SYBASE®
An SAP® Company

FIX Adapter Guide

Sybase CEP Option R4

DOCUMENT ID: DC01258-01-0400-04
LAST REVISED: August 2011

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

Introduction

The Financial Information Exchange (FIX) adapter enables you to send and receive FIX messages from the Sybase® CEP server.

The FIX adapter is an out of process adapter, and can function as both an input and output adapter.

The adapter works with two plug-ins to process the FIX messages.

- The Data Mapping Plug-in
- The Connection Plug-in

The FIX adapter works with these two plug-ins and the Sybase CEP Server to publish to, and subscribe from, streams on the Sybase CEP Server.

Default implementations of the Connection and Data Mapping plug-ins have been provided. These allow for the direct exchange of FIX messages between a counterparty FIX server. They also enable the reading and writing of FIX messages to a file or database table.

Custom implementations of the plug-ins can be written to extend the connectivity of the FIX adapter to other FIX message sources.

Supported Platforms

The Sybase CEP FIX Adapter is available for all platforms and operating systems supported by CEP R4 server.

Supported platforms and Operating systems:

Platform	**Supported OS**
Linux-64 (AMD/Intel)	Red Hat 5.0 (AMD/Intel), SUSE 10 (AMD)
Sun-64 (Sparc)	Sun Solaris 10
Sun-64 (UltraSparc T2)	Sun Solaris 10
Sun-64 (AMD)	Sun Solaris 10
Window (32,64)	Windows 2003 Server (64-bit), XP Professional (32-bit and 64-bit)

The FIX adapter can be configured to run as a service on Windows platforms. For Unix users using one of the FIX adapter's default connection plug-ins, unixODBC must be installed.

Supported FIX Protocol Versions

The FIX adapter supports FIX protocol versions 4.0 to 4.4.

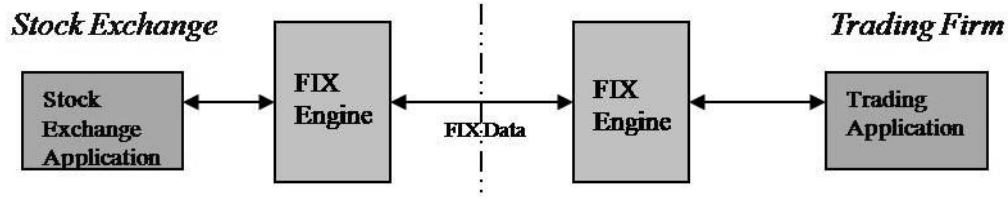
By default, the Mapping Plug-in shipped with the FIX adapter does not support mapping nested fields from FIX to CEP, or from CEP to FIX. However, if you use the **C8_FIXMsgBody** column to send raw FIX messages, the nested fields will be preserved. Although the nested fields are preserved, you cannot modify the values of the nested fields using the FIX Adapter.

Overview

The Financial Information eXchange (FIX) protocol was developed to facilitate real-time financial transactions.

FIX is primarily used by banks, brokerage firms, information technology providers, exchanges, and institutional investors, to electronically communicate trade-related messages. In its simplest form, an application generates FIX messages which are then transmitted using a FIX Engine. A second FIX engine then receives the messages, and re-transmits them to its respective application.

Example

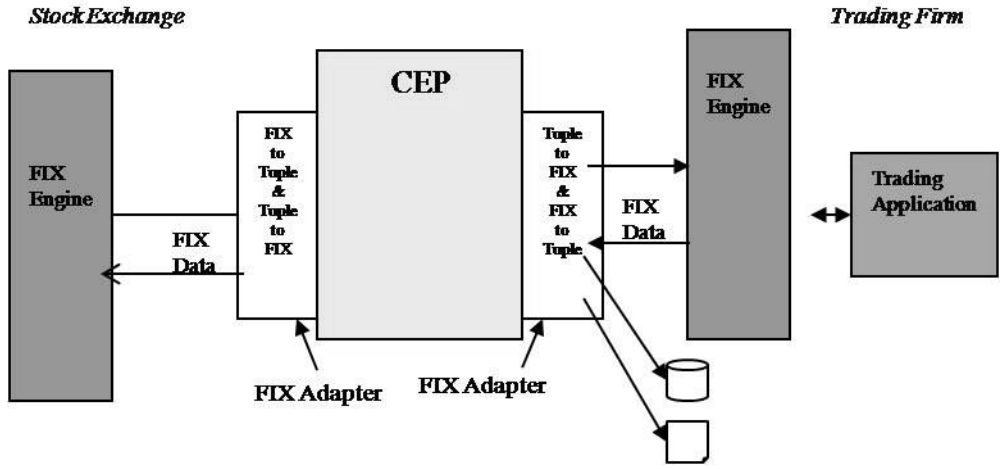


The stock exchange application generates messages and transmits them using the stock exchange's FIX engine. In this example, the stock exchange's FIX engine is the "initiator".

