD</
parameter> <!--Here, it need YOURSERVICEPASSWORD instead of
YOURCLIENTPASSWORD-->
    ......
    <ramp:RampartConfig xmlns:ramp="http://ws.apache.org/rampart/
policy">
     ......
       <ramp:property
name="org.apache.ws.security.crypto.merlin.keystore.password">YOU
RSERVICEPASSWORD</ramp:property><!--Here, it need
YOURSERVICEPASSWORD instead of YOURCLIENTPASSWORD-->
    </ramp:RampartConfig>
```

12. Run `ant create_sample_aar` to create the sample .aar file.
13. Remove any `StockTraderService_noSec.aar` or `StockTraderService_transportUT.aar` files from the adapter `examples/service` directory if you previously ran those examples.
14. Copy the `examples/service/StockTraderService_messageUT.aar` file to the `tomcat/webapps/axis2/WEB-INF/services` directory under your Web server.
15. Modify `set_example_env.bat` or `set_example_env.sh` by setting `ADAPTER_EXAMPLE_USERNAME` and `ADAPTER_EXAMPLE_PASSWORD` to `sybase`.
16. Start the Web server.
17. Start the ESP node by running the `start_node.bat` or `start_node.sh` script.
18. Start the ESP project by running the `start_project.bat` or `start_project.sh` script.

19. Subscribe to the stream in the project by running the `subscribe.bat` or `subscribe.sh` script.
20. Start the adapter by running the `start_adapter.sh` or `start_adapter.sh` script.
    Data begins flowing in the subscription window.

Example: Using a Web Services (SOAP) Input Adapter with Transport Level Security

Set up a Web Services (SOAP) Input adapter that uses transport level username/token security and communicates over HTTPS.

The steps below result in the creation of a keystore. Provide a password for the user "sybase" but do not change the username "sybase".

1. Install Apache Tomcat.
2. Ensure that the JDK_HOME environment variable is properly set. If it is not, you can set using `set_example_env.bat` or `set_example_env.sh`.
3. Run `create_server_keystore <YOURSTOREPASSWORD>` to create a `server.jks`. Answer "localhost" to "What is your first and last name".

   The `server.jks` is created under the current working directory. The Tomcat SSL HTTP connector and Web Services (SOAP) adapter use the `server.jks` to set up the HTTPS connection between them.
4. Add the following to the `tomcat/conf/server.xml` file:

```
<Connector port="8443"

protocol="org.apache.coyote.http11.Http11Protocol"
                   SSLEnabled="true" maxThreads="150"
                   scheme="https" secure="true"
                   keystoreFile="ESP_INSTALL\adapters\webservices
\examples\input_transportUT\server.jks"
                   keystorePass="YOURSTOREPASSWORD"
                   clientAuth="false"
                   sslProtocol="TLS" />
```

5. Add Apache Axis2™ to Tomcat. Copy `axis2.war` to `tomcat/webapps`, and start Tomcat.

   Axis2 is automatically unzipped.
6. Copy the files in `rampart/modules` to `tomcat/webapps/axis2/WEB-INF/modules`.
7. Copy the files in `rampart/lib` to `tomcat/webapps/axis2/ WEB-INF/lib`.
8. Add the following to the `<Tomcat>\webapps\axis2\WEB-INF\conf\axis2.xml` file:

```
<transportReceiver name="https"
class="org.apache.axis2.transport.http.AxisServletListener">
```

```
<parameter name="port">8443</parameter>
</transportReceiver>
```

9. Modify the `adapter_config.xml` file as follows:

```
<security>
            <sslTrustStore>server.jks</sslTrustStore>
            <sslTrustStorePassword>YOURSTOREPASSWORD</
sslTrustStorePassword><!--Just change the element to the same as
your input-->
            <TransportUsernameToken>
                <credentials>
                 <!-- The user value should not be changed in this
adapter example -->
                    <User>sybase</User>
                    <!-- The password value shall match with the
parameter "TransportUTPassword" in service.xml-->
                    <Password encrypted="false">YOURPASSWORD</
Password><!--Just change the element to the same as your input-->
                   <EncryptionAlgorithm>RSA</EncryptionAlgorithm>
                </credentials>
            </TransportUsernameToken>
    </security>
```

Set the `<User>` and `<Password>` to the username and password used by node1 in `$ESP_HOME/cluster/examples`:

```
<EspProjects>
      <EspProject>
        <Name>StockTraderProject</Name>
        <Uri>esp://localhost:19011/w1/p1</Uri>
        <Security>
            <User>sybase</User>
            <Password>sybase</Password>
            <AuthType>user_password</AuthType>
```

10. Modify the `services.xml` as follows:

```
<parameter name="TransportUTPassword">YOURPASSWORD</parameter>
```

11. Run `ant create_sample_aar` to create the sample .aar file.
12. Remove any `StockTraderService_noSec.aar` or `StockTraderService_messageUT.aar` files from the adapter `examples/service` directory if you previously ran those examples.
13. Copy the `examples/service/StockTraderService_transportUT.aar` file to the `tomcat/webapps/axis2/WEB-INF/services` directory under your Web server.
14. Modify `set_example_env.bat` or `set_example_env.sh` by setting `ADAPTER_EXAMPLE_USERNAME` and `ADAPTER_EXAMPLE_PASSWORD` to `sybase`.
15. Start the Web server.
16. Start the ESP node by running the `start_node.bat` or `start_node.sh` script.

17. Start the ESP project by running the `start_project.bat` or `start_project.sh` script.
18. Subscribe to the stream in the project by running the `subscribe.bat` or `subscribe.sh` script.
19. Start the adapter by running the `start_adapter.sh` or `start_adapter.sh` script.
    Data begins flowing in the subscription window.

Example: Using a Simple Web Services (SOAP) Output Adapter

Set up a basic Web Services (SOAP) Output adapter. This Web service has no security and communicates over HTTP.

1. Install Apache Tomcat.
2. Add Apache Axis2™ to Tomcat. Copy `axis2.war` to `tomcat/webapps`, and start Tomcat.

   Axis2 is automatically unzipped.
3. Ensure that the JDK_HOME environment variable is properly set. If it is not, you can set using `set_example_env.bat` or `set_example_env.sh`.
4. Remove any `StockTraderService_messageUT.aar` or `StockTraderService_transportUT.aar` files from the adapter `examples/service` directory if you previously ran those examples.
5. Copy the `examples/service/StockTraderService_noSec.aar` file to the `tomcat/webapps/axis2/WEB-INF/services` directory under your Web server.
6. Modify the `adapter_config.xml` file by setting `<User>` and `<Password>` to `sybase`:

```
<EspProjects>
      <EspProject>
        <Name>StockTraderProject</Name>
        <Uri>esp://localhost:19011/w1/p1</Uri>
        <Security>
            <User>sybase</User>
            <Password>sybase</Password>
            <AuthType>user_password</AuthType>
```

7. Modify `set_example_env.bat` or `set_example_env.sh` by setting `ADAPTER_EXAMPLE_USERNAME` and `ADAPTER_EXAMPLE_PASSWORD` to `sybase`.
8. Start the Web server.
9. Start the ESP node by running the `start_node.bat` or `start_node.sh` script.
10. Start the ESP project by running the `start_project.bat` or `start_project.sh` script.

11. Start the adapter by running the `start_adapter.sh` or `start_adapter.sh` script.
12. Upload data to Event Stream Processor by using the `upload.bat` or `upload.sh` script.
    Look for output in the Web server's `output log` file to verify whether the adapter is working.

WebSphere MQ Adapter

Event Stream Processor supplies WebSphere MQ adapters that enable you to read and write to and from the WebSphere MQ queue and an Event Stream Processor stream.

Event Stream Processor permits a WebSphere MQ server to read and write to the Event Stream Processor engine. You can customize these internal adapters to suit your needs. Because WebSphere MQ messages are unstructured, properly define a schema and prepare the MQ messages. The full range of Event Stream Processor datatypes is permitted in the schema definition. You can send binary data, strings, and so on, into and out of the Event Stream Processor engine.

Ensure MQ Client adapters have MQ 7.0.1 Client software installed. Failure to match the adapter to installed software results in errors. SAP WebSphere MQ adapters are designed to work with WebSphere MQ client software on the same host computer as the Server. The WebSphere MQ Server, however, can reside on the same computer or on another computer.

WebSphere MQ Input and Output adapters support opcodes for inserting, deleting, updating, and upserting data between the Event Stream Processor and WebSphere queues.

WebSphere MQ Input Adapter

Adapter type: wsmq_in. The default WebSphere MQ Input adapter reads a string in CSV format.

Ensure the order of the data in the message matches the schema of the input stream to which the adapter is attached. The WebSphere MQ Input adapter applies stream names and opcode instructions (INSERT, DELETE, UPDATE, UPSERT) to CSV data pulled from a queue. See the **expectStreamNameOpcode** property.

Note: If you are reading data from WSMQ using the WebSphere MQ Input adapter, but publishing data using a JMS program rather than the WebSphere MQ Output adapter, set the following line in the JMS program:
`queue.setTargetClient(com.ibm.mq.jms.JMSC.MQJMS_CLIENT_NONJMS_MQ);.`

To run the adapter successfully in Linux and UNIX installations, set the LD_LIBRARY_PATH to point to the MQ client libraries.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Note: To avoid data loss in the case of a server failure, SAP recommends you attach a log store to an input window.

Property Label	Description
Queue Name	Property ID: **QueueName** Type: `string` (Required) The name of the queue on the server from which the adapter receives messages. This queue must be managed by the indicated Queue Manager Name. No default value.
Queue Manager Name	Property ID: **QueueManagerName** Type: `string` (Required) The name of the queue manager on the server from which the adapter receives messages. No default value.
MQ System Name	Property ID: **SystemName** Type: `string` (Required) The name of the MQ server system. This may be a symbolic name or an IP address. No default value.
Port	Property ID: **Port** Type: `string` (Required) The port number on the MQ server system to which the MQ server queue listener is attached. No default value.
MQ Channel	Property ID: **Channel** Type: `string` (Required) The name of the MQ server channel associated with the queue. No default value.
Maximum Input Buffer Size	Property ID: **MaxBufferSize** Type: `uint` (Required) The maximum size of the buffer, in bytes. No default value.

Property Label	Description
Sync Point Commit Mode	Property ID: **syncpointmode** Type: `boolean` (Optional) Enables sync point commit mode. Set this to ensure guaranteed delivery of messages from the adapter to the Server. Default value is false.
Batch Size	Property ID: **batchsize** Type: `uint` (Optional) Specifies number of records in a batch to commit in sync point commit mode. Default value is 1.
CSV Delimiters	Property ID: **CsvSeparators** Type: `string` (Optional) The CSV field separators. Can be multiple characters. Default value is a comma (,).
CSV Escape Characters	Property ID: **CsvEscapeChars** Type: `string` (Optional) The character that escapes the meaning of special characters, including the delimiters, escape characters, and quote characters. Can be multiple characters. Default value is a backslash (\ ).
CSV Quote Characters	Property ID: **CsvQuoteChars** Type: `string` (Optional) The characters to delineate the beginning and end of a field. Default value is double quotes (").
Perform CSV Trimming	Property ID: **CsvTrimming** Type: `boolean` (Optional) If set to true, this trims leading and trailing whitespace from the beginning and end of each field. If true, a quoted field containing nothing but spaces is interpreted as NULL. Default value is true.

Property Label	Description
Stream Name, Opcode Expected	Property ID: **expectStreamNameOpcode** Type: `boolean` (Advanced) If true, the first two fields in CSV records are interpreted as stream name, opcode. Default value is false.
Timestamp Column Format	Property ID: **TimestampColumnFormat** Type: `string` (Optional) The format for timestamp values. Default value is YYYY-MM-DDTHH:MM:SS.SSS.
Date Column Name	Property ID: **DateColumnName** Type: `string` (Advanced) The format in which date values are stored in the file. Default value is YYYY-MM-DDTHH:MM:SS.SSS.
Delay Between Reconnection Attempts	Property ID: **reconnectAttemptDelayMSec** Type: `int` (Advanced) Number of milliseconds between attempts to reconnect to the WebSphere MQ server. Default value is 1000.
Maximum Number of Reconnection Attempts	Property ID: **maxReconnectAttempts** Type: `int` (Advanced) Number of attempts at reconnecting to the WebSphere MQ server before stopping. Use -1 to retry an unlimited number of times. Default value is 1.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

See also

- *Chapter 4, Guaranteed Delivery* on page 1013

WebSphere MQ Output Adapter

Adapter type: wsmq_out. The default WebSphere MQ Output adapter publishes a string in CSV format.

The WebSphere MQ adapter does not produce a header line because the schema of the stream publishing to the adapter determines the order and datatypes of the fields. Columns are published in the default display format for the appropriate datatype. The adapter prepends stream names and opcode instructions (insert, delete, update, upsert) to CSV data added to a queue. See the **prependStreamNameOpcode** property.

To run the adapter successfully in Linux and UNIX installations, set the LD_LIBRARY_PATH to point to the MQ client libraries to run the adapter successfully.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Note: This adapter uses TCP/IP for transfers. To use other protocols, determine the appropriate configuration and interface properties for those protocols.

Property Label	Description
Queue Name	Property ID: **QueueName** Type: `string` (Required) Name of the queue on the server to send messages. This queue must be managed by the indicated Queue Manager Name. No default value.
Queue Manager Name	Property ID: **QueueManagerName** Type: `string` (Required) Name of the queue manager on the server to send messages. No default value.
MQ System Name	Property ID: **SystemName** Type: `string` (Required) Name of the MQ server system. This may be a symbolic name or an IP address. No default value.
Port	Property ID: **Port** Type: `string` (Required) Port number on the MQ server system to which the MQ server queue listener is attached. No default value.

Property Label	Description
MQ Channel	Property ID: **Channel** Type: `string` (Required) Name of the MQ server channel associated with the queue. No default value.
CSV Field Separator	Property ID: **CsvSeparatorChar** Type: `string` (Optional) The CSV field separator, specify a single character. Default value is a comma (,).
CSV Escape Character	Property ID: **CsvEscapeChar** Type: `string` (Optional) The character to escape the meaning of special characters, including the field separator, escape character, and quote character. Default value is a backslash (\ ).
CSV Quote Character	Property ID: **CsvQuoteChar** Type: `string` (Optional) The character to delineate the beginning and end of a field, which can include anything. Any embedded quote characters are escaped. Default value is a double quote (").
Prepend Stream Name, Opcode	Property ID: **prependStreamNameOpcode** Type: `boolean` (Advanced) If true, every CSV record is prepended with stream name and an opcode. Default value is false.
Timestamp Column Format	Property ID: **TimestampColumnFormat** Type: `string` (Advanced) The format for timestamp values. Default value is YYYY-MM-DDTHH:MM:SS.SSS.
Date Column Format	Property ID: **DateColumnFormat** Type: `string` (Advanced) The format in which date values are stored in the file. Default value is YYYY-MM-DDTHH:MM:SS.SSS.

Property Label	Description
Runs Adapter in GD Mode	Property ID: **enableGDMode** Type: `boolean` (Advanced) If set to true, the adapter runs in guaranteed delivery (GD) mode and all GD-related parameters become required. Default value is false.
Name of Column Holding GD Key	Property ID: **gdKeyColumn** Type: `string` (Advanced) Specifies column name in the Flex operator holding the GD key. The GD key is a constantly increasing value that uniquely identifies every event regardless of the opcode in the stream of interest. No default value.
Name of Column Holding opcode	Property ID: **gdOpcodeColumn** Type: `string` (Advanced) Specifies name of column in Flex operator holding opcode. The opcode is the operation code (for example, inserts, update, or delete) of the event occurring in the stream of interest. No default value.
Name of Truncate Stream	Property ID: **gdControlStream** Type: `string` (Advanced) Specifies name of the control window in the GD setup. The control window is a source stream that informs the Flex operator of which data has been processed by the adapter and can be safely deleted. No default value.
Purge After Number of Records	Property ID: **gdPurgeInternal** Type: `int` (Advanced) Specifies number of records after which to purge the Flex operator. Default value is 1000.
Batch Size to Update Truncate Stream	Property ID: **gdBatchSize** Type: `int` (Advanced) Specifies number of records after which the control window must be updated with the latest GD key. Default value is 1000.

Property Label	Description
Delay Between Reconnection Attempts	Property ID: **reconnectAttemptDelayMSec** Type: `int` (Advanced) Number of milliseconds between attempts to reconnect to the WebSphere MQ server. Default value is 1000.
Maximum Number of Reconnection Attempts	Property ID: **maxReconnectAttempts** Type: `int` (Advanced) Number of attempts at reconnecting to the WebSphere MQ server before stopping. Use -1 to retry an unlimited number of times. Default value is 1.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

See also

- *Chapter 4, Guaranteed Delivery* on page 1013

Queue Configuration

Use this sample code to call a queue manager, server queue, channel, and listener.

A standard MQ server configuration provides:

- A default queue manager called `queue.manager.1`.
- A local server queue called `QUEUE1`.
- A Server-Connection channel called `channel1`.
- A listener called `listener1` on TCP/IP port 2001.

