software AG

webMethods EntireX

EntireX Adapter

Version 10.7

October 2020

WEBMETHODS

This document applies to webMethods EntireX Version 10.7 and all subsequent releases.

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

Document ID: EXX-EEXXADAPTER-107-20220422

Table of Contents

webMethods EntireX Adapter for Integration Server

The webMethods EntireX Adapter for Integration Server enables webMethods Integration Server to consume (IS outbound) as well as provide (IS inbound) EntireX services. EntireX supports bidirectional integration with Natural, COBOL and PL/I on z/OS, BS2000, z/VSE and other platforms. This includes zero footprint integration with CICS, IMS and IBM i (AS/400).

This document describes how to install, configure and use the webMethods EntireX Adapter for Integration Server. It contains information for administrators and application developers who want to call remote procedures on mainframes and other hosts, using the EntireX RPC technique. To use this guide effectively, you should be familiar with EntireX's Remote Procedure Call (RPC) technology. See *EntireX RPC Programming*.

Introduction	EntireX Adapter's functionality and features such as calling remote procedures on mainframes and other hosts.
Package Management	How to set up and manage your EntireX Adapter packages and set up access control lists (ACL).
Adapter Connections	How to configure and manage EntireX Adapter connections.
Adapter Services	How to configure and manage EntireX Adapter services. EntireX Adapter services allow clients to interact with EntireX RPC servers, using a configured adapter connection.
Listeners	How to configure and manage listeners.
Settings and Information	Monitoring and tracing in the IS Administration Console.
Application Monitoring	How to monitor response times in your distributed applications.
Built-in Services for Creating Document Types, Flows and IDL Files	Built-in services are available in the WmEntireX package for creating document types, flows and IDL files from EntireX Adapter objects and/or IDL files.
Direct RPC	Direct communication from an RPC client or server to the EntireX Adapter.
IDL Extraction from Integration Server	Creating an IDL file can be automated using IDL Extractor for Integration Server.
SSL/TLS and Security Support	SSL/TLS and Security Support in the EntireX Adapter.
Preparing for CICS Socket Listener	The EntireX Adapter supports connections to CICS Socket Listener to call DFHCOMMAREA, Channel Container, and Large Buffer programs. Here we provide details and installation information.
Preparing IBM CICS for ECI	The EntireX Adapter supports connections to IBM CICS® ECI to call programs in CICS. Here we describe how to set up the External Call Interface (ECI) within CICS.
Preparing for IMS Connect	The EntireX Adapter supports connections to IMS Connect. Here we describe how to extract from Message Format Service (MFS).

Converting IS Data Structures with the COBOL Converter	Adapter services for connection type COBOL Converter can convert Integration Server data structures to/from a byte array representing the COBOL binary data.
Extracting IDL using the REST API	How to extract an IDL file using the REST API, for example to be used in a command line.
Creating or Updating Connections using the REST API	How to create or update connections, adapter services and listener objects using the REST API.
Post-installation Steps for AS/400	After installation of the EntireX Adapter, the AS/400 connection type is not visible and not usable. To access the AS/400 system you need the IBM ToolBox for Java (JTOpen). This document describes how to download and install the IBM ToolBox for Java.

Related Literature

- **Designer**
  - Integration Server Wrapper
  - IDL Extractor for Integration Server
- **Error Messages**
  - *Message Class 0800 - webMethods EntireX Adapter for Integration Server*
  - *Message Class 2011 - Connections to IMS Connect*
  - *Message Class 2012 - Connections to CICS ECI*
  - *Message Class 2014 - COBOL Converter*
  - *Message Class 2015 - AS/400*
  - *Message Class 2016 - Connections to CICS Socket Listener*
  - *Message Class 2023 - Connections to ApplinX*
  - *Message Class 0400 - Direct RPC*
  - *Message Class 1018 - EntireX RPC-ACI Bridge*
- **webmethods Product Suite**
  - *Integration Server Administrator's Guide*. See *https://empower.softwareag.com/Products/Documentation/default.asp* under "webMethods" > "webMethods Product Suite" > "Integration Server" > "Documentation by Product"
  - the documentation on the Designer. See *https://empower.softwareag.com/Products/Documentation/default.asp* under "webMethods" > "webMethods Product Suite" > "Documentation by Product"
  - The Designer Online Help.
  - *Software AG Installer*. See *https://empower.softwareag.com/Products/Documentation/default.asp* under "Software AG Installer and Update Manager"

- *webMethods Adapter Development Kit User's Guide*. See *https://empower.softwareag.com/Products/Documentation/default.asp* under "webMethods" > "webMethods Adapters and eStandards Modules" > "Adapters"

- **Other**
  - the documentation of the Eclipse Update Manager, located at *http://help.eclipse.org/helios/topic/org.eclipse.platform.doc.user/tasks/tasks-129.htm*
  - Natural Product Documentation at *https://empower.softwareag.com/Products/Documentation/default.asp*.