The trading firm's FIX engine then receives the messages, and one or more of the firm's applications accepts the messages in FIX format. The trading firm's FIX engine is called the "acceptor".

The FIX Adapter and CEP

The FIX Adapter can publish FIX messages to Sybase CEP.



The FIX Adapter on the left acts as an acceptor, receiving FIX messages from the FIX engine and converting them to tuples. The tuples are published to Sybase CEP before being re-converted to FIX messages and sent to the accepting FIX engine on the right.

The FIX Adapter can also be configured to write FIX messages to a database or file.

Installation and Configuration

Installation

The FIX Adapter is an out-of-process adapter. It can be installed in either standalone or dependent mode.

As with all Sybase CEP out-of-process adapters, the FIX Adapter does not need to be on the same platform as the Sybase CEP Server it is communicating with.

The FIX Adapter installer uses the CEP Adapter Installer framework, a graphical installer that includes SySAM install and supports silent installation. The FIX Adapter is available for all platforms supported by Sybase CEP Server.

Version information for upgrades or reinstallations can be retrieved from the command line utilities, such as **c8_server -version**.

Standalone Mode

A standalone mode installation will install all of the files required by the FIX Adapter, including the CEP common libraries. This mode should be used on a system that does not already have Sybase CEP Server or Sybase CEP Studio installed.

If you are installing the adapter and specify a directory that contains a previous installation, a pop-up message indicates that the adapter is already installed. You will then have the option to either uninstall the previous installation, or to upgrade without uninstalling.

Dependent Mode

A dependent mode installation can be used if the system already has Sybase CEP Server or Studio installed. The installer will only install files that are specific to the FIX adapter. By default, the FIX Adapter will use the CEP common libraries already present from the Sybase CEP Server and Sybase CEP Studio installation.

If you are upgrading or reinstalling the adapter and specify a directory that contains a previous installation, a pop-up message indicates that the adapter is already installed. You will then have the option to either uninstall the previous installation, or to upgrade without uninstalling.

Note: Dependent mode installation should only be used if the Sybase CEP Server and Sybase CEP Studio are of the same version as the FIX Adapter. If the versions are not the same, a standalone mode installation should be performed. Note that for dependent mode installations, the Sybase CEP Server should be at the same ESD level or newer.

Post-Installation Tasks

If you performed a licensed installation, it is recommended that you copy the license file from the FIX adapter's directory to the default location for all other Sybase CEP licenses, typically:

```
cp /software/SybaseC8/server/enterprise-adapters/fix/
SYSAM-2_0/licenses/SYBASE.lic $SYBASE_C8/server/SYSAM-2_0/
licenses/SYBASE_FIX.lic
```

Note: Do not overwrite any existing licenses. Copy the license to a new name, such as "SYBASE_FIX.lic".

Prior to running the FIX Adapter, you must define the **LD_LIBRARY_PATH** such that the FIX libraries and the Sybase CEP common libraries can be found:

```
setenv LD_LIBRARY_PATH $SYBASE_C8/server/bin:$SYBASE_C8/server/
enterprise-adapters/fix/bin
```

Known Issues

644323	FIX Adapter installation silently modifies the user's login scripts (.cshrc, .kshrc, etc.). It silently appends it's installation directory to the LD_LIBRARY_PATH.
644330	Files are left in user's home directory after uninstalling the FIX adapter. After each installation, the FIX adapter creates a file similar to `*.cshrcxxxxxx`, where "xxxxxx" is a random number.
644362	FIX Adapter's installer copies the license file to `$SYBASE_C8/server/enterprise-adapters/SYSAM-2_0/licenses` but all of the sample configuration files refer to `$SYBASE_C8/server/SYSAM-2_0/licenses`. All other licenses normally go into `$SYBASE_C8/server/SYSAM-2_0/licenses`. Additionally, the installer copies the license file to two different directories: `/software/SybaseC8/server/enterprise-adapters/fix/SYSAM-2_0/licenses/SYBASE.lic` and `/software/SybaseC8/server/enterprise-adapters/SYSAM-2_0/licenses/SYBASE.lic` An administrator who is unaware of this risks their adapter going into grace mode and shutting down while in production.