```
Create a default queue manager called queue.manager.1 and
start it:
crtmqm -q queue.manager.1
strmqm
dspmq # display list of active queues
Now create a local queue, a channel and a listener:
```

```
runmqsc
define qlocal(QUEUE1)
5
define channel (channel1) chltype (svrconn) trptype (tcp)
\
mcauser ('mqm')
define listener (listener1) trptype (tcp) control (qmgr) \
port (2001)
start listener (listener1)
end
```

Note: In this configuration example, backslashes (\) are used for readability, and because of space constraints. When configuring queues in the system, keep this information on one line.

CHAPTER 3 Schema Discovery

You can use the schema discovery feature to discover external schemas and create CCL schemas based on the format of the data from the datasource connected to an adapter.

Every row in a stream or window must have the same structure, or schema, which includes the column names, the column datatypes, and the order in which the columns appear. Multiple streams or windows may use the same schema, but a stream or window can only have one schema.

Rather than manually creating a new schema, you can use schema discovery to discover and automatically create a schema based on the format of the data from the datasource connected to your adapter. For example, for the Database Input adapter, you can discover a schema that corresponds to a specific table from a database the adapter is connected to.

While using discovery is a convenient way to create your CCL schema, pay particular attention to the datatypes your CCL columns inherit from the external data source. For example, whenever possible, discovery maintains the same level of precision or greater when mapping source data types to ESP data types. Some databases, such as SAP Sybase IQ, support microsecond precision for the `SQL_TIMESTAMP` and `SQL_TYPE_TIMESTAMP` data types. As such, discovery maps these types to the ESP data type `bigdatetime`, which also supports microsecond precision. If your ESP project does not require this level of precision, you can, after generating your schema through discovery, modify the schema to use a lower-precision data type such as `timestamp` (millisecond precision).

To discover a schema, you need to first configure the adapter properties. Each adapter that supports schema discovery has unique properties that must be set to enable schema discovery.

See also

- *Deprecated Adapter Support for Schema Discovery* on page 1024
- *File CSV Input Adapter* on page 1026
- *File XML Input Adapter* on page 1035
- *JMS CSV Input Adapter* on page 1041
- *JMS Object Array Input Adapter* on page 1060
- *JMS XML Input Adapter* on page 1070

Adapter Support for Schema Discovery

Lists all adapters currently available from SAP, whether they support schema discovery, and if so, the properties they use to enable it.

For additional details on the adapter properties, see the specific adapter section.

Adapter	Supports Schema Discovery	Properties
Adaptive Server Enterprise Output	Yes	DB Service Name The name of the database service that represents the ASE database into which information will be loaded.
AtomReader Input	No	—
Database Input	Yes	Database Service Name of database service from which the adapter obtains the database connection.
Database Output	Yes	Database Service Name of service entry to use.
ESP Add-In for Microsoft Excel	No	—
ESP Web Services Provider	No	—
File CSV Input	Yes	File Input Transporter • Dir • File • AccessMode • (Optional) ScanDepth CSV String to ESP Formatter • ExpectStreamNameOpcode
File CSV Output	No	—
File FIX Input	No	—
File FIX Output	No	—
File JSON Input	No	—
File JSON Output	No	—
File XML Document Input	No	—
File XML Document Output	No	—

Adapter	Supports Schema Discovery	Properties
File XML Record Input	Yes	File Input Transporter • Dir • File • AccessMode • (Optional) ScanDepth
File XML Record Output	No	—
FIX Input	No	—
Flex Output	No	—
FTP CSV Input	No	—
FTP CSV Output	No	—
FTP XML Input	No	—
FTP XML Input	No	—
HTTP Output	No	—
JDBC Input	Yes	JDBC Input Transporter • Host • Port • User • Password • DbName • DbType • DbDriver
JDBC Output	—	JDBC Output Transporter • Host • Port • User • Password • DbName • DbType • DbDriver

Adapter	Supports Schema Discovery	Properties
JMS CSV Input	Yes	JMS Input Transporter • ConnectionFactory • JndiContextFactory • JndiURL • DestinationType • DestinationName • MessageType • (Optional) ScanDepth
JMS CSV Output	No	—
JMS FIX Input	No	—
JMS FIX Output	No	—
JMS Object Array Input	No	—
JMS Object Array Output	No	—
JMS XML Input	Yes	JMS Input Transporter • ConnectionFactory • JndiContextFactory • JndiURL • DestinationType • DestinationName • MessageType • (Optional) ScanDepth
JMS XML Output	No	—
KDB Input	Yes	• KDB Server • KDB Port • KDB User • KDB Password
KDB Output	Yes	• KDB Server • KDB Port • KDB User • KDB Password

Adapter	Supports Schema Discovery	Properties
Log File Input	No	—
NYSE Input	Yes	Discovery Directory Path Absolute path to the adapter discovery directory.
Open Input	No	—
Open Output	No	—
Random Tuples Generator Input	No	—
RAP Output	No	—
Replication Server Input	Yes	• RSSD Host • RSSD Port • RSSD Database Name • RSSD User Name • RSSD Password
Reuters Marketfeed Input	Yes	Discovery Path
Reuters Marketfeed Output	No	—
Reuters OMM Input	Yes	Discovery Path
Reuters OMM Output	No	—
RTView Output	No	—
Sample Input	Yes	Discovery Directory Path
Sample Output	Yes	Discovery Directory Path
SAP HANA Output	Yes	DB Service Name The name of the database service that represents the ASE database into which information will be loaded.

Adapter	Supports Schema Discovery	Properties
SAP RFC Input	Yes	• Adapter Configuration File • Adapter Mapping File • SAP Host • SAP System Number • SAP Client • Username • Password
SAP RFC Output	Yes	• Adapter Configuration File • Adapter Mapping File • SAP Host • SAP System Number • SAP Client • Username • Password
SAP Sybase IQ Output	Yes	DB Service Name The name of the database service that represents the IQ database into which information will be loaded.
SMTP Output	No	—
Socket FIX Input	No	—
Socket FIX Output	No	—
Socket CSV Input	No	—
Socket CSV Output	No	—
Socket JSON Input	No	—
Socket JSON Output	No	—
Socket XML Input	No	—
Socket XML Output	No	—
Tibco Rendezvous Input	No	—
Tibco Rendezvous Output	No	—

Adapter	Supports Schema Discovery	Properties
Web Services (SOAP) Input Adapter	Yes	• Adapter Configuration File • Adapter Mapping File • Discovery WSDL URL • Discovery Working Directory • Discovery Service Name **Note:** The adapter does not support schema discovery if you are using HTTP Basic Access Authentication.
Web Services (SOAP) Input Adapter	Yes	• Adapter Configuration File • Adapter Mapping File • Discovery WSDL URL • Discovery Working Directory • Discovery Service Name **Note:** The adapter does not support schema discovery if you are using HTTP Basic Access Authentication.
WebSphere MQ Input	No	—
WebSphere MQ Output	No	—

See also

- *Adaptive Server Enterprise Output Adapter* on page 10
- *File CSV Input Adapter Configuration* on page 44
- *File XML Record Input Adapter Configuration* on page 136
- *JMS CSV Input Adapter Configuration* on page 333
- *SAP RFC Input and Output Adapter* on page 763
- *JDBC Input Adapter Configuration* on page 304
- *JDBC Output Adapter Configuration* on page 316
- *JMS XML Input Adapter Configuration* on page 397
- *KDB Input and Output Adapter* on page 424
- *Database Adapter* on page 21
- *Replication Server Adapter* on page 561
- *Reuters Marketfeed Adapter* on page 585
- *Reuters OMM Adapter* on page 660
- *Sample Input and Output Adapter* on page 751

- *SAP Sybase IQ Output Adapter* on page 810
- *NYSE Technologies Input Adapter* on page 442
- *Web Services (SOAP) Input and Output Adapter* on page 941

CHAPTER 4 Guaranteed Delivery

Guaranteed delivery (GD) is a delivery mechanism that guarantees data is processed from a stream to an adapter.

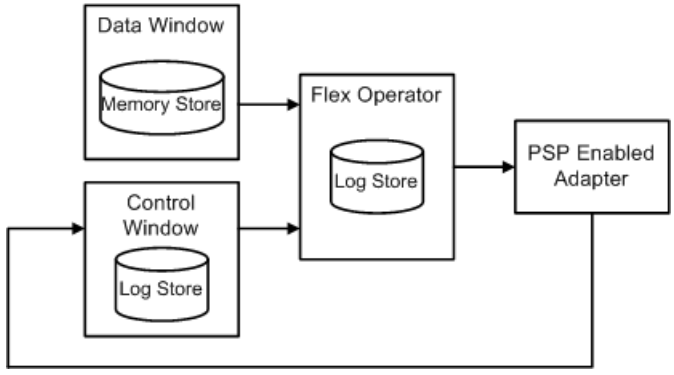
GD ensures that data continues to be processed when:

- The Server fails.
- The destination (third-party server) fails.
- The destination (third-party server) does not respond for a period of time.

Persistent subscribe pattern (PSP) is used to implement GD in output adapters. Input adapters support GD using facilities provided by the source rather than using PSP. The WebSphereMQ Input and Output adapter, all JMS Input and Output adapters, and the TIBCO Rendezvous adapter all support GD. These adapters have specific PSP and GD parameters that are unique to them. Examples for enabling GD in one of the JMS CSV Output adapters and the WebSphere Output adapter are located `<ESP_HOME>/examples/ccl/JmsOutBoundAdapterWithGDSupport` and `<ESP_HOME>/examples/ccl/WsmqOutBoundAdapterWithGDSupport` respectively.

PSP works through a combination of a window (input, output), a control window, and a Flex operator with a log store. The window and control window plug into the Flex operator. Data from the window on which PSP is enabled is entered into the flex operator, which generates a sequence number and opcode from the data, and places them at the beginning of each row of data. The Flex operator sends this data to the adapter that is attached to it, and the adapter passes the information on to the control window. Finally, the control window informs the Flex operator of the data that has been processed by the adapter, and the Flex operator removes this data from the log store.

Figure 16: PSP Overview



See also

- *TIBCO Rendezvous Adapter* on page 921
- *WebSphere MQ Input Adapter* on page 995
- *WebSphere MQ Output Adapter* on page 999
- *JMS CSV Input Adapter* on page 1041
- *JMS CSV Output Adapter* on page 1045
- *JMS Custom Input Adapter* on page 1050
- *JMS Custom Output Adapter* on page 1055
- *JMS FIX Input Adapter* on page 362
- *JMS FIX Output Adapter* on page 366
- *JMS Object Array Input Adapter* on page 1060
- *JMS Object Array Output Adapter* on page 1065
- *JMS XML Input Adapter* on page 1070
- *JMS XML Output Adapter* on page 1074
- *Output Stream Parameters* on page 930
- *SAP RFC Input and Output Adapter* on page 763
- *Web Services (SOAP) Input and Output Adapter* on page 941

Log Window

The log window is a Flex operator that is assigned to a log store and is the centre of the guaranteed delivery (GD) mechanism.

For persistent subscription using persistent subscribe pattern (PSP), attach the output adapter to a log window instead of a stream of interest. The stream definition for the log window contains all the columns belonging to the stream of interest, plus two additional columns. These additional columns are the gdKey (long) and the gdOpcode (integer).

The gdKey is a constantly increasing value that uniquely identifies every event, regardless of the opcode in the stream of interest. This serves as the key for the log window. The gdOpcode is the operation code (for example, INSERT, UPDATE, or DELETE) of the event that occurs in the stream of interest.

The log window takes two inputs namely from the stream whose data needs to be delivered in a guaranteed fashion (stream of interest) and from the truncate window. The log window has a method associated with each input. The method associated with the stream of interest:

1. Increments the gdKey by 1, starting from 0, on every incoming event. On restart, it starts from the last generated sequence number by self inspecting the data it has previously output.
2. Determines the opcode of the incoming event.

3. Outputs the gdKey and the gdOpcode determined in the previous two steps, along with all the columns of the input event from the stream of interest.

The method associated with the truncate window is responsible for ensuring that the data in the log window does not grow indefinitely. Every time an event occurs on the truncate window, this method deletes all events in the log window that has a gdKey less than or equal to the provided gdKey, and provided that the purge data flag is set to true.

Truncate Window

Output adapters use the truncate window to inform the log window of which data has been processed by the adapter and can be safely deleted.

It has three columns, simpleKey (integer), gdKey (long), and purge (boolean). The simpleKey column is currently just a dummy value of 0 or 1, and its sole purpose is to serve as a key for the truncate window. The gdKey column contains the value of the gdKey that the output adapter has successfully processed. The log window uses this to delete all the data that has a gdKey equal to or lesser than the provided value. In the purge column, a value of true indicates that the data in the log window needs to be deleted. The output adapter updates this column.