1 About this Documentation

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

In addition, you can also access the cloud product documentation via **https://www.softwareag.cloud**. Navigate to the desired product and then, depending on your solution, go to “Developer Center”, “User Center” or “Documentation”.

Product Training

You can find helpful product training material on our Learning Portal at **https://knowledge.softwareag.com**.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

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

2 Introduction to the EntireX Adapter

The EntireX Adapter is an add-on to the webMethods Integration Server that enables you to interact with remote applications on mainframes and other hosts. The adapter provides seamless and real-time bi-directional communication with EntireX RPC servers and clients, Natural RPC servers and clients, ACI Servers, ApplinX, IMS Connect, CICS and IBM® AS/400®. Using the EntireX Adapter, Integration Server applications can create and run services and listeners that can exchange messages with server and client applications on mainframe and other hosts.

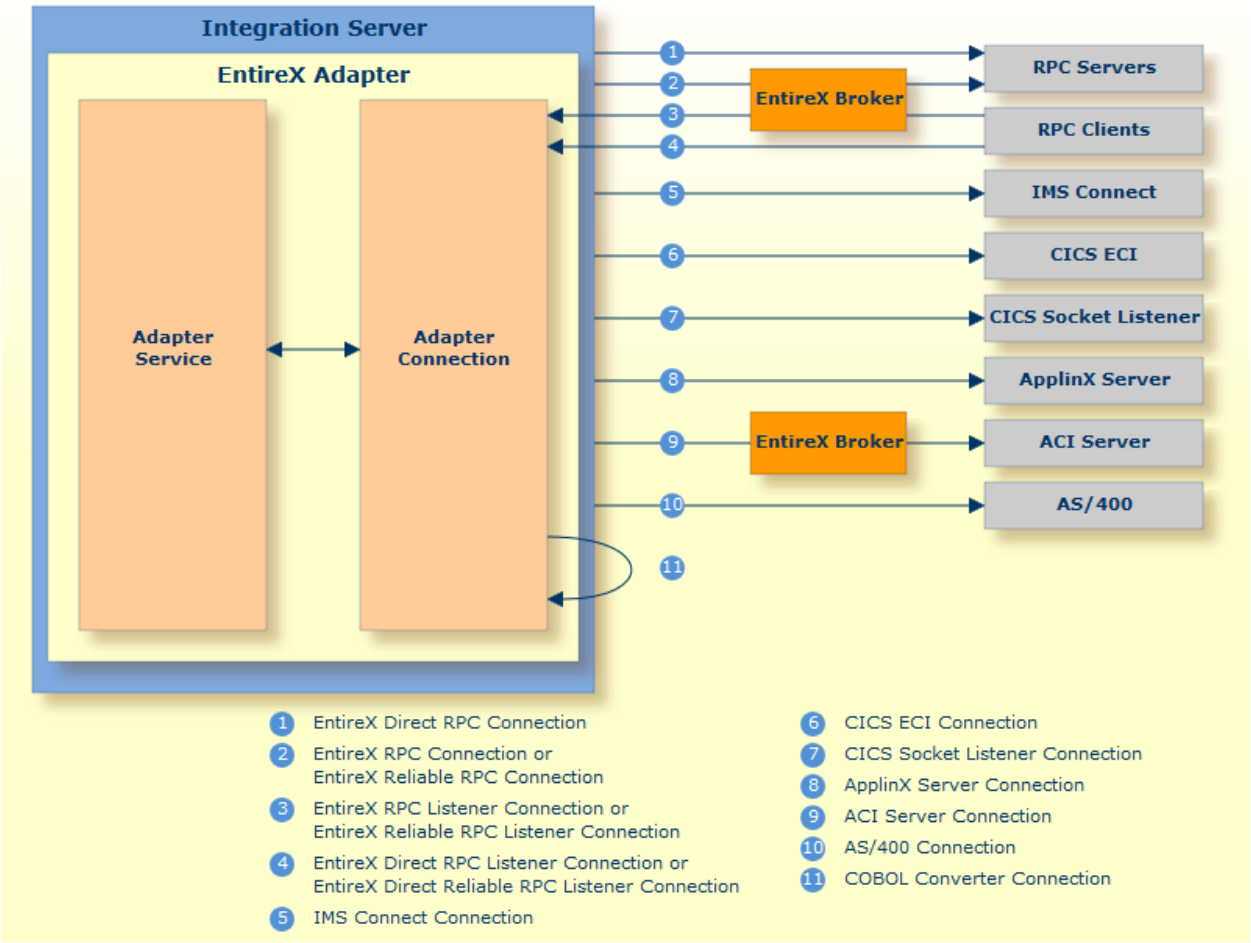
See also *EntireX Adapter Prerequisites* in the EntireX Release Notes.