Configuration

The default name for the FIX Adapter configuration file is `fix_adapter_prefs.xml`, located in the same directory as the `c8_fix_adapter` executable.

When starting the FIX adapter, always use the `--config` option to explicitly specify the configuration file name and location (for example, `[c:\sybasec8\enterprise-adapters\fix\bin]c8_fix_adapter.exe --config=my_fix_adapter_prefs.xml`).

The FIX Adapter configuration file contains three main sections:

- FIX Adapter-specific settings (<section name="C8FixAdapter">).
- Logging-related settings (<section name="c8/Logging/LogWriters">).
- Other settings (section name="C8/General">).

FIX Adapter-Specific Settings

Settings controlling retry behavior when initializing publishers/subscribers on adapter startup:

```
<!-- Retry settings to use when initializing publishers/subscribers
-->
    <!-- ConnectionRetryInterval set to 0 or missing means disable
retries -->
   <!-- When ConnectionRetryInterval is set to a non-zero value, the
adapter -->
    <!-- will retry the number of times specified by the
ConnectionTotalRetries -->
    <!-- setting, or keep retrying until connection succeeds or the
adapter is -->
   <!-- shutdown if ConnectionTotalRetries is set to 0 or is missing
-->
    <preference name="ConnectionRetryInterval">1</preference>
    <preference name="ConnectionTotalRetries">3</preference>
```

<ConnectionTotalRetries>	Indicates the number of times to retry the initialization of publishers and subscribers. If set to 0 or missing, the adapter retries until the publishers or subscribers initialize, or the adapter is shut down.
<ConnectionRetryInterval>	When this is set to a non-zero value, the adapter retries the number of times specified by the <ConnectionTotalRetries> setting (or continuously if <ConnectionTotalRetries> is 0 or missing) using the specified value as the retry interval, in seconds. If set to 0, retrying is disabled.

Settings related to the Connection Plug-in:

```
<section name="ConnectionPlugin">
        <preference
name="LibraryName">c8_fix_adapter_plugin_example_lib</preference>
```

```
        <preference
name="InitializeFunction">C8FIXConnPluginInitialize</preference>
        <preference
name="SendMessageFunction">C8FIXConnPluginSendMessage</preference>
       <preference name="ShutdownFunction">C8FIXConnPluginShutdown</
preference>
        <!-- Add implementation-specific settings to the following
section -->
        <section name="CustomSettings"/>
    </section>
```

<LibraryName>	The name of the run-time library (.dll/.so) where the Connection Plug-in functions are defined.
<InitializeFunction>	The name of the Connection Plug-in initialize function.
<SendMessageFunction>	The name of the Connection Plug-in send message function.
<ShutdownFunction>	The name of the Connection Plug-in shutdown function.
<CustomSettings>	Settings specific to the Connection Plug-in implementation.

Settings related to the Data Mapping Plug-in:

```
<section name="DataMappingPlugin">
        <preference
name="LibraryName">c8_fix_adapter_plugin_example_lib</preference>
        <preference name="InitializeFunction">C8FIXDMInitialize</
preference>
        <preference
name="ConvertFIXToTupleFunction">C8FIXDMConvertToTuple</preference>
        <preference
name="ConvertTupleToFIXFunction">C8FIXDMConvertToFIX</preference>
        <preference
name="ConvertFIXRawToTupleFunction">C8FIXDMConvertToTuple</
preference>
        <preference
name="ConvertTupleToFIXRawFunction">C8FIXDMConvertToFIX</
preference>
        <preference
name="DestroyFIXMessageFunction">C8FIXDMDestroyFIXMessage</
preference>
        <preference name="ShutdownFunction">C8FIXDMShutdown</
preference>
        <!-- Add implementation-specific settings to the following
section -->
        <section name="CustomSettings"/>
    </section>
```

<LibraryName>	The name of the run-time library (.dll/.so) where the Data Mapping Plug-in functions are defined.
<InitializeFunction>	The name of the Data Mapping Plug-in initialize function.

<ConvertFIXtoTuple-Function>	The name of the Data Mapping Plug-in FIX-to-tuple conversion function.
<ConvertTupleToFIX-Function>	The name of the Data Mapping Plug-in tuple-to-FIX conversion function.
<ConvertFIXRawToTupleFunction	The name of the Data Mapping Plug-in FIX raw-to-tuple conversion function.
<ConvertTupleToFIX-RawFunction>	The name of the Data Mapping Plug-in tuple-to-FIX raw conversion function.
<DestroyFIXMessage-Function>	The function called to properly deallocate a FIX message.
<ShutdownFunction>	The name of the Data Mapping Plug-in shutdown function.
<CustomSettings>	Settings specific to the Data Mapping Plug-in implementation.