Assign this window to a log store to ensure data recovery from this window in the case of a failure.

CHAPTER 5 Appendix A: Adapter Parameters Datatypes

A comprehensive list of datatypes you can use with adapters supplied by Event Stream Processor, or any custom internal or external adapters you create.

Some exceptions for custom external adapters are noted in the datatype descriptions.

Note: This table includes all the adapter related datatypes supported by Event Stream Processor. For more information on specific datatypes supported by an adapter, as well as its datatype mapping description, see the section on that adapter.

Datatype	Description
`boolean`	Value is true or false. The format for values outside of the allowed range for `boolean` is 0/1/false/true/y/n/on/off/yes/no, which is case insensitive.
`choice`	A list of custom values from which a user would select one value.
`configFilename`	Variable-length character string, with byte values encoded in UTF-8. Maximum string length is platform-dependent, but no more than 65535 bytes.
`directory`	Variable-length character string, with byte values encoded in UTF-8. Maximum string length is platform-dependent, but no more than 65535 bytes.
`double`	Floating point value. The range of allowed values is 2.22507e-308 to 1.79769e+308
`filename`	Variable-length character string, with byte values encoded in UTF-8. Maximum string length is platform-dependent, but no more than 65535 bytes.
`int`	A signed 32-bit integer value. The range of allowed values is -2147483648 to +2147483647 (-23^1 to 2^31-1). Constant values that fall outside of this range are automatically processed as long datatypes. To initialize a variable, parameter, or column with the lowest negative value, specify (-2...7) -1 instead to avoid CCL compiler errors. For example, specify (-2147483647) -1 to initialize a variable, parameter, or column with a value of -2147483648.

Datatype	Description
`password`	Variable-length character string, with byte values encoded in UTF-8. Maximum string length is platform-dependent, but no more than 65535 bytes. While entering value for this field, user can see '*' for every character.
`permutation`	Variable-length character string, with byte values encoded in UTF-8. Maximum string length is platform-dependent, but no more than 65535 bytes. This datatype is not supported for custom external adapters.
`range`	An integer value for which user can define lower and upper limits. e.g. `<Parameter id="port" label="KDB Port" descr="IP port of the database listener" type="range" rangeLow="0" rangeHigh="65535" default="5001" use="required" />`
`query`	A string value Studio creates from the tablename. This datatype is not supported for custom external adapters.
`runtimeDirectory`	Variable-length character string, with byte values encoded in UTF-8. Maximum string length is platform-dependent, but no less than 65535 bytes. This datatype is not supported for custom external adapters.
`runtimeFilename`	Runtime file name, if different from discovery time file name. Variable-length character string, with byte values encoded in UTF-8. Maximum string length is platform-dependent, but no more than 65535 bytes. This datatype is not supported for custom external adapters.
`string`	Variable-length character string, with byte values encoded in UTF-8. Maximum string length is platform-dependent, but no more than 65535 bytes.
`tables`	This is list of choices returned by **getTables()** defined in adapter.
`text`	A value capable of storing multiline text. This datatype is not supported for custom external adapters.
`uint`	Positive integer value. The range of allowed values is 0 to 0xffffffff.

CHAPTER 6

Appendix B: Date and Timestamp Formats for Input Adapters

SAP supports numerous formats for date and timestamp datatypes.

Use the info below to create a custom format for your date and timestamp datatypes.

Character	Description
%a	The day of the week, using the locale's weekday names. You can specify either the abbreviated or full name.
%A	Equivalent to %a.
%b	The month, using the locale's month names. You can specify either the abbreviated or full name.
%B	Equivalent to %b.
%c	The locale's appropriate date and time representation.
%C	The century number [00,99]. Leading zeros are permitted but not required.
%d	The day of the month [01,31]. Leading zeros are permitted but not required.
%D	The date as %m / %d / %y.
%e	Equivalent to %d.
%h	Equivalent to %b.
%H	The hour (24-hour clock) [00,23]. Leading zeros are permitted but not required.
%I	The hour (12-hour clock) [01,12]. Leading zeros are permitted but not required.
%j	The day number of the year [001,366]. Leading zeros are permitted but not required.
%m	The month number [01,12]. Leading zeros are permitted but not required.
%M	The minute [00,59]. Leading zeros are permitted but not required.
%n	Any white space.

Character	Description
%p	The locale's equivalent of a.m or p.m.
%r	12-hour clock time using the AM/PM notation.
%R	The time as %H : %M.
%S	The seconds [00,60]. Leading zeros are permitted but not required.
%t	Any white space.
%T	The time as %H : %M : %S.
%U	The week number of the year (Sunday as the first day of the week) as a decimal number [00,53]. Leading zeros are permitted but not required.
%w	The weekday as a decimal number [0,6], with 0 representing Sunday. Leading zeros are permitted but not required.
%W	The week number of the year (Monday as the first day of the week) as a decimal number [00,53]. Leading zeros are permitted but not required.
%x	The date, using the locale's date format.
%X	The time, using the locale's time format.
%y	The year within the century. When a century is not otherwise specified, values in the range [69,99] shall refer to years 1969 to 1999 inclusive, and values in the range [00,68] shall refer to years 2000 to 2068 inclusive. Leading zeros are permitted but not required.
%Y	The year, including the century. For example, 1988.
%%	Replaced by %.

CHAPTER 7

Appendix C: Date and Timestamp Formats for Output Adapters

SAP supports numerous formats for date and timestamp datatypes.

Use the info below to create a custom format for your date and timestamp datatypes.

Character	Description
%a	The locale's abbreviated weekday name.
%A	The locale's full weekday name.
%b	The locale's abbreviated month name.
%B	The locale's full month name.
%c	The locale's appropriate date and time representation.
%C	The year divided by 100, and truncated to an integer as a decimal number [00,99].
%d	The day of the month as a decimal number [01,31].
%D	Equivalent to %m / %d / %y.
%e	The day of the month as a decimal number [1,31]. A single digit is preceded by a space.
%F	Equivalent to %Y - %m - %d. This is the ISO 8601:2000 standard date format.
%g	The last 2 digits of the week-based year, as a decimal number [00,99].
%G	The week-based year as a decimal number. For example, 1977.
%h	Equivalent to %b.
%H	The hour (24-hour clock) as a decimal number [00,23].
%I	The hour (12-hour clock) as a decimal number [01,12].
%j	The day of the year as a decimal number [001,366].
%m	The month as a decimal number [01,12].
%M	The minute as a decimal number [00,59].

Character	Description
%n	A <newline>.
%p	The locale's equivalent of either a.m. or p.m.
%r	The time in a.m. and p.m. notation.
%R	The time in 24-hour notation (%H : %M).
%S	The second as a decimal number [00,60].
%t	A <tab>.
%T	The time in the following format %H : %M : %S.
%u	The weekday as a decimal number [1,7], with 1 representing Monday.
%U	The week number of the year as a decimal number [00,53]. The first Sunday of January is the first day of week 1, and days in the new year before this are in week 0.
%V	The week number of the year (Monday as the first day of the week) as a decimal number [01,53]. If the week containing 1 January has four or more days in the new year, then it is considered week 1. Otherwise, it is the last week of the previous year, and the next week is week 1. Both January 4th and the first Thursday of January are always in week 1.
%w	The weekday as a decimal number [0,6], with 0 representing Sunday.
%W	The week number of the year as a decimal number [00,53]. The first Monday of January is the first day of week 1, and days in the new year before this are in week 0.
%x	The locale's appropriate date representation.
%X	The locale's appropriate time representation.
%y	The last two digits of the year as a decimal number [00,99].
%Y	The year as a decimal number. For example, 1997.
%z	The offset from UTC in the ISO 8601:2000 standard format (+hhmm or -hhmm), or by no characters if no time zone is determinable. For example, "-0430" means 4 hours 30 minutes behind UTC (west of Greenwich).
%Z	The time zone name or abbreviation, or by no bytes if no time zone information exists.
%%	Replaced by %.

CHAPTER 8 Appendix D: Deprecated Adapters

These adapters are deprecated and have been replaced by new pre-configured adapters that are built based on the new ESP adapter toolkit. While these deprecated adapters are still currently available for backwards compatibility, it is recommended that you switch to the replacement adapters as these offer additional features and will continue to be enhanced through future releases.

Deprecated Adapter Summary

Summary of deprecated adapters, their type, whether they are managed, studio configurablem and whether it supports guaranteed delivery.

Adapter	Type	Managed	Studio Configurable	Supports Guaranteed Delivery
File CSV Input	Internal	Yes	Yes	No
File CSV Output	Internal	Yes	Yes	No
File XML Input	Internal	Yes	Yes	No
File XML Output	Internal	Yes	Yes	No
JMS CSV Input	Internal	Yes	Yes	Yes
JMS CSV Output	Internal	Yes	Yes	Yes
JMS Custom Input	Internal	Yes	Yes	Yes
JMS Custom Output	Internal	Yes	Yes	Yes
JMS Object Array Input	Internal	Yes	Yes	Yes
JMS Object Array Output	Internal	Yes	Yes	Yes
JMS XML Input	Internal	Yes	Yes	Yes

Adapter	Type	Managed	Studio Configurable	Supports Guaranteed Delivery
JMS XML Output	Internal	Yes	Yes	Yes
Socket (as Client) CSV Input	Internal	Yes	Yes	No
Socket (as Client) CSV Output	Internal	Yes	Yes	No
Socket (as Client) XML Input	Internal	Yes	Yes	No
Socket (as Client) XML Output	Internal	Yes	Yes	No
Socket (as Server) CSV Input	Internal	Yes	Yes	No
Socket (as Server) CSV Output	Internal	Yes	Yes	No
Socket (as Server) XML Input	Internal	Yes	Yes	No
Socket (as Server) XML Output	Internal	Yes	Yes	No

Deprecated Adapter Support for Schema Discovery

Lists deprecated adapters and whether they support schema discovery, and if so, the properties they use to enable it.

Adapter	Supports Schema Discovery	Properties
File CSV Input	Yes	Directory The absolute path to the data files you want the adapter to read.
File CSV Output	No	—

Adapter	Supports Schema Discovery	Properties
File XML Input	Yes	Directory The absolute path to the data files you want the adapter to read.
File XML Output	No	—
JMS CSV Input	Yes	• Delimiter – field delimiter • Connection Factory – connection factory class name • JNDI Context Factory – context factory for JNDI context initialization • JNDI URL • Destination Type • Destination Name
JMS CSV Output	No	—
JMS Custom Input	No	—
JMS Custom Output	No	—
JMS Object Array Input	Yes	• Connection Factory – connection factory class name • JNDI Context Factory – context factory for JNDI context initialization • JNDI URL • Destination Type • Destination Name
JMS Object Array Output	No	—
JMS XML Input	Yes	• Connection Factory – connection factory class name. • JNDI Context Factory – context factory for JNDI context initialization • JNDI URL • Destination Type • Destination Name
JMS XML Output	No	—

Adapter	Supports Schema Discovery	Properties
Socket (as Client) CSV Input	No	—
Socket (as Client) CSV Output	No	—
Socket (as Client) XML Input	No	—
Socket (as Client) XML Output	No	—
Socket (as Server) CSV Input	No	—
Socket (as Server) CSV Output	No	—
Socket (as Server) XML Input	No	—
Socket (as Server) XML Output	No	—

See also

- *Chapter 3, Schema Discovery* on page 1005
- *File CSV Input Adapter* on page 1026
- *File XML Input Adapter* on page 1035
- *JMS CSV Input Adapter* on page 1041
- *JMS Object Array Input Adapter* on page 1060
- *JMS XML Input Adapter* on page 1070

File CSV Input Adapter

Adapter type: dsv_in. The File CSV Input adapter reads a file in Event Stream Processor delimited format.

Use this adapter to poll new data appended to the data file. The file does not require a header. If the file includes a header, it specifies the field names.

Sample record formats for the data file:

```
1. hasHeader=true
delimiter=,
expectStreamNameOpcode=false

Ts,ItemID,Price,Quantity,WarehouseZipCode,DeliveryZipCode
2004/06/17 10:00:00.000000,SKU1276532,50.00,1,10012,94086
2004/06/17 10:00:05.000000,SKU6723143,23.00,2,10012,94043

2. expectStreamNameOpcode=true
delimiter=,

Trades_in,i,2004/06/17
10:00:00.000000,SKU1276532,50.00,1,10012,94086
Trades_in,i,2004/06/17
```

```
10:00:05.000000,SKU6723143,23.00,2,10012,94043

3. expectStreamNameOpcode=false
timestampFormat=%Y/%m/%d %H:%M:%S
delimiter=,

2004/06/17 10:00:00.000000,SKU1276532,50.00,1,10012,94086
2004/06/17 10:00:05.000000,SKU6723143,23.00,2,10012,94043
```

This adapter supports schema discovery in normal mode only. If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

The File CSV Input adapter operates in two loading modes: dynamic and normal. In normal mode, the adapter reads records from a single source file until there are no more records or you stop the process. In dynamic mode, the adapter reads records from a source directory on a first-come, first-served basis. These files follow a format defined by a regular expression or the **filePattern** parameter. You can also specify a POSIX standard regular expression, such as [a-z]{4}\.csv, or a wildcard character sequence that is supported by the native operating system, such as *.csv. For example, if the expression is *.csv, then the adapter searches for .csv files only. After reading each file, the adapter checks the directory for new files and continues processing. If there is an error in opening or reading a file, the adapter skips it.

In dynamic mode, if a file is overwritten or updated after the adapter processes it and the **reprocess** parameter is set to false, the file it is not processed again. If the **removeAfterProcess** parameter is set to false, the adapter does not delete the file after it processes it. Both **reprocess** and **removeAfterProcess** can be set to false simultaneously, but only one can be set to true at a time.

Property Label	Description
Directory	Property ID: **dir** Type: `string` (Required for adapter operation and schema discovery) Specify the absolute path to the data files you want the adapter to read. For example, `<username>/<folder name>`. Default value is "." (current directory in which the Server is running). Use a forward slash for both UNIX and Windows paths.
File (in Directory)	Property ID: **file** Type: `tables` (Dependent required) File to read. Set only in normal mode; in dynamic mode, the value is ignored. No default value..

Property Label	Description
Stream name, opcode expected	Property ID: **expectStreamNameOpcode** Type: `boolean` (Required) If true, the adapter interprets the first two fields as stream name and opcode respectively. Messages with unmatched stream names are discarded. When you are using schema discovery on this adapter with this property enabled, two columns for the stream name and opcode are created in the schema. Manually remove these two columns from your schema. Default value is false.
Field Count	Property ID: **fieldCount** Type: `uint` (Optional) Count of fields in CSV file, if different from the value for the source stream. Default value is 0.
Repeat Date Field Count	Property ID: **repeatCount** Type: `int` (Optional) Number of times the input data is repeated. If set to -1, the input data is repeated indefinitely. Default value is 0. **Note:** You can use this parameter to test a continuous streaming source.
Repeat Data Field Name	Property ID: **repeatField** Type: `string` (Optional) On each repeat, increment the value in this field. You must specify the stream column if **repeatCount** has a nonzero value. Default value is a hyphen (-). • If **repeatCount** has a nonzero value, specify the stream column name. • If the repeatColumn is a key column in the stream, ensure there are no duplicates when specifying multiple rows in the input file. • If the adapter is attached to a window, the **repeatField** must be a key column.

Property Label	Description
Delimiter	Property ID: **delimiter** Type: `string` (Advanced) Symbol used to separate the column. Default value is a comma (,).
Has Header	Property ID: **hasHeader** Type: `boolean` (Advanced) Determines whether the first line of the file contains the description of the fields. Default value is false .
Directory (runtime)	Property ID: **runtimeDir** Type: `runtimeDirectory` (Advanced) Location of the data files at runtime, if the value is different from the location defined at discovery time. No default value.
File Pattern	Property ID: **filePattern** Type: `string` (Advanced) In dynamic mode, the **filePattern** looks up files in a directory. In normal mode, it is the pattern used to look up files for discovery. If both **filePattern** and **fileRegex** are specified, **filePattern** is ignored and **fileRegex** is used. **filePattern** allows wildcard characters supported by the native operating system. Default value is `*.csv`.
Regular Expression	Property ID: **fileRegex** Type: `string` (Optional) In dynamic mode, the regular expression property allows the user to specify a POSIX regular expression standard to find matching file names in the directory. If both **filePattern** and **fileRegex** are specified, then **filePattern** is ignored and **fileRegex** is used. No default value.

Property Label	Description
Poll Period (seconds)	Property ID: **pollperiod** Type: `uint` (Advanced) In normal mode, specifies the period for polling a file to check for new records. **pollperiod** is ignored in dynamic mode. Default value is 0.
Directory Poll Period	Property ID:**dirPollPeriod** Type: `uint` (Advanced) In dynamic mode, continuously checks for new files in the directory. Default value is 60 seconds.
Convert to Safe Opcodes	Property ID: **safeOps** Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT, and converts DELETE to SAFEDELETE. Default value is false.
Skip Deletes	Property ID: **skipDels** Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. Default value is false.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Format string for parsing date values. Default value is `%Y-%m-%dT%H:%M:%S`.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Format string for parsing timestamp values. Default value is `%Y-%m-%dT%H:%M:%S`.
Block Size	Property ID: **blockSize** Type: `int` (Advanced) Number of records to block into one pseudotransaction. Default value is 1.

Property Label	Description
Use Envelopes	Property ID: **useEnvelopes** Type: `boolean` (Advanced) Specify the block type the adapter uses to pass data to the engine. If you specify a **blockSize** property greater than zero, by default, the adapter packages rows into transaction blocks to send to the engine. To get the adapter to package rows into envelope blocks instead, set this property to true. Default value is false.
Field Mapping	Property ID: **permutation** Type: `permutation` Mapping between Event Stream Processor and external fields, for example: <esp_columnname>=<database_columnname>:<esp_columnname>=<database_columnname>. No default value.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.
Dynamic Loading	Property ID: **dynamicMode** Type: `boolean` (Optional) Set to true to enable dynamic loading mode, which causes the adapter to read records from files as they arrive in the directory. Records are processed one after another on a first-come, first-serve basis. Default value is false.

Property Label	Description
Remove File After Processing	Property ID: **removeAfterProcess** Type: `boolean` (Optional) In dynamic mode, if **removeAfterProcess** is set to true, the file is removed from the directory after the adapter processes it. You cannot set this property to true if **reprocess** is also set to true. Default value is false .
Reprocess File	Property ID: **reprocess** Type: `boolean` (Optional) In dynamic mode, if **reprocess** is set to true, the file is processed again after being updated or overwritten. You cannot set this property to true if **removeAfterProcess** is also set to true. Default value is false.

Known limitations:

- In normal mode only, when polling, you can append to the file, but cannot overwrite or replace it. The stream names in the file rows are ignored and all data is sent to the same stream.
- For discovery to work correctly, set the delimiter character and the header presence flag to match the actual data.
- Do not mix files with different delimiters or files with and without headers in the same directory. Files with wrong delimiters or headers are incorrectly discovered.

See also

- *Deprecated Adapter Support for Schema Discovery* on page 1024
- *Chapter 3, Schema Discovery* on page 1005

File CSV Output Adapter

Adapter type: dsv_out. The File CSV Output adapter writes data as a file in Event Stream Processor delimited format.

The file does not require a header. If the file includes a header, it specifies the field names. This adapter does not support schema discovery.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Directory	Property ID: **dir** Type: `directory` (Required) Specify the absolute path to the data files you want the adapter to read. For example, `<username>/<folder name>`. No default value. Use a forward slash for both UNIX and Windows paths.
File (in Directory)	Property ID: **file** Type: `tables` (Required) File to which the adapter writes data. No default value.
Include Base Content	Property ID:**outputBase** Type: `boolean` (Optional) Records the initial contents of the stream, not just the updates. Default value is false.
Only Base Content	Property ID: **onlyBase** Type: `boolean` (Optional) Sends a one-time snapshot of initial contents in a stream. Default value is false.
Prepend StreamNameOpcode	Property ID: **prependStreamNameOpcode** Type: `boolean` (Optional) If true, each merge starts the stream name and the opcode. Default value is false.
Delimiter	Property ID: **delimiter** Type: `string` (Advanced) Symbol used to separate the columns. Default value is a comma (,).

Property Label	Description
Has Header	Property ID: **hasHeader** Type: `boolean` (Advanced) Determines whether the first line of the file contains the description of the fields. Default value is false .
Directory (runtime)	Property ID: **runtimeDir** Type: `runtimeDirectory` (Advanced) Location of the data files at runtime, if different from discovery time. No default value.
File Pattern	Property ID: **filePattern** Type: `string` (Advanced) Pattern used to look up files for discovery. Default value is `*.csv` .
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Format string to parse data values. Default value is `%Y-%m-%dT%H:%M:%S`.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Format string to parse timestamp values. Default value is `%Y-%m-%dT%H:%M:%S`.
Field Mapping	Property ID: **permutation** Type: `permutation` Mapping between Event Stream Processor and external fields, for example: <esp_columnname>=<database_columnname>:<esp_columnname>=<database_columnname>. No default value.

Property Label	Description
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

File XML Input Adapter

Adapter type: xml_in. The File XML Input adapter reads a file in XML format.

This adapter polls for new data appended to a file, and supports schema discovery.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Sample record format for the data file:

```
<Trades Id="0" Symbol="EBAY" TradeTime="2000-05-04T12:00:00"
Price="140.0" Shares="50" />
<Trades Id="1" Symbol="EBAY" TradeTime="2000-05-04T12:00:01"
Price="150.0" Shares="500" />
```

Sample record format for the data file with an opcode:

```
<Trades ESP_OPS="i" Id="0" Symbol="EBAY"
TradeTime="2000-05-04T12:00:00" Price="140.0" Shares="50" />
```

Note: To insert a record, specify ESP_OPS="i". To update a record, specify ESP_OPS="u". To delete a record, specify ESP_OPS="d". If no opcode is specified, the record is interpreted as an upsert.

Property Label	Description
Directory	Property ID: **dir** Type: `directory` (Required for adapter use and schema discovery) Specify the absolute path to the data files you want the adapter to read. For example, `<username>/<folder name>`. No default value. Use a forward slash for both UNIX and Windows paths.
File (in Directory)	Property ID: **file** Type: `tables` (Required) File to which the adapter writes data. No default value.
Match Stream Name	Property ID: **matchStreamName** Type: `boolean` (Optional) If true, XML element name is matched against the stream name. Unmatched messages are discarded. Default value is false.
Repeat Count	Property ID: **repeatCount** Type: `uint` (Optional) Number of times the input data is repeated. If set to -1, the input data is repeated indefinitely. Default value is 0. **Note:** You can use this parameter to test a continuous streaming source.

Property Label	Description
Repeat Field	Property ID: **repeatField** Type: `string` (Optional) Determines which numeric field's values are bumped on each repeat. Default value is a hyphen (-). • If **repeatCount** has a nonzero value, specify the stream column name. • If the repeatColumn is a key column in the stream, ensure there are no duplicates when specifying multiple rows in the input file. • If the adapter is attached to a window, the **repeatField** must be a key column.
Directory (runtime)	Property ID: **runtimeDir** Type: `runtimeDirectory` (Advanced) Location of the data files at runtime, if different from discovery time. No default value.
File Pattern	Property ID: **filePattern** Type: `string` (Advanced) Pattern used to look up files for discovery. Default value is `*.xml` .
Poll Period (seconds)	Property ID: **pollperiod** Type: `uint` (Advanced) Period for polling new contents, in seconds. Default value is 0.
Convert to Safe Opcodes	Property ID: **safeOps** Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT. Converts DELETE to SAFEDELETE. Default value is false.

Property Label	Description
Skip Deletes	Property ID: **skipDels** Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. Default value is false.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Format string for parsing data values. Default value is %Y-%m-%dT%H:%M:%S.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Format string for parsing timestamp values. Default value is %Y-%m-%dT%H:%M:%S .
Block Size	Property ID: **blockSize** Type: `int` (Advanced) Determines the number of records to block into one pseudotransaction. Default value is 1 .
Use Envelopes	Property ID: **useEnvelopes** Type: `boolean` (Advanced) Specify the block type the adapter uses to pass data to the engine. If you specify a **blockSize** property greater than zero, by default, the adapter packages rows into transaction blocks to send to the engine. To get the adapter to package rows into envelope blocks instead, set this property to true. Default value is false.

Property Label	Description
Field Mapping	Property ID: **permutation** Type: `permutation` Mapping between Event Stream Processor and external fields, for example: <esp_columnname>=<database_column-nname>:<esp_columnname>=<database_column-nname>. No default value.
PropertySet	Property ID:**propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Known limitations:

- When polling, you can append to a file, but cannot overwrite or replace it.
- The stream name in the file entries is ignored.
- Do not mix data files and model XML files in the same directory. This causes Event Stream Processor XML files to be discovered as invalid.

See also

- *Deprecated Adapter Support for Schema Discovery* on page 1024
- *Chapter 3, Schema Discovery* on page 1005

File XML Output Adapter

Adapter type: xml_out. The File XML Output adapter writes data as a file in XML format.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Directory	Property ID: **dir** Type: `directory` (Required) Specify the absolute path to the data files you want the adapter to read. For example, `<username>/<folder name>`. No default value. Use a forward slash for both UNIX and Windows paths.
File (in Directory)	Property ID: **file** Type: `tables` (Required) File the adapter writes data to. No default value.
Include Base Content	Property ID: **outputBase** Type: `boolean` (Optional) Records the initial contents of the stream, not just the updates. Default value is false.
Only Base Content	Property ID: **onlyBase** Type: `boolean` (Optional) Sends a one-time snapshot of initial contents of the stream. Default value is false.
Directory (runtime)	Property ID: **runtimeDir** Type: `runtimeDirectory` (Advanced) Location of the data files at runtime, if different from discovery time. No default value.
File Pattern	Property ID: **filePattern** Type: `string` (Advanced) Pattern used to look up files for discovery. Default value is `*.xml`.

Property Label	Description
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Format string for parsing data values. Default value is `%Y-%m-%dT%H:%M:%S`.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Format string for parsing timestamp values. Default value is `%Y-%m-%dT%H:%M:%S`.
Field Mapping	Property ID: **permutation** Type: `permutation` Mapping between Event Stream Processor and external fields, for example: <esp_columnname>=<database_columnname>:<esp_columnname>=<database_columnname>. No default value.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

JMS CSV Input Adapter

Adapter type: jms_csv_in. The JMS CSV Input adapter subscribes to text messages formatted as a delimited list of values, and writes them as stream records.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

When the `delimiter` is set to a comma (,) and the `expectStreamNameOpcode` is set to true, the JMS CSV Input adapter expects the input stream to be formatted as:

```
aaa,
11,111,1.100000,2008-03-13T08:19:30,111.1111,2008-03-13T08:19:30.12
3,false,FF00FE05FF,
2008-03-13T08:19:30.123456,64000,922.0,337.000000000000000
```

Property Label	Description
Delimiter	Property ID: **delimiter** Type: `string` (Required for adapter operation and schema discovery) Field delimiter. Default value is a comma (,).
Connection Factory	Property ID: **connectionFactory** Type: `string` (Required for adapter operation and schema discovery) Connection factory class name. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ™ –** ConnectionFactory • **TIBCO –** QueueConnectionFactory • **WebSphere MQ –** MyMQConnFactory No default value.
JNDI Context Factory	Property ID: **jndiContextFactory** Type: `string` (Required for adapter operation and schema discovery) Context factory for JNDI context initialization. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** org.apache.activemq.jndi.ActiveMQInitialContextFactory • **TIBCO –** com.tibco.tibjms.naming.TibjmsInitialContextFactory • **WebSphere MQ –** com.sun.jndi.fscontext.RefFSContextFactory No default value.

Property Label	Description
JNDI URL	Property ID: **jndiURL** Type: `string` (Required for adapter operation and schema discovery) JNDI URL. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** tcp://server:61616 • **TIBCO –** tibjmsnaming://server:7222 • **WebSphere MQ –** file:/var/mqm/jndi/ WebSphere MQ is different as it requires a separate naming server to be configured with it. By default, WebSphere MQ only provides a file-based naming server. No default value.
Destination Type	Property ID: **destinationType** Type: `choice` (Required for adapter operation and schema discovery) Destination type. Valid values are: • QUEUE • TOPIC Default value is QUEUE.
Destination Name	Property ID: **destinationName** Type: `string` (Required for adapter operation and schema discovery) Destination name. No default value.
Subscription Mode	Property ID: **subscriptionMode** Type: `choice` (Optional) Specifies the subscription mode for TOPIC. Default value is NONDURABLE. Valid values are DURABLE and NONDURABLE.

Property Label	Description
Client ID	Property ID: **clientID** Type: `string` (Required if subscription mode is set to DURABLE) Specifies the client identifier for a JMS client. Required for creating a durable subscription in JMS. Can be any string, but must be unique for each topic. No default value.
Subscription Name	Property ID: **subscriptionName** Type: `string` (Required if subscription mode is set to DURABLE) Specifies a unique name identifying a durable subscription. Required for creating a durable subscription in JMS. Can be any string, but must be unique within a given client ID. No default value.
Batch Size	Property ID: **batchsize** Type: `uint` (Optional) Specifies number of records in a batch to commit in durable subscription mode. Default value is 1.
Stream Name Opcode Expected	Property ID: **expectStreamNameOpcode** Type: `boolean` (Advanced) If true, the first two fields in CSV records are interpreted as stream name, and opcode. An empty string is a valid value. Default value is false.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Date format. Default value is YYYY-MM-DDTHH:MM:SS.SSS .
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Timestamp format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.

Property Label	Description
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Known limitations:

- If the connection to the message broker is lost, the adapter does not attempt to reconnect.

See also

- *Chapter 4, Guaranteed Delivery* on page 1013
- *Deprecated Adapter Support for Schema Discovery* on page 1024
- *Chapter 3, Schema Discovery* on page 1005

JMS CSV Output Adapter

Adapter type: jms_csv_out. The JMS CSV Output adapter publishes stream data as text messages formatted as a delimited list of values to a JMS queue or topic.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Delimiter	Property ID: **delimiter** Type: `string` (Required) Field delimiter. Default value is a comma (,).

Property Label	Description
Connection Factory	Property ID: **connectionFactory** Type: `string` (Required) Connection factory class name. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** ConnectionFactory • **TIBCO –** QueueConnectionFactory • **WebSphere MQ –** MyMQConnFactory No default value.
JNDI Context Factory	Property ID: **jndiContextFactory** Type: `string` (Required) Context factory for JNDI context initialization. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** org.apache.activemq.jndi.ActiveMQInitial-ContextFactory • **TIBCO –** com.tibco.tibjms.naming.TibjmsInitialContext-Factory • **WebSphere MQ –** com.sun.jndi.fscontext.RefFSContext-Factory No default value.
JNDI URL	Property ID: **jndiURL** Type: `string` (Required) JNDI URL. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** tcp://server:61616 • **TIBCO –** tibjmsnaming://server:7222 • **WebSphere MQ –** file:/var/mqm/jndi/ WebSphere MQ is different as it requires a separate naming server to be configured with it. By default, WebSphere MQ only provides a file-based naming server. No default value.

Property Label	Description
Destination Type	Property ID: **destinationType** Type: `choice` (Required)Destination type. Valid values are: • QUEUE • TOPIC Default value is QUEUE.
Destination Name	Property ID: **destinationName** Type: `string` (Required) Destination name. No default value.
Delivery Mode	Property ID: **deliveryMode** Type: `choice` (Optional) Type of delivery mode. Valid values are: • PERSISTENT • NON_PERSISTENT Default value is PERSISTENT.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Property Label	Description
Column To Message Property Map	Property ID: **columnPropertyMap** Type: `string` (Advanced) A comma-delimited list of `ColumnName=PropertyName` mappings that enables message filtering on the message broker side using the JMS selector mechanism. Ensure that there are no spaces in the value of this property. For each mapped column name, the outbound message is paired with a corresponding JMS property that has a value equal to the column value. `ColumnName1=PropertyName1,ColumnName2=PropertyName2...` No default value.
Prepend Stream Name, Opcode	Property ID: **prependStreamNameOpcode** Type: `boolean` (Advanced) If true, every CSV record is prepended with stream name, and opcode. No default value.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Date format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Timestamp format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.
Runs Adapter in GD Mode	Property ID: **enableGDMode** Type: `boolean` (Advanced) If set to true, the adapter runs in guaranteed delivery (GD) mode and all GD-related parameters become required. Default value is false.

Property Label	Description
Name of Column Holding GD Key	Property ID: **gdKeyColumn** Type:`string` (Advanced) Specifies column name in the Flex operator holding the GD key. The GD key is a constantly increasing value that uniquely identifies every event regardless of the opcode in the stream of interest. No default value.
Name of Column Holding opcode	Property ID: **gdOpcodeColumn** Type: `string` (Advanced) Specifies name of column in Flex operator holding opcode. The opcode is the operation code (for example, inserts, update, or delete) of the event occurring in the stream of interest. No default value.
Name of Truncate Stream	Property ID: **gdControlStream** Type: `string` (Advanced) Specifies name of the control window in the GD setup. The control window is a source stream that informs the Flex operator of which data has been processed by the adapter and can be safely deleted. No default value.
Purge After Number of Records	Property ID: **gdPurgeInternal** Type: `int` (Advanced) Specifies number of records after which to purge the Flex operator. Default value is 1000.
Batch Size to Update Truncate Stream	Property ID: **gdBatchSize** Type: `int` (Advanced) Specifies number of records after which the control window must be updated with the latest GD key. Default value is 1000.

Known limitations:

- If the connection to the message broker is lost, the adapter does not attempt to reconnect.

See also

- *Chapter 4, Guaranteed Delivery* on page 1013

JMS Custom Input Adapter

Adapter type: jms_custom_in. The JMS Custom Input adapter subscribes to custom-formatted Java object messages from a JMS queue or topic, and writes them as stream records.

A custom-provided implementation performs the format conversions of this interface:

```
package com.sybase.esp.adapters;
public interface ExternalToESPConverter {
    public ESPMessage externalToESP(Serializable externalMessage)
throws Exception;
}
```

Ensure that the objects returned by the `externalToESP` method implement this interface:

```
package com.sybase.esp.adapters;
public interface ESPMessage extends Serializable {
    public String getStreamName();
    public String getOpCode();
    public Map<String, Serializable> getColumnValues();
}
```

The objects returned by the `getStreamName`, `getOpCode`, `getColumnValues` methods are interpreted as the name of the stream to write to, the opcode, and the stream record as a column to message property map value.

Ensure that stream column types correspond to Java classes as follows:

Stream Column Type	Java Class
bigdatetime	java.lang.Double
binary	java.lang.String
boolean	java.lang.Boolean
integer	java.lang.Integer
interval	java.lang.Long
date	java.util.Date
float	java.lang.Double
long	java.lang.Long
money1	java.math.BigDecimal
money2	java.math.BigDecimal
money3	java.math.BigDecimal

Stream Column Type	Java Class
money4	java.math.BigDecimal
money5	java.math.BigDecimal
money6	java.math.BigDecimal
money7	java.math.BigDecimal
money8	java.math.BigDecimal
money9	java.math.BigDecimal
money10	java.math.BigDecimal
money11	java.math.BigDecimal
money12	java.math.BigDecimal
money13	java.math.BigDecimal
money14	java.math.BigDecimal
money15	java.math.BigDecimal
string	java.lang.String
timestamp	java.util.Date

Ensure that implementations of the ExternalToEFSConverter interface provide a constructor with a single argument of `java.lang.String` type, or a default constructor with no arguments.

Note: Records with unmatched stream names are ignored. Records with null opcodes are interpreted as upserts. The values of non-key columns may be absent or null.

If an implementation is not provided, the default implementation is used, which interprets each external message as an instance of the DefaultEFSMessage class and does not perform a conversion.

Ensure that a Java archive containing an implementation of the ExternalToEFSConverter interface is provided, and place it in the `lib` subfolder of the Event Stream Processor installation folder.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Note: This adapter supports schema discovery.

Property Label	Description
Converter Class Name	Property ID: **converterClassName** Type: `string` (Required) External to ESP message converter fully qualified class name. No default value.
Connection Factory	Property ID: **connectionFactory** Type: `string` (Required) Connection factory class name. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** ConnectionFactory • **TIBCO –** QueueConnectionFactory • **WebSphere MQ –** MyMQConnFactory No default value.
JNDI Context Factory	Property ID: **jndiContextFactory** Type: `string` (Required) Context factory for JNDI context initialization. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** org.apache.activemq.jndi.ActiveMQInitialContextFactory • **TIBCO –** com.tibco.tibjms.naming.TibjmsInitialContextFactory • **WebSphere MQ –** com.sun.jndi.fscontext.RefFSContextFactory No default value.

Property Label	Description
JNDI URL	Property ID: **jndiURL** Type: `string` (Required) JNDI URL. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** tcp://server:61616 • **TIBCO –** tibjmsnaming://server:7222 • **WebSphere MQ –** file:/var/mqm/jndi/ WebSphere MQ is different as it requires a separate naming server to be configured with it. By default, WebSphere MQ only provides a file-based naming server. No default value.
Destination Type	Property ID: **destinationType** Type: `choice` (Required) Destination type. Valid values are: • QUEUE • TOPIC Default value is QUEUE.
Destination Name	Property ID: **destinationName** Type: `string` (Required) Destination name. No default value.
Converter Parameter	Property ID: **converterParam** Type: `string` (Optional) External to Event Stream Processor message converter start-up parameter. No default value.
Subscription Mode	Property ID: **subscriptionMode** Type: `choice` (Optional) Specifies the subscription mode for TOPIC. Default value is NONDURABLE. Valid values are DURABLE and NONDURABLE.

Property Label	Description
Client ID	Property ID: **clientID** Type: `string` (Required if subscription mode is set to DURABLE) Specifies the client identifier for a JMS client. Required for creating a durable subscription in JMS. Can be any string, but must be unique for each topic. No default value.
Subscription Name	Property ID: **subscriptionName** Type: `string` (Required if subscription mode is set to DURABLE) Specifies a unique name identifying a durable subscription. Required for creating a durable subscription in JMS. Can be any string, but must be unique within a given client ID. No default value.
Batch Size	Property ID: **batchsize** Type: `uint` (Optional) Specifies number of records in a batch to commit in durable subscription mode. Default value is 1.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Known limitations:

- If the connection to the message broker is lost, the adapter does not attempt to reconnect.

See also

- *Chapter 4, Guaranteed Delivery* on page 1013

JMS Custom Output Adapter

Adapter type: jms_custom_out. The JMS Custom Output adapter publishes stream records as custom-formatted Java objects to a JMS queue or topic.

A custom-provided implementation performs the format conversions of this interface:

```
package com.sybase.esp.adapters;
public interface ESPToExternalConverter {
    public Serializable ESPToExternal(ESPMessage ESPMessage) throws
Exception;
}
```

Ensure that stream column types correspond to Java classes as follows:

Stream Column Type	Java Class
bigdatetime	java.lang.Double
binary	java.lang.String
boolean	java.lang.Boolean
integer	java.lang.Integer
interval	java.lang.Long
date	java.util.Date
float	java.lang.Double
long	java.lang.Long
money1	java.math.BigDecimal
money2	java.math.BigDecimal
money3	java.math.BigDecimal
money4	java.math.BigDecimal
money5	java.math.BigDecimal
money6	java.math.BigDecimal
money7	java.math.BigDecimal
money8	java.math.BigDecimal
money9	java.math.BigDecimal

Stream Column Type	Java Class
`money10`	`java.math.BigDecimal`
`money11`	`java.math.BigDecimal`
`money12`	`java.math.BigDecimal`
`money13`	`java.math.BigDecimal`
`money14`	`java.math.BigDecimal`
`money15`	`java.math.BigDecimal`
`string`	`java.lang.String`
`timestamp`	`java.util.Date`

Ensure that implementations of the ESPToExternalConverter interface provide a constructor with a single argument of `java.lang.String` type or a default constructor with no arguments.

Note: The stream name, the opcode and the map of column name value of the ESPMessage object are guaranteed to be valid, even if some non-key column values may be null.

Ensure that a Java archive containing an implementation of the ExternalToEFSConverter interface is provided, and place it in the `lib` subfolder of the Event Stream Processor installation folder.

If an implementation is not provided, the default implementation is used and the ESPMessage object is returned with no actual conversion performed.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Converter Class Name	Property ID: **converterClassName** Type: `string` (Required) External to Event Stream Processor message converter fully qualified class name. No default value.

Property Label	Description
Connection Factory	Property ID: **connectionFactory** Type: `string` (Required) Connection factory class name. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** ConnectionFactory • **TIBCO –** QueueConnectionFactory • **WebSphere MQ –** MyMQConnFactory No default value.
JNDI Context Factory	Property ID: **jndiContextFactory** Type: `string` (Required) Context factory for JNDI context initialization. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** org.apache.activemq.jndi.ActiveMQInitialContextFactory • **TIBCO –** com.tibco.tibjms.naming.TibjmsInitialContextFactory • **WebSphere MQ –** com.sun.jndi.fscontext.RefFSContextFactory No default value.
JNDI URL	Property ID: **jndiURL** Type: `string` (Required) JNDI URL. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** tcp://server:61616 • **TIBCO –** tibjmsnaming://server:7222 • **WebSphere MQ –** file:/var/mqm/jndi/ WebSphere MQ is different as it requires a separate naming server to be configured with it. By default, WebSphere MQ only provides a file-based naming server. No default value.

Property Label	Description
Destination Type	Property ID: **destinationType** Type: `choice` (Required) Destination type. Valid values are: • QUEUE • TOPIC Default value is QUEUE.
Destination Name	Property ID: **destinationName** Type: `string` (Required) Destination name. No default value.
Delivery Mode	Property ID: **deliveryMode** Type: `choice` (Optional) Type of delivery mode. Valid values are: • PERSISTENT • NON_PERSISTENT Default value is PERSISTENT.
Converter Parameter	Property ID: **converterParam** Type: `string` (Optional) External to ESP message converter start-up parameter. No default value.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Property Label	Description
Column To Message Property Map	Property ID: **columnPropertyMap** Type: `string` (Advanced)A comma-delimited list of `ColumnName=PropertyName` mappings that enables message filtering on the message broker side using the JMS selector mechanism. Ensure that there are no spaces in the value of this property. For each mapped column name, the outbound message is paired with a corresponding JMS property whose value equals the column value. `ColumnName1=PropertyName1,ColumnName2=PropertyName2...` No default value.
Runs Adapter in GD Mode	Property ID: **enableGDMode** Type: `boolean` (Advanced) If set to true, the adapter runs in guaranteed delivery (GD) mode and all GD-related parameters become required. Default value is false.
Name of Column Holding GD Key	Property ID: **gdKeyColumn** Type: `string` (Advanced) Specifies column name in the Flex operator holding the GD key. The GD key is a constantly increasing value that uniquely identifies every event regardless of the opcode in the stream of interest. No default value.
Name of Column Holding opcode	Property ID: **gdOpcodeColumn** Type: `string` (Advanced) Specifies name of column in Flex operator holding opcode. The opcode is the operation code (for example, inserts, update, or delete) of the event occurring in the stream of interest. No default value.
Name of Truncate Stream	Property ID: **gdControlStream** Type: `string` (Advanced) Specifies name of the control window in the GD setup. The control window is a source stream that informs the Flex operator of which data has been processed by the adapter and can be safely deleted. No default value.

Property Label	Description
Purge After Number of Records	Property ID: **gdPurgeInternal** Type: `int` (Advanced) Specifies number of records after which to purge the Flex operator. Default value is 1000.
Batch Size to Update Truncate Stream	Property ID: **gdBatchSize** Type: `int` (Advanced) Specifies number of records after which the control window must be updated with the latest GD key. Default value is 1000.

Known limitations:

- If the connection to the message broker is lost, the adapter does not attempt to reconnect.

See also

- *Chapter 4, Guaranteed Delivery* on page 1013

JMS Object Array Input Adapter

Adapter type: jms_objarray_in. The JMS Object Array Input adapter subscribes to messages formatted as arrays of Java objects from a JMS queue or topic, and writes these messages as stream records.

Note: A null element in the array generates a null value for the corresponding column.

Ensure that stream column types correspond to Java classes as follows:

Stream Column Type	Java Class
`bigdatetime`	`java.lang.Double`
`binary`	`java.lang.String`
`boolean`	`java.lang.Boolean`
`integer`	`java.lang.Integer`
`interval`	`java.lang.Long`
`date`	`java.util.Date`

Stream Column Type	Java Class
float	java.lang.Double
long	java.lang.Long
money1	java.math.BigDecimal
money2	java.math.BigDecimal
money3	java.math.BigDecimal
money4	java.math.BigDecimal
money5	java.math.BigDecimal
money6	java.math.BigDecimal
money7	java.math.BigDecimal
money8	java.math.BigDecimal
money9	java.math.BigDecimal
money10	java.math.BigDecimal
money11	java.math.BigDecimal
money12	java.math.BigDecimal
money13	java.math.BigDecimal
money14	java.math.BigDecimal
money15	java.math.BigDecimal
string	java.lang.String
timestamp	java.util.Date

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Connection Factory	Property ID: **connectionFactory** Type: `string` (Required for adapter operation and schema discovery) Connection factory class name. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ™ –** ConnectionFactory • **TIBCO –** QueueConnectionFactory • **WebSphere MQ –** MyMQConnFactory No default value.
JNDI Context Factory	Property ID: **jndiContextFactory** Type: `string` (Required for adapter operation and schema discovery) Context factory for JNDI context initialization. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** org.apache.activemq.jndi.ActiveMQInitialContextFactory • **TIBCO –** com.tibco.tibjms.naming.TibjmsInitialContextFactory • **WebSphere MQ –** com.sun.jndi.fscontext.RefFSContextFactory No default value. .

Property Label	Description
JNDI URL	Property ID: **jndiURL** Type: `string` (Required for adapter operation and schema discovery) JNDI URL. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** tcp://server:61616 • **TIBCO –** tibjmsnaming://server:7222 • **WebSphere MQ –** file:/var/mqm/jndi/ WebSphere MQ is different as it requires a separate naming server to be configured with it. By default, WebSphere MQ only provides a file-based naming server. No default value.
Destination Type	Property ID: **destinationType** Type: `choice` (Required for adapter operation and schema discovery) Destination type. Valid values are: • QUEUE • TOPIC Default value is QUEUE.
Destination Name	Property ID: **destinationName** Type: `string` (Required for adapter operation and schema discovery) Destination name. No default value.
Subscription Mode	Property ID: **subscriptionMode** Type: `choice` (Optional) Specifies the subscription mode for TOPIC. Default value is NONDURABLE. Valid values are DURABLE and NONDURABLE.

Property Label	Description
Client ID	Property ID:**clientID** Type: `string` (Required if subscription mode is set to DURABLE) Specifies the client identifier for a JMS client. Required for creating a durable subscription in JMS. Can be any string, but must be unique for each topic. No default value.
Subscription Name	Property ID: **subscriptionName** Type: `string` (Required if subscription mode is set to DURABLE) Specifies a unique name identifying a durable subscription. Required for creating a durable subscription in JMS. Can be any string, but must be unique within a given client ID. No default value.
Batch Size	Property ID: **batchsize** Type: `uint` (Optional) Specifies number of records in a batch to commit in durable subscription mode. Default value is 1.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Property Label	Description
Stream Name Opcode Expected	Property ID: **expectStreamNameOpcode** Type: `boolean` (Advanced) If true, the first two fields in CSV records are interpreted as stream name, and opcode. Default value is false .

Known limitations:

- If the connection to the message broker is lost, the adapter does not attempt to reconnect.

See also

- *Chapter 4, Guaranteed Delivery* on page 1013
- *Deprecated Adapter Support for Schema Discovery* on page 1024
- *Chapter 3, Schema Discovery* on page 1005

JMS Object Array Output Adapter

Adapter type: jms_objarray_out. The JMS Object Array Output adapter publishes stream data as an array of Java objects to a JMS queue or topic.

Note: A null value in the column generates a null element for the corresponding array.

Ensure that stream column types correspond to Java classes as follows:

Stream Column Type	Java Class
`bigdatetime`	`java.lang.Double`
`binary`	`java.lang.String`
`boolean`	`java.lang.Boolean`
`integer`	`java.lang.Integer`
`interval`	`java.lang.Long`
`date`	`java.util.Date`
`float`	`java.lang.Double`
`long`	`java.lang.Long`

Stream Column Type	Java Class
money1	java.math.BigDecimal
money2	java.math.BigDecimal
money3	java.math.BigDecimal
money4	java.math.BigDecimal
money5	java.math.BigDecimal
money6	java.math.BigDecimal
money7	java.math.BigDecimal
money8	java.math.BigDecimal
money9	java.math.BigDecimal
money10	java.math.BigDecimal
money11	java.math.BigDecimal
money12	java.math.BigDecimal
money13	java.math.BigDecimal
money14	java.math.BigDecimal
money15	java.math.BigDecimal
string	java.lang.String
timestamp	java.util.Date

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Connection Factory	Property ID: **connectionFactory** Type: `string` (Required) Connection factory class name. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** ConnectionFactory • **TIBCO –** QueueConnectionFactory • **WebSphere MQ –** MyMQConnFactory No default value.
JNDI Context Factory	Property ID: **jndiContextFactory** Type: `string` (Required) Context factory for JNDI context initialization. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** org.apache.activemq.jndi.ActiveMQInitialContextFactory • **TIBCO –** com.tibco.tibjms.naming.TibjmsInitialContextFactory • **WebSphere MQ –** com.sun.jndi.fscontext.RefFSContextFactory No default value.
JNDI URL	Property ID: **jndiURL** Type: `string` (Required) JNDI URL. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** tcp://server:61616 • **TIBCO –** tibjmsnaming://server:7222 • **WebSphere MQ –** file:/var/mqm/jndi/ WebSphere MQ is different as it requires a separate naming server to be configured with it. By default, WebSphere MQ only provides a file-based naming server. No default value.

Property Label	Description
Destination Type	Property ID: **destinationType** Type: `choice` (Required) Destination type. Valid values are: • QUEUE • TOPIC Default value is QUEUE.
Destination Name	Property ID: **destinationName** Type: `string` (Required) Destination name.
Delivery Mode	Property ID: **deliveryMode** Type: `choice` (Optional) Type of delivery mode. Valid values are: • PERSISTENT • NON_PERSISTENT Default value is PERSISTENT.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Property Label	Description
Column To Message Property Map	Property ID: **columnPropertyMap** Type: `string` (Advanced) A comma-delimited list of `ColumnName=PropertyName` mappings that enables message filtering on the message broker side using the JMS selector mechanism. Ensure that no spaces are present in the value of this property. For each mapped column name, the outbound message is paired with a corresponding JMS property for which the value equals the column value. `ColumnName1=PropertyName1,ColumnName2=PropertyName2...` No default value.
Prepend Stream Name, Opcode	Property ID: **prependStreamNameOpcode** Type: `boolean` (Advanced) If true, every CSV record is prepended with stream name, opcode. No default value.
Runs Adapter in GD Mode	Property ID: **enableGDMode** Type: `boolean` (Advanced) If set to true, the adapter runs in guaranteed delivery (GD) mode and all GD-related parameters become required. Default value is false.
Name of Column Holding GD Key	Property ID: **gdKeyColumn** Type: `string` (Advanced) Specifies column name in the Flex operator holding the GD key. The GD key is a constantly increasing value that uniquely identifies every event regardless of the opcode in the stream of interest. No default value.

Property Label	Description
Name of Column Holding opcode	Property ID: **gdOpcodeColumn** Type: `string` (Advanced) Specifies name of column in Flex operator holding opcode. The opcode is the operation code (for example, inserts, update, or delete) of the event occurring in the stream of interest. No default value.
Name of Truncate Stream	Property ID: **gdControlStream** Type: `string` (Advanced) Specifies name of the control window in the GD setup. The control window is a source stream that informs the Flex operator of which data has been processed by the adapter and can be safely deleted. No default value.
Purge After Number of Records	Property ID: **gdPurgeInternal** Type: `int` (Advanced) Specifies number of records after which to purge the Flex operator. Default value is 1000.
Batch Size to Update Truncate Stream	Property ID: **gdBatchSize** Type: `int` (Advanced) Specifies number of records after which the control window must be updated with the latest GD key. Default value is 1000.

Known limitations:

- If the connection to the message broker is lost, the adapter does not attempt to reconnect.

See also

- *Chapter 4, Guaranteed Delivery* on page 1013

JMS XML Input Adapter

Adapter type: jms_xml_in. The JMS XML Input adapter subscribes to XML-formatted text messages from a JMS queue or topic, and writes the messages as stream records.

Ensure that each message consists of an XML element. If opted, the element name corresponds to the stream name.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

The ESP-OPS attribute is optional. If omitted, the message is interpreted as an upsert. Ensure that the rest of the attributes have the same names as the corresponding stream columns, and that the columns with null values are omitted. This adapter supports schema discovery.

Property Label	Description
Connection Factory	Property ID: **connectionFactory** Type: `string` (Required for adapter operation and schema discovery) Connection factory class name. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ™ –** ConnectionFactory • **TIBCO –** QueueConnectionFactory • **WebSphere MQ –** MyMQConnFactory No default value.
JNDI Context Factory	Property ID: **jndiContextFactory** Type: `string` (Required for adapter operation and schema discovery) Context factory for JNDI context initialization. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** org.apache.activemq.jndi.ActiveMQInitialContextFactory • **TIBCO –** com.tibco.tibjms.naming.TibjmsInitialContextFactory • **WebSphere MQ –** com.sun.jndi.fscontext.RefFSContextFactory No default value.

Property Label	Description
JNDI URL	Property ID: **jndiURL** Type: `string` (Required for adapter operation and schema discovery) JNDI URL. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** tcp://server:61616 • **TIBCO –** tibjmsnaming://server:7222 • **WebSphere MQ –** file:/var/mqm/jndi/ WebSphere MQ is different as it requires a separate naming server to be configured with it. By default, WebSphere MQ only provides a file-based naming server. No default value.
Destination Type	Property ID: **destinationType** Type: `choice` (Required for adapter operation and schema discovery) Destination type. Valid values are: • QUEUE • TOPIC Default value is QUEUE.
Destination Name	Property ID: **destinationName** Type: `string` (Required for adapter operation and schema discovery) Destination name. No default value.
Subscription Mode	Property ID: **subscriptionMode** Type: `choice` (Optional) Specifies the subscription mode for TOPIC. Default value is NONDURABLE. Valid values are DURABLE and NONDURABLE.

Property Label	Description
Client ID	Property ID: **clientID** Type: `string` (Required if subscription mode is set to DURABLE) Specifies the client identifier for a JMS client. Required for creating a durable subscription in JMS. Can be any string, but must be unique for each topic. No default value.
Subscription Name	Property ID: **subscriptionName** Type: `string` (Required if subscription mode is set to DURABLE) Specifies a unique name identifying a durable subscription. Required for creating a durable subscription in JMS. Can be any string, but must be unique within a given client ID. No default value.
Batch Size	Property ID: **batchsize** Type:`uint` (Optional) Specifies number of records in a batch to commit in durable subscription mode. Default value is 1.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.
Match Stream Name	Property ID: **matchStreamName** Type: `boolean` (Advanced) Ignore message if the XML element name does not match the source stream name. Default value is false.

Property Label	Description
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Date format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Timestamp format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.

Known limitations:

- If the connection to the message broker is lost, the adapter does not attempt to reconnect.

See also

- *Chapter 4, Guaranteed Delivery* on page 1013
- *Deprecated Adapter Support for Schema Discovery* on page 1024
- *Chapter 3, Schema Discovery* on page 1005

JMS XML Output Adapter

Adapter type: jms_xml_out. The JMS XML Output adapter publishes stream data as XML-formatted text messages to a JMS queue or topic.

Ensure that each message consists of an XML element with the same name as the stream name.

The first attribute is the Event Stream Processor opcode. The rest of the attributes have the same names as the corresponding stream columns. Ensure that any columns with null values are omitted.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Connection Factory	Property ID: **connectionFactory** Type: `string` (Required) Connection factory class name. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** ConnectionFactory • **TIBCO –** QueueConnectionFactory • **WebSphere MQ –** MyMQConnFactory No default value.
JNDI Context Factory	Property ID: **jndiContextFactory** Type: `string` (Required) Context factory for JNDI context initialization. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** org.apache.activemq.jndi.ActiveMQInitialContextFactory • **TIBCO –** com.tibco.tibjms.naming.TibjmsInitialContextFactory • **WebSphere MQ –** com.sun.jndi.fscontext.RefFSContextFactory No default value.
JNDI URL	Property ID: **jndiURL** Type: `string` (Required) JNDI URL. Consult your third-party vendor documentation for specific formats. Here are some examples: • **ActiveMQ –** tcp://server:61616 • **TIBCO –** tibjmsnaming://server:7222 • **WebSphere MQ –** file:/var/mqm/jndi/ WebSphere MQ is different as it requires a separate naming server to be configured with it. By default, WebSphere MQ only provides a file-based naming server. No default value.

Property Label	Description
Destination Type	Property ID: **destinationType** Type: `choice` (Required) Destination type. Valid values are: • QUEUE • TOPIC Default value is QUEUE.
Destination Name	Property ID: **destinationName** Type: `string` (Required) Destination name. No default value.
Delivery Mode	Property ID: **deliveryMode** Type: `choice` (Optional) Type of delivery mode. Valid values are: • PERSISTENT • NON_PERSISTENT Default value is PERSISTENT.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Property Label	Description
Column To Message Property Map	Property ID: **columnPropertyMap** Type: `string` (Advanced)A comma-delimited list of `ColumnName=PropertyName` mappings that enables message filtering on the message broker side using the JMS selector mechanism. Ensure that there are no spaces present in the value of this property. For each mapped column name, the outbound message is paired with a corresponding JMS property that has a value equal to the column value. `ColumnName1=PropertyName1,ColumnName2=PropertyName2...` No default value.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Date format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Timestamp format. Default value is YYYY-MM-DDTHH:MM:SS.SSS.
Runs Adapter in GD Mode	Property ID: **enableGDMode** Type: `boolean` (Advanced) If set to true, the adapter runs in guaranteed delivery (GD) mode and all GD-related parameters become required. Default value is false.
Name of Column Holding GD Key	Property ID: **gdKeyColumn** Type: `string` (Advanced) Specifies column name in the Flex operator holding the GD key. The GD key is a constantly increasing value that uniquely identifies every event regardless of the opcode in the stream of interest. No default value.

Property Label	Description
Name of Column Holding op-code	Property ID: **gdOpcodeColumn** Type: `string` (Advanced) Specifies name of column in Flex operator holding opcode. The opcode is the operation code (for example, inserts, update, or delete) of the event occurring in the stream of interest. No default value.
Name of Truncate Stream	Property ID: **gdControlStream** Type: `string` (Advanced) Specifies name of the control window in the GD setup. The control window is a source stream that informs the Flex operator of which data has been processed by the adapter and can be safely deleted. No default value.
Purge After Number of Records	Property ID: **gdPurgeInternal** Type: `int` (Advanced) Specifies number of records after which to purge the Flex operator. Default value is 1000.
Batch Size to Update Truncate Stream	Property ID: **gdBatchSize** Type: `int` (Advanced) Specifies number of records after which the control window must be updated with the latest GD key. Default value is 1000.

Known limitations:

- If the connection to the message broker is lost, the adapter does not attempt to reconnect.

See also

- *Chapter 4, Guaranteed Delivery* on page 1013

Socket (As Client) CSV Input Adapter

Adapter type: dsv_sockout_in. The Socket (as Client) CSV Input adapter receives data in delimited format from outgoing network adapters.

The adapter initiates the connection to an external datasource, and an external program sends out the data. The data does not require a header (accepted by **esp_convert**). If the file includes a header, the header specifies the field names.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Sample record formats for the data file:

```
1. hasHeader=true
delimiter=,
expectStreamNameOpcode=false

Ts,ItemID,Price,Quantity,WarehouseZipCode,DeliveryZipCode
2004/06/17 10:00:00.000000,SKU1276532,50.00,1,10012,94086
2004/06/17 10:00:05.000000,SKU6723143,23.00,2,10012,94043

2. expectStreamNameOpcode=true
delimiter=,

Trades_in,i,2004/06/17
10:00:00.000000,SKU1276532,50.00,1,10012,94086
Trades_in,i,2004/06/17
10:00:05.000000,SKU6723143,23.00,2,10012,94043

3. expectStreamNameOpcode=false
timestampFormat=%Y/%m/%d %H:%M:%S
delimiter=,

2004/06/17 10:00:00.000000,SKU1276532,50.00,1,10012,94086
2004/06/17 10:00:05.000000,SKU6723143,23.00,2,10012,94043
```

Property Label	Description
Server	Property ID: **host** Type: `string` (Required) Server host name. Default value is localhost.
Port	Property ID: **port** Type: `int` (Required) Server port. If **port** is set to -1, the adapter reads from the Ephemeral Port File. Default value is 12345.
Stream name, opcode expected	Property ID: **expectStreamNameOpcode** Type: `boolean` (Optional) If true, the adapter interprets the first two fields as a stream name and opcode respectively. Adapters discard messages with unmatched stream names. Default value is false.

Property Label	Description
Field Count	Property ID: **fieldCount** Type: `uint` (Optional) Counts the number of fields in a CSV file, if different from the source stream. Default value is 0.
Delimiter	Property ID: **delimiter** Type: `string` (Advanced) Symbol used to separate the columns. Default value is a comma (,).
Has Header	Property ID: **hasHeader** Type: `boolean` (Advanced) Determines whether the first line of the file contains the description of the fields. Default value is false .
Ephemeral Port File	Property ID: **epFile** Type: `filename` (Advanced) File that contains the server port number, if **Port** is -1. No default value.
Retry Period	Property ID: **retryperiod** Type: `uint` (Advanced) Period for trying to re-establish an outgoing connection. In seconds. Default value is 1.
Enter Initial State	Property ID: **initial** Type: `choice` (Advanced) When the adapter enters the initial loading state. Default value is never.
Convert to Safe Opcodes	Property ID: **safeOps** Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT. Converts DELETE to SAFEDELETE. Default value is false.

Property Label	Description
Skip Deletes	Property ID: **skipDels** Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. Default value is false.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Format string for parsing timestamp values. Default value is %Y-%m-%dT%H:%M:%S.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Format string for parsing date values. Default value is %Y-%m-%dT%H:%M:%S.
Block Size	Property ID: **blockSize** Type: `int` (Advanced) Determines number of records to block into one pseudo-transaction. Default value is 1.
Field Mapping	Property ID: **permutation** Type: `permutation` Mapping between Event Stream Processor and external fields, for example:
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Known limitations:

- The adapter ignores the stream name in the file rows.
- All data is sent to the same stream.

Socket (As Client) CSV Output Adapter

Adapter type: dsv_sockout_out. The Socket (as Client) CSV Output adapter sends data in delimited format to the outgoing network.

The Socket (as Client) CSV Output adapter initiates the connection to an external datasource and sends out the data. The data does not require a header (accepted by **esp_convert**). If the file includes a header, it specifies the field names. The adapter retries a connection if the connection breaks.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Server	Property ID: **host** Type: `string` (Required) Server host name. Default value is localhost.
Port	Property ID: **port** Type: `int` (Required) Server port. If **port** is set to -1, the adapter reads from the Ephemeral Port File. Default value is 12345.
Prepend stream name, opcode	Property ID: **prependStreamNameOpcode** Type: `boolean` (Optional) If true, the first two fields are interpreted as stream name and opcode respectively. The adapter discards messages with unmatched stream names. Default value is false.
Delimiter	Property ID: **delimiter** Type: `string` (Advanced) Symbol used to separate the columns. Default value is a comma (,) .

Property Label	Description
Has Header	Property ID: **hasHeader** Type: `boolean` (Advanced) Determines whether the first line of the file contains the description of the fields. Default value is false.
Ephemeral Port File	Property ID: **epFile** Type: `filename` (Advanced) File that contains the server port number, if **port** is -1. No default value.
Retry Period, s	Property ID: **retryperiod** Type: `uint` (Advanced) Period for trying to re-establish an outgoing connection, in seconds. Default value is 1.
Include Base Content	Property ID: **outputBase** Type: `boolean` (Optional) Starts by recording the initial contents of the stream, not just the updates. Default value is false.
Only Base Content	Property ID: **onlyBase** Type: `boolean` (Advanced) Sends the initial contents of the stream once. Default value is false .
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Format string for parsing timestamp values. Default value is %Y-%m-%dT%H:%M:%S.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Format string for parsing date values. Default value is %Y-%m-%dT%H:%M:%S.

Property Label	Description
Field Mapping	Property ID: **permutation** Type: `permutation` Mapping between Event Stream Processor and external fields, for example:
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Socket (As Client) XML Input Adapter

Adapter type: xml_sockout_in. The Socket (As Client) XML Input adapter receives data in Event Stream Processor format from the outgoing network adapters.

The adapter initiates a connection with an outgoing network adapter, which can then send data to the input adapter. It is possible for the data not to have the header, or for the header not to specify the field names.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Sample record format for the data file:

```
<Trades Id="0" Symbol="EBAY" TradeTime="2000-05-04T12:00:00"
Price="140.0" Shares="50" />
<Trades Id="1" Symbol="EBAY" TradeTime="2000-05-04T12:00:01"
Price="150.0" Shares="500" />
```

Property Label	Description
Server	Property ID: **host** Type: `string` (Required) The server host name. Default value is localhost.

Property Label	Description
Port	Property ID: **port** Type: `int` (Required) Server port. If **port** is set to -1, the adapter reads from the Ephemeral Port File. Default value is 12345.
Match Stream Name	Property ID: **matchStreamName** Type: `boolean` (Optional) Ignores messages if the XML element name does not match the source stream name. Default value is false.
Ephemeral Port File	Property ID: **epFile** Type: `filename` (Advanced) The file that contains the server port number, if **port** is set to -1. Default value is false.
Retry period (seconds)	Property ID: **retryperiod** Type: `uint` (Advanced) Indicates the time period for attempting to re-establish an outgoing connection, in seconds. Default value is 1.
Enter Initial State	Property ID: **initial** Type: `choice` (Advanced) Indicates when the adapter enters the initial loading state. Default value is never.
Convert to Safe Opcodes	Property ID: **safeOps** Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT, and DELETE to SAFEDELETE. Default value is false.
Skip Deletes	Property ID: **skipDels** Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. Default value is false.

Property Label	Description
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) Format string for parsing date values. Default value is %Y-%m-%dT%H:%M:%S.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) Format string for parsing timestamp values. Default value is %Y-%m-%dT%H:%M:%S.
Block Size	Property ID: **blockSize** Type: `int` (Advanced) Number of records to block into one pseudo-transaction. Default value is 1.
Field Mapping	Property ID: **permutation** Type: `permutation` Mapping between Event Stream Processor and external fields. Format is the ESP column name equals the database column name to which you are mapping. Multiple mappings are separated by a colon. For example, <esp_columnname>=<database_columnname>:<esp_columnname>=<database_columnname>. No default value.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Known limitations:

- The adapter ignores the stream name in the file rows.
- All the data is sent to the same stream.

Socket (As Client) XML Output Adapter

Adapter type: xml_sockout_out. The Socket (As Client) XML Output adapter sends data in Event Stream Processor format to the outgoing network adapter.

The adapter initiates a connection with another program and then sends the data. If the connection is broken, the adapter retries the connection.

You can configure this adapter to send only the base state of the stream. The adapter sends data once and exits, but can be restarted later.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Server	Property ID: **host** Type: `string` (Required) The server host name. Default value is localhost.
Port	Property ID: **port** Type: `int` (Required) Server port. If **port** is set to -1, the adapter reads from the Ephemeral Port File. Default value is 12345.
Include Base Content	Property ID: **outputBase** Type: `boolean` (Optional) Records the initial contents of the stream and not just the updates. Default value is false.
Ephemeral Port File	Property ID: **epFile** Type: `filename` (Advanced) The file that contains the server port number, if **port** is -1. No default value.
Retry period, s	Property ID: **retryperiod** Type: `uint` (Advanced) The time period for attempting to re-establish an outgoing connection, in seconds. Default value is 1.

Property Label	Description
Only Base Content	Property ID: **onlyBase** Type: `boolean` (Advanced) Sends only the initial contents of the stream. Default value is false.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) The format string for parsing date values. Default value is %Y-%m-%dT%H:%M:%S.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) The format string for parsing timestamp values. Default value is %Y-%m-%dT%H:%M:%S.
Field Mapping	Property ID: **permutation** Type: `permutation` (Advanced) Maps the internal ESP fields to the application or diplay fields. Default value is %Y-%m-%dT%H:%M:%S.
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Socket (As Server) CSV Input Adapter

Adapter type: dsv_sockin_in. The Socket (As Server) CSV Input adapter receives data in Event Stream Processor delimited format from the incoming network adapters.

Another program initiates the connection and then sends the data to the adapter.

It is possible for the data not to have the header, or for the header not to specify the field names.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Port	Property ID: **port** Type: `int` (Required) Server port. If **port** is set to -1, the adapter reads from the Ephemeral Port File. Default value is 12345.
Stream name, opcode expected	Property ID: **expectStreamNameOpcode** Type: `boolean` (Optional) If true, the first two fields are interpreted as stream name and opcode respectively. Messages with unmatched stream names are discarded. Default value is false.
Delimiter	Property ID: **delimiter** Type: `string` (Advanced) Symbol used to separate the columns. Default value is a comma (,).
Has Header	Property ID: **hasHeader** Type: `boolean` (Advanced) Determines whether the first line of the file contains the description of the fields. Default value is false.
Ephemeral Port File	Property ID: **epFile** Type: `filename` (Advanced) The file that contains the server port number, if **port** is -1.
Initial Listen Period (seconds)	Property ID: **retryperiod** Type: `uint` (Advanced) How long to wait for the first incoming connection before switching to the continuous state. Default value is 0.

Property Label	Description
Enter Initial State	Property ID: **initial** Type: `choice` (Advanced) Indicates when the adapter enters the initial loading state. Default value is never.
Convert to Safe Opcodes	Property ID: **safeOps** Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT, and DELETE to SAFEDELETE. Default value is False.
Skip Deletes	Property ID: **skipDels** Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. Default value is false.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) The format string for parsing date values. Default value is %Y-%m-%dT%H:%M:%S.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) The format string for parsing timestamp values. Default value is %Y-%m-%dT%H:%M:%S .
Block Size	Property ID: **blockSize** Type: `int` (Advanced) The number of records to block into one pseudo-transaction. Default value is 1.

Property Label	Description
Use Envelopes	Property ID: **useEnvelopes** Type: `boolean` (Advanced) Specify the block type the adapter uses to pass data to the engine. If you specify a **blockSize** property greater than zero, by default, the adapter packages rows into transaction blocks to send to the engine. To get the adapter to package rows into envelope blocks instead, set this property to true. Default value is false.
Field Mapping	Property ID: **permutation** Type: `permutation` Mapping between Event Stream Processor and external fields, for example:
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Known limitations:

- The stream name in the file rows is ignored.
- All the data is sent to the same stream.
- Supports only one network connection.

Socket (As Server) CSV Output Adapter

Adapter type: dsv_sockin_out. The Socket (As Server) CSV Output adapter sends data in Event Stream Processor delimited format to the incoming network adapters.

The adapter can be configured to send only the base state of the stream. The socket closes after sending the base state of the stream but may be repeatedly reconnected.

It is possible for the data not to have the header, or for the header not to specify the field names.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Port	Property ID: **port** Type: `int` (Required) Server port. If **port** is set to -1, the adapter reads from the Ephemeral Port File. Default value is 12345.
Include Base Content	Property ID: **outputBase** Type: `boolean` (Optional) Records the initial contents of the stream, not just the updates. Default value is false.
Prepend Stream Name, Opcode	Property ID: **prependStreamNameOpcode** Type: `boolean` (Optional) If true, each message starts with the stream name and the opcode. Default value is false.
Ephemeral Port File	Property ID: **epFile** Type: `filename` (Advanced) The file that contains the server port number, if **port** is -1. No default value.
Only Base Content	Property ID: **onlyBase** Type: `boolean` (Advanced) The adapter sends only the initial contents of the stream, once. Default value is false.
Delimiter	Property ID: **delimiter** Type: `string` (Advanced) The symbol used to separate the columns. Default value is a comma (,).
Has Header	Property ID: **hasHeader** Type: `boolean` (Advanced) Whether the first line of the file contains the description of the fields. Default value is false .

Property Label	Description
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) The format string for parsing date values. Default value is %Y-%m-%dT%H:%M:%S.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) The format string for parsing timestamp values. Default value is %Y-%m-%dT%H:%M:%S .
Field Mapping	Property ID: **permutation** Type: `permutation` Mapping between Event Stream Processor and external fields, for example:
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Known limitations:

- Supports only one network connection.

Socket (As Server) XML Input Adapter

Adapter type: xml_sockin_in. The Socket (As Server) XML Input adapter receives data in Event Stream Processor format from the incoming network adapter.

Another program initiates the connection and then sends the data.

This adapter can be configured to send only the base state of the stream, and can be repeatedly reconnected.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Port	Property ID: **port** Type: `int` (Required) Server port. If **port** is set to -1, the adapter reads from the Ephemeral Port File. Default value is 12345.
Match Stream Name	Property ID: **matchStreamName** Type: `boolean` (Optional) If true, the XML element names are matched against the stream name. Unmatched messages are discarded. Default value is false.
Ephemeral Port File	Property ID: **epFile** Type: `filename` (Advanced) The file that contains the **port** number, if port is -1. No default value .
Initial Listen Period (seconds)	Property ID: **retryperiod** Type: `uint` (Advanced) Designates the length of time to wait for the first incoming connection before switching to the continuous state. Default value is 0.
Enter Initial State	Property ID: **initial** Type: `choice` (Advanced) Indicates when the adapter enters the initial loading state. Default value is never.
Convert to Safe Opcodes	Property ID: **safeOps** Type: `boolean` (Advanced) Converts the opcodes INSERT and UPDATE to UPSERT, and DELETE to SAFEDELETE. Default value is false.

Property Label	Description
Skip Deletes	Property ID: **skipDels** Type: `boolean` (Advanced) Skips the rows with opcodes DELETE or SAFEDELETE. Default value is false.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) The format string for parsing date values. Default value is %Y-%m-%dT%H:%M:%S.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) The format string for parsing timestamp values. Default value is %Y-%m-%dT%H:%M:%S .
Block Size	Property ID: **blockSize** Type: `int` (Advanced) Number of records to block into one pseudo-transaction. Default value is 1.
Use Envelopes	Property ID: **useEnvelopes** Type: `boolean` (Advanced) Specify the block type the adapter uses to pass data to the engine. If you specify a **blockSize** property greater than zero, by default, the adapter packages rows into transaction blocks to send to the engine. To get the adapter to package rows into envelope blocks instead, set this property to true. Default value is false.
Field Mapping	Property ID: **permutation** Type: `permutation` Mapping between Event Stream Processor and external fields, for example:

Property Label	Description
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Known limitations:

- The adapter ignores the stream name in the file entries.
- All the data is sent to the same stream.
- Supports only one network connection at a time.

Socket (As Server) XML Output Adapter

Adapter type: xml_sockin_out. The Socket (As Server) XML Output adapter receives data in Event Stream Processor format from the outgoing network adapters.

Another program initiates the connection and then receives the data from the output adapter.

This adapter can be configured to send only the base state of the stream. The socket closes after sending the base state of the stream but can be repeatedly reconnected.

If you use the CCL **ATTACH ADAPTER** statement to attach an adapter, you must supply the adapter type.

Property Label	Description
Port	Property ID: **port** Type: `int` (Required) Server port. If **port** is set to -1, the adapter reads from the Ephemeral Port File. Default value is 12345 .

Property Label	Description
Include Base Content	Property ID: **outputBase** Type: `boolean` (Optional) Starts by recording the initial contents of the stream, not just the updates. Default value is false.
Ephemeral Port File	Property ID: **epFile** Type: `filename` (Advanced) The file that contains the **port** number, if port is -1. No default value.
Only Base Content	Property ID: **onlyBase** Type: `boolean` (Advanced) The adapter sends the initial contents of the stream, once. Default value is false.
Date Format	Property ID: **dateFormat** Type: `string` (Advanced) The format string for parsing date values. Default value is %Y-%m-%dT%H:%M:%S.
Timestamp Format	Property ID: **timestampFormat** Type: `string` (Advanced) The format string for parsing timestamp values. Default value is %Y-%m-%dT%H:%M:%S .
Field Mapping	Property ID: **permutation** Type: `permutation` Mapping between Event Stream Processor and external fields. Format is the ESP column name equals the database column name to which you are mapping. Multiple mappings are separated by a colon. For example, <esp_columnname>=<database_columnname>:<esp_columnname>=<database_columnname>. No default value.

Property Label	Description
PropertySet	Property ID: **propertyset** Type: `string` (Advanced) Specifies the name of the property set. Property sets are reusable sets of properties that are stored in the project configuration file. Using these sets allows you to move adapter configuration properties out of the CCL file and into the CCR file. If you specify the same properties in the project configuration file and the **ATTACH ADAPTER** statement, the values in the property set override the values defined in the **ATTACH ADAPTER** statement. No default value.

Known limitations:

- Supports only one network connection at a time.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X