Architecture and Components

The EntireX Adapter enables you to create, configure and use the following components:

- **Adapter Connections**
  Adapter connections enable the webMethods Integration Server to connect to COBOL and Natural applications on different platforms. The EntireX Adapter's connections are generated at design time with the Software AG Designer.
- **Adapter Services**
  Adapter services call remote COBOL and Natural server applications. The EntireX Adapter's services are generated at design time with the Software AG Designer.
- **Listener Notifications**
  Adapter listeners are called by remote COBOL and Natural client applications. The EntireX Adapter's listeners are generated at design time with the Software AG Designer.

The following diagram illustrates how the EntireX Adapter interfaces with EntireX RPC servers and clients.


Integration Server
EntireX Adapter
Adapter Service
Adapter Connection
EntireX Broker
EntireX Broker
RPC Servers
RPC Clients
IMS Connect
CICS ECI
CICS Socket Listener
ApplinX Server
ACI Server
AS/400
1 EntireX Direct RPC Connection
2 EntireX RPC Connection or EntireX Reliable RPC Connection
3 EntireX RPC Listener Connection or EntireX Reliable RPC Listener Connection
4 EntireX Direct RPC Listener Connection or EntireX Direct Reliable RPC Listener Connection
5 IMS Connect Connection
6 CICS ECI Connection
7 CICS Socket Listener Connection
8 ApplinX Server Connection
9 ACI Server Connection
10 AS/400 Connection
11 COBOL Converter Connection

3 Package Management

The EntireX Adapter is provided as a package called WmEntireX. This chapter describes how to set up and manage your EntireX Adapter packages, set up access control lists (ACLs).

Package Dependency Requirements and Guidelines

This section contains a list of dependency requirements and guidelines for user-defined packages. For instructions on setting package dependencies, see *webMethods Service Development Help*.

- A user-defined package must be dependent on its associated adapter package, WmEntireX. (The WmEntireX package is dependent on the WmART package.)
- Package dependencies ensure that at startup the Integration Server automatically loads or reloads all packages in the proper order: the WmART package first, the adapter package next, and the user-defined package(s) last. The WmART package is automatically installed when you install the Integration Server. You should not need to reload the WmART package manually.

  Tip: When you create connections, adapter services, and adapter notifications, define them in user-defined packages rather than in the WmEntireX package. This makes it easier to manage packages. As you create user-defined packages in which to store connections, adapter services, and adapter notifications, use the package management functionality provided in the Designer and set the user-defined packages to be dependent on the WmEntireX package. That way, whenever the WmEntireX package loads or reloads, the user-defined packages load automatically as well.

- If the connections and adapter services of an adapter are defined in different packages, then:
  - a package that contains the connection(s) must have a dependency on the adapter package.
  - packages that contain adapter services must have a dependency on their associated connection package.
- Keep connections for different adapters in separate packages so that you do not create interdependencies between adapters. If a package contains connections for two different adapters, and you reload one of the adapter packages, the connections for both adapters will reload automatically.
- The Integration Server will not allow you to enable a package if it has a dependency on another package that is disabled. That is, before you can enable your package, you have to enable all packages on which your package depends. For information on enabling packages, see ***Enabling and Disabling Packages***.
- The Integration Server will allow you to disable a package even if another package that is enabled has a dependency on it. Therefore, you have to manually disable any user-defined packages that have a dependency on the adapter package before you disable the adapter package. For information on disabling packages, see ***Enabling and Disabling Packages***.

- You can give connections, adapter services, and notifications the same name provided that they are in different folders and packages.

Enabling and Disabling Packages

All packages are automatically enabled by default. To temporarily prohibit access to the elements in a package, disable the package. When you disable a package, the server unloads all of its elements from memory. Disabling a package prevents the Integration Server from loading that package at startup.