Settings related to publishers:

```
 <section name="Publishers">
         <preference name="PublisherCount">2</preference>
         <!-- Publisher sections. These must be named Publisher1 -->
         <!-- through PublisherN, where N corresponds to the -->
         <!-- PublisherCount value specified. -->
         <section name="Publisher1">
                <preference name="C8StreamUri">ccl://localhost:6789/
Stream/Default/TestFIX/instream1</preference>
         </section>
         <section name="Publisher2">
                <preference name="C8StreamUri">ccl://localhost:6789/
Stream/Default/TestFIX/instream2</preference>
                 <preference
name="SessionFilter">FIX4.3::sender1::target1</preference>
                 <preference name="MessageFilter">A,8,9</preference>
         </section>
     </section>
```

<PublisherCount>	Specifies the number of publishers.
<Publisher1> ... <PublisherN>	Settings pertaining to each publisher, where N is the value specified in <PublisherCount>
<C8StreamUri>	The URI of the stream that the publisher publishes to.
<SessionFilter>	A comma-separated list of FIX session IDs to include when publishing. If this property is empty or unspecified, then all FIX messages belonging to all session IDs will be included.

<MessageFilter>	A comma-separated list of FIX message types to include when publishing to the stream. If this property is empty or unspecified, then FIX messages of all types will be included.

Settings related to subscribers:

```
 <section name="Subscribers">
        <preference name="SubscriberCount">2</preference>
       <!-- Subscriber sections. These must be named Subscriber1 -->
        <!-- through SubscriberN, where N corresponds to the -->
        <!-- SubscriberCount value specified. -->
        <section name="Subscriber1">
               <preference name="C8StreamUri">ccl://localhost:6789/
Stream/Default/TestFIX/outstream1</preference>
                <preference
name="SessionFilter">FIX4.2::sender1::target1</preference>
                <preference name="MessageFilter">7</preference>
         </section>
         <section name="Subscriber2">
               <preference name="C8StreamUri">ccl://localhost:6789/
Stream/Default/TestFIX/outstream2</preference>
         </section>
     </section>
```

<SubscriberCount>	Specifies the number of subscribers.
<Subscriber1> ... <SubscriberN>	Settings pertaining to each subscriber, where N is the value specified in <SubscriberCount>
<C8StreamUri>	The URI of the stream that the subscriber subscribes from.
<SessionFilter>	A comma-separated list of FIX session IDs to include when subscribing from the stream. If this property is empty or unspecified, then all FIX messages belonging to all session IDs will be included.
<MessageFilter>	A comma-separated list of FIX message types to include when subscribing from the stream. If this property is empty or unspecified, then FIX messages of all types will be included.

Settings for the logging-related section:

```
<section name="FileLogger">
        <preference name="Type">File</preference>
        <preference name="Layout">Text</preference>
        <preference name="LogLevel">Info</preference>
        <preference name="DetailsLevel">Normal</preference>
        <preference name="Filename">c:\logs\fix_adapter.log</
preference>
```

```
        <preference name="MaxSize">100MB</preference>
        <preference name="ReadPerms">owner</preference>
        <preference name="Rotate">10</preference>
    </section>
```

Note: The logging-related section is configured in the same fashion as CEP Server.

Other settings:

```
<preference name="LicenseFolder">C:\SybaseC8\enterprise-adapters
\fix\SySAM-2_0\licenses</preference>
```

<LicenseFolder>	Used to specify the location of the SySAM license folder.

Message Filtering

Messages can be filtered by session ID and/or FIX message type using the **SessionFilter** and **MessageFilter** tags in the config file:

```
<section name=''Publishers''>
                  <preference name=''PublisherCount''>2</preference>
                  <!-- Publisher sections. These must be named
Publisher1 -->
                <!-- through PublisherN, where N corresponds to the
-->
                  <!-- PublisherCount value specified. -->
                  <section name=''Publisher1''>
                           <preference name=''C8StreamUri''>ccl://
localhost:6789/Stream/Default/TestFIX/instream1</preference>
                           <preference
name=''SessionFilter''>FIX4.3::sender1::target1</preference>
                          <preference name=''MessageFilter''>A,8,9</
preference>
                  </section>
```

Note: The filters are an OR relationship. SessionFilter allows the filtering of FIX messages by FIX session ID, whereas MessageFilter allows the filtering of FIX messages by their FIX message type.

In the above example, either messages from session "FIX4.3::sender1::target1", or messages of message types A, 8, or 9, will be published to the stream with URI "ccl://localhost:6789/Stream/Default/TestFIX/instream1" (but not both).

If the SessionFilter is not specified, the behavior defaults to "all session IDs". Similarly, if no MessageFilter is specified, the behaviour defaults to "all message types".

Filters can be specified such that the same FIX message is published to multiple streams, and the SessionFilter and MessageFilter tags can also be used for Subscribers when reading messages from the CEP streams.

Connection Plug-In

The Connection Plug-in allows connectivity between the FIX adapter and a FIX data source, such as a counterparty FIX session, a file system, or a database table.

Three distinct implementations of the Connection Plug-in are provided with the FIX Adapter: the QuickFIX Plug-in, the File Plug-in, and the Database Plug-in.

QuickFIX Plug-In

QuickFIX is a full feature open source FIX engine.

The QuickFIX plug-in allows the FIX adapter to exchange FIX messages with a counterparty FIX server. When in use, the QuickFIX connection plug-in creates its own set of log files. The QuickFIX logs are stored in the location specified in the **FileLogPath** preference in the QuickFIX configuration file.

Example QuickFIX-specific configuration files (such as `sample-quickfix-acceptor.cfg` and `sample-quickfix-initiator.cfg`) are included with the FIX Adapter.

Custom Settings for the QuickFIX Plug-in

Custom settings for the QuickFIX plug-in include:

```
<section name="CustomSettings">
                <preference name="ConfigFilePath">sample-quickfix-
initiator.cfg</preference>
    </section>
```

If you want to configure the FIX Adapter to use the file persistence feature of QuickFix in order to preserve message sequence numbers when a FIX session is re-established, you must set the "MessageStoreType" property to "file":

```
    <section name="ConnectionPlugin">
        <preference name="LibraryName">c8_adapters_fix_plugins_lib</
preference>
        <preference
name="InitializeFunction">c8FixQuickfixPluginInitialize</
preference>
        <preference
name="SendMessageFunction">c8FixQuickfixPluginSendMessage</
preference>
        <preference
name="ShutdownFunction">c8FixQuickfixPluginShutdown</preference>
        <!-- Add implementation-specific settings to the following
section -->
        <section name="CustomSettings">
           <preference name="ConfigFilePath">c:\sybasec8\enterprise-
adapters\fix\examples\enterpriseadapters\fixadapter\sample-
```

```
quickfix-acceptor.cfg</preference>
            <preference name="MessageStoreType">file</preference>
        </section>
```

In addition, you will also need to add the appropriate settings to the QuickFIX-specific configuration file. These settings are:

- `FileStorePath`.
- `PersistMessages`, set to Y.
- `RefreshOnLogon`, set to Y.

For information on configurinq QuickFIX, consult the QuickFIX documentation and website located at http://www.quickfixengine.org.

Database Plug-In

The Database Plug-in allows the FIX adapter to read and write FIX messages to a database table, via ODBC.

The **InputConnectString** setting specifies the ODBC connection string for the database from which the plug-in reads FIX messages. If this setting is present, the plug-in will read FIX messages from a database table.

The **OutputConnectString** setting specifies the ODBC connection string for the database to which the plug-in writes FIX messages. If this setting is present, the plug-in will write FIX messages to a database table.

The **InputQuerySQL** setting specifies the SQL query statement used to retrieve FIX messages from the database.

The **OutputInsertSQL** setting specifies the SQL insert statement used to write a single FIX message to the database.

The Database Plug-in supports polling when reading. To enable polling, set the **InputPollInterval** setting to a value greater than 0 (where this value represents the number of seconds to wait between polls). Polling is disabled if the **InputPollInterval** value is set to 0, in which case the plug-in will only read from the database table once.

Custom settings for the Database Plug-in:

```
    <section name="CustomSettings">
        <preference name="InputConnectString">driver={IBM DB2 ODBC
DRIVER};DBALIAS=cep_l;hostname=localhost;port=6789;protocol=Local;u
id=username;pwd=password</preference>
        <preference
name="OutputConnectString">Dsn=cb;Trusted_Connection=yes</
preference>
        <preference name="InputPollInterval">0</preference>
        <preference name="InputQuerySQL">select * from FixMessages</
preference>
        <preference name="OutputInsertSQL">insert into FixMessages
values (?,null)</preference>           </section>
```

File Plug-In

The File Plug-in allows the FIX adapter to read and write FIX messages to the file system.

The File Plug-in supports polling when reading. To enable polling, set the **InputPollInterval** setting to a value greater than 0 (where this value represents the number of seconds to wait between polls). Polling is disabled if the **InputPollInterval** value is set to 0, in which case the plug-in will only read from the file once.

When writing, the **OutputMaxFileSize** setting determines (in KB) the maximum file size before a new file will be created.

The **InputFilePath** specifies the name and path of the file from which the plug-in reads FIX messages. If the setting is present, the plug-in will read FIX messages from a file.

The **OutputFilePath** specifies the name and path of the file to which the plug-in writes FIX messages. If this setting is present, the plug-in will write FIX messages to a file.

Custom settings for the File Plug-in:

```
<section name="CustomSettings">
                    <preference name="InputFilePath">message-in.txt</
preference>
                  <preference name="OutputFilePath">message-out.txt</
preference>
                  <preference name="InputPollInterval">1</preference>
                  <preference name="OutputMaxFileSize">1</preference>
    </section>
```

Writing a Custom Connection Plug-In

The Connection Plug-in is highly customizable and unique implementations can be written using C or C++.

When writing a custom Connection Plug-in, include the following functions and ensure that their method signatures match what is given here:

```
    //initialize the connection plugin
    C8Bool C8FIXConnInitialize();

    //send message to message source
    C8Bool C8FIXConnSendMessage( const void* message)

    //shutdown plug-in, disconnect to the message source
    C8Bool C8FIXConnShutdown();
```

The **C8FIXConnInitialize()** function is called by the FIX Adapter when it starts up. Any initialization required by the plug-in, such as establishing a connection to the FIX message source, must be performed here.

The **C8FIXConnSendMessage()** function is called by the FIX Adapter after it receives a tuple from the CEP server and has converted the tuple into a FIX message via the Data Mapping plug-in **C8FIXDMConvertToFIX()** function. The FIX message that is passed as the parameter of

this function call is of the form returned by the Data Mapping plug-in **C8FIXDMConvertToFIX()** function.

The **C8FIXConnShutdown()** function is called by the FIX Adapter when the adapter is shutting down. Any clean-up required by the plug-in must be performed here.

Note: The functions can have any name but they must match the configuration file you intend to use. The function signatures must conform to what is noted above. When using C++ to write a custom Connection Plug-in, ensure that the functions above use C-style calling conventions and name mangling.

For the FIX Adapter to function as an input adapter, set the Connection Plug-in to receive FIX messages from the message source. Once a FIX message has been received, the plug-in notifies the FIX Adapter by invoking the following function:

```
//callback from the Connection Plug-in to notify the FIX Adapter
//that a FIX message was received from the external datasource
FIX_ADAPTER_EXPORT C8Bool C8FIXMessageReceived( void* message, char*
msgType, char* sessionId );
```

Note: This function is declared in the `c8_fix_adapter.h` header file installed under the `adapters\fix\include` directory. The Connection plug-in code should include this header file.

The Connection plug-in can also log messages through the Adapter by calling the following functions, which are similarly declared in `c8_fix_adapter.h`:

```
//log messages/errors through the FIX adapter
FIX_ADAPTER_EXPORT void C8FIXLogMessage( C8Int i_err_code, const
enum C8LogLevels i_level, const C8Char *i_err_text, ... );
FIX_ADAPTER_EXPORT void C8FIXLogWarning(C8Int i_err_code, const
C8Char *i_err_text_id, ...);
FIX_ADAPTER_EXPORT void C8FIXLogError(C8Int i_err_code, const C8Char
*i_err_text_id, ...);
```

The Connection plug-in can invoke functions provided by the out-of-process CEP C SDK. When doing so, there is no need to call **C8ClientSDKInitialize()** and **C8ClientSDKShutdown()** functions as these have already been called by the FIX Adapter.

Additionally, you must specify the names of the custom functions in the configuration file:

```
        <preference name="LibraryName">c8_adapters_fix_plugins_lib</
preference>
        <preference
name="InitializeFunction">c8FixQuickfixPluginInitialize</
preference>
        <preference
name="SendMessageFunction">c8FixQuickfixPluginSendMessage</
preference>
        <preference
name="ShutdownFunction">c8FixQuickfixPluginShutdown</preference>
```

When you have finished writing the customized plug-in, compile it in a manner similar to:

```
g++ -I${HOME}/sybasec8/server/sdk/c/include \
-o libcustomized_fix_app_lib.so \
-L${HOME}/sybasec8/server/sdk/c/lib/ \
-lc8_adapters_fix_lib -lc8_sdk_server_lib -lnspr4 -lplc4 \
-llibxml2 -fPIC -shared customized_fix_app.cpp
```

Ensure that the name of the shared library for the new customized plug-in matches what is specified in the config file:

```
        <section name="ConnectionPlugin">
            <preference name="LibraryName">customized_fix_app_lib</
preference>
```

Finally, ensure that the new shared library is placed in a directory that is in the **LD_LIBRARY_PATH**.

Note: When writing a custom plug-in, the FIX adapter-related libraries that should be linked to the plug-in are `c8_adapter_fix_lib.dll` on Windows and `libc8_adapters_fix_lib.so` on Linux/Solaris.

Data Mapping Plug-In

The Data Mapping Plug-in converts FIX messages into tuple columns and tuple columns into FIX messages.

The Data Mapping plug-in also performs validations on FIX messages against a database.

The default Data-Mapping Plug-in implementation is designed to match the three Connection Plug-in implementations. It is also highly customizable, and unique implementations can be written using C or C++. When writing a custom Data Mapping Plug-in, include the following functions and ensure that their method signatures match what is given here:

```
//initialize the Data Mapping plug-in
    C8Bool C8FIXDMInitialize();

    //convert from FIX message to CEP tuple
    C8Message* C8FIXDMConvertToTuple( const void* message, const
C8Schema* schema );

    //convert from CEP tuple to FIX message
    void* C8FIXDMConvertToFIX( const C8Message* tuple, char*&
session_id, char*& msgtype );

    //convert from raw FIX message to CEP tuple
    C8Message* C8FIXDMConvertFIXRawToTuple( const char* message,
const  C8Schema* schema );

    //convert from CEP tuple to a raw FIX message
    char* C8FIXDMConvertTupleToFIXRaw( const C8Message* tuple );

    //properly deallocate a FIX message returned by the
C8FIXDMConvertToFIX function
    void C8FIXDMDestroyFIXMessage( void* message );
```

```
//shutdown the Data Mapping plug-in
C8Bool C8FIXDMShutdown();
```

The **C8FIXDMInitialize()** function is called by the FIX Adapter when it starts up. Any initialization required by the plug-in must be performed here.

The **C8FIXDMConvertToTuple()** function is called by the FIX Adapter after it receives a FIX message from the Connection plug-in and needs to convert it into a tuple before sending it to the CEP server.

The **C8FIXDMConvertToFIX()** function is called by the FIX Adapter after it receives a tuple from the CEP server and needs to convert the tuple into a FIX message before sending it to the Connection plug-in.

The **C8FIXDMConvertFIXRawToTuple()** function is called to convert a FIX message in raw (string) format into a CEP tuple. This function is optional.

The **C8FIXDMConvertTupleToFIXRaw()** function is called to convert a CEP tuple into a FIX message in raw (string) format. This function is optional.

The **C8FIXDMDestroyFIXMessage()** is called by the FIX Adapter to properly deallocate a FIX message that was previously returned by the C8FIXDMConvertToFIX() function.

The **C8FIXDMShutdown()** function gets called by the FIX Adapter when the adapter is shutting down. Any clean-up required by the plug-in must be performed here.

Note: When using C++ to write a custom Data Mapping Plug-in, ensure that functions are declared using C-style calling conventions and name mangling.

Note: When writing a custom plug-in, the FIX adapter-related libraries that should be linked to the plug-in are `c8_adapter_fix_lib.dll` on Windows and `libc8_adapters_fix_lib.so` on Linux/Solaris.

The Data Mapping plug-in can invoke functions provided by the out-of-process CEP C SDK. When doing so, there is no need to call **C8ClientSDKInitialize()** and **C8ClientSDKShutdown()** functions as these have already been called by the FIX Adapter.

The Data Mapping plug-in can also log messages through the FIX Adapter, by calling the following functions (declared in the `c8_fix_adapter.h` header file installed under the `adapters\fix\include` directory):

```
//log messages/errors through the FIX adapter
FIX_ADAPTER_EXPORT void C8FIXLogMessage( C8Int i_err_code, const
enum C8LogLevels i_level, const C8Char *i_err_text, ... );
FIX_ADAPTER_EXPORT void C8FIXLogWarning(C8Int i_err_code, const
C8Char *i_err_text_id, ...);
FIX_ADAPTER_EXPORT void C8FIXLogError(C8Int i_err_code, const C8Char
*i_err_text_id, ...);
```

The Data Mapping plug-in code should include this header file if these logging functions are to be used.

Custom Settings for the Data Mapping Plug-In
Custom settings for the Data Mapping Plug-in include:

```
<section name="CustomSettings">
                    <preference name="ValidateFIXMessages">true</
preference>
                 <preference name="FIXMessageC8Name">C8_FIXMsgBody</
preference>
                    <section name="FieldMappings">
            <preference name="FieldMappingCount">3</preference>
            <!-- Field mapping sections. These must be named
FieldMapping1 -->
          <!-- through FieldMappingN, where N corresponds to the -->
            <!-- FieldMappingCount value specified. -->
            <section name="FieldMapping1">
                <preference name="C8Name">MsgType</preference>
                <preference name="FIXTag">35</preference>
            </section>
            <section name="FieldMapping2">
                <preference name="C8Name">SenderCompID</preference>
                <preference name="FIXTag">49</preference>
            </section>
            <section name="FieldMapping3">
                <preference name="C8Name">TargetCompID</preference>
                <preference name="FIXTag">56</preference>
            </section>
                </section>
    </section>
```

The **ValidateFIXMessages** setting determines if the structure of the FIX message should be validated. If set to true, validation is enabled and FIX messages that do not pass validation are rejected.

The **FIXMessageC8Name** setting indicates the name of the CEP stream column that will be used to store the entire FIX message (defaults to "C8_FIXMsgBody").

The **FieldMappings** setting describes the field mapping rules used to map FIX message fields to CEP stream columns.

The **FieldMappingCount** indicates the number of field mapping rules. If set to 0, field mapping is disabled.

The FieldMapping1 ... FieldMappingN setting pertains to each field mapping rule, where N is the value specified in **FieldMappingCount**.

The **C8Name** setting specifies the CEP stream column name.

The **FIXTag** setting specifies the FIX message field tag.

Running the Example FIX Adapter

The FIX adapter example demonstrates how the FIX adapter functions as both an input and an output adapter.

The example is located at: `/SybaseC8/enterprise-adapters/fix/examples/EnterpriseAdapters/FixAdapter`.

This example makes use of three FIX Adapter instances.

The first FIX adapter instance is configured to use the File Plug-in. It reads FIX messages from a file and publishes them to CEP. In addition, it subscribes to a CEP stream and writes FIX messages received from that stream out to a different file. This instance functions as both an input and an output adapter.

The second FIX adapter instance is configured to use the QuickFIX plug-in, as an initiator. It subscribes to a CEP stream and transmits FIX messages received from that stream over a FIX session. This instance functions as an output adapter.

The third FIX adapter instance is also configured to use the QuickFIX plug-in, but as an acceptor. It receives FIX messages over a FIX session from the second FIX adapter instance and publishes those FIX messages back to CEP. This instance functions as an input adapter.

In order for this example to function properly, the adapters must be started in the order listed.

Note: If there are any problems in running the example, ensure that the SySAM location in the xml files is correct, and that the `.../SybaseC8/server/bin` directory is in the path.

To run FIX adapter example:

1. Load the FIXAdapter.ccp project onto Sybase CEP Studio.

2. Open 3 CMD windows and start a different instance of the FIX adapter with the following:

CMD 1: c8_fix_adapter --config=sample-prefs-quickfix-acceptor.xml

CMD 2: c8_fix_adapter --config=sample-prefs-quickfix-initiator.xml

CMD 3: c8_fix_adapter --config=sample-prefs-fileplugin.xml

3. Once the FIX adapter instances are working, open stream viewers on the two input and output streams to verify that FIX messages are flowing in and out of the Sybase CEP Server.

The following sections must be altered if they are expected to run on a UNIX-based system:

- `<section name="MappedFileLogger">`
- `<preference name="Filename">C:\Program Files \SybaseC8\Server\logs\server.log</preference>` to `<preference`

```
name="Filename">/software/SybaseC8/server/logs/
server.log</preference>
```

- `<section name="MappedFileLogger"> <section name="C8/General"> <preference name="LicenseFolder">C:\Program Files\SybaseC8\Server\SySAM-2_0\licenses</preference>` to `<preference name="LicenseFolder">/software/SybaseC8/server/SYSAM-2_0/licenses</preference>`

The following location contains additional configuration file examples for using QuickFix, database, or file connectivity; however, these are not intended to be running examples: `.../SybaseC8/enterprise-adapters/fix/adapters/fix/sample`

Index

O

P

Q

R

S

T

W