To enable a package

1 Open the IS Administration Console if it is not already open.

2 In the **Packages** menu of the navigation area, click **Management**.

3 Click **No** in the **Enabled** column.

As a result, the server displays a tick (✓) and "Yes" in the **Enabled** column.

Note: Enabling an adapter package will not cause its associated user-defined package(s) to be reloaded. For information on reloading packages, see the *webMethods Service Development Help*.

Important: Before you manually enable a user-defined package, you first have to enable its associated adapter package (WmEntireX). Similarly, if your adapter has multiple user-defined packages, and you want to disable some of them, disable the adapter package first. Otherwise, errors will be issued when you try to access the remaining enabled user-defined packages. See *Managing Packages* in the *webMethods Service Development Help* doumentation or choose **Software AG Designer Guides** > **webMethods Service Development Help** > **Managing Packages** in the Designer online help.

To disable a package

1 Open the IS Administration Console if it is not already open.

2 In the **Packages** menu of the navigation area, click **Management**.

3 Click **Yes** in the **Enabled** column for the package that you want to disable. The server issues a prompt to verify that you want to disable the package. Click **OK** to disable the package. When the package is disabled, the server displays "No" in the **Enabled** column.

Note: A disabled package will remain disabled until you explicitly enable it using the IS Administration Console. It will not be listed in the Designer.

Importing and Exporting Packages

To export packages, use the Designer. You can export the package to a ZIP file and save it to your hard drive. The ZIP file can then be imported for use by another Integration Server.

Important: Do not rename packages you export. The rename function is comparable to moving a package: when you import the renamed package, you lose any triggers, connections, and notifications associated with this package. For details on managing packages, see *webMethods Service Development Help*.

Group Access Control

To control which development group has access to which adapter services, use access control lists (ACLs). You can use ACLs to prevent one development group from inadvertently updating the work of another group, or to allow or deny access to services that are restricted to one group but not to others. For general information on assigning and managing ACLs, see *webMethods Service Development Help*.

4 EntireX Adapter Connections

This chapter describes how to configure and manage EntireX Adapter connections.

Tip: It is recommended to create connections with the webMethods Integration Server Wrapper of the Designer. Connections that are used dynamically can be created with the IS Administration Console.

Before Configuring or Managing Adapter Connections

To prepare to configure or manage adapter connections

1. Install the webMethods Integration Server and the EntireX Adapter on the same machine, using the Software AG Installer.
2. Make sure you have webMethods administrator privileges so that you can access the EntireX Adapter's administrative screens. See the *Administering webMethods Integration Server* for information on setting user privileges.
3. Start your Integration Server and the IS Administration Console, if they are not already running.
4. Using the IS Administration Console, make sure the WmEntireX package is enabled. See ***Importing and Exporting Packages*** for instructions.
5. If you have not already done so, use the Designer to create a user-defined package to contain the connection. See ***Package Management***.

Configuring Adapter Connections

When you configure EntireX Adapter connections, you specify information that the Integration Server uses to connect to an EntireX RPC server or client, or to IMS Connect or CICS ECI. You create EntireX Adapter connections using the Designer. This ensures that the information generated from the interface definition stored in the Software AG IDL file is present. A wizard inside the Designer allows you to configure the most common parameters of the connection. Using the IS Administration Console, you have access to all parameters of a connection.

The EntireX Adapter supports the following connection types:

See *RPC-based Components* for details on EntireX RPC servers and clients. The following table gives an overview of the connection types discussed above:

Connection Type	Connects to	Integration Server outbound	Supports Transactions of Type	Integration Server inbound	Note
RPC Connection	RPC Server via EntireX Broker	x	local	-	1,6
Direct RPC Connection	RPC Server	x	local	-	1,6
Reliable RPC Connection	RPC Server via EntireX Broker	x	local	-	1,6
RPC Listener Connection	RPC Client via EntireX Broker	-	n/a	x	2
Direct RPC Listener Connection	RPC Client	-	n/a	x	2
Reliable RPC Listener Connection	RPC Client via EntireX Broker	-	n/a	x	2
Direct Reliable RPC Listener Connection	RPC Client	-	n/a	x	2
IMS Connect Connection	IMS Connect	x	-	-	1
CICS ECI Connection	CICS ECI	x	-	-	1
CICS Socket Listener Connection	CICS Socket Listener	x	local	-	1,6
ApplinX Connection	ApplinX Server	x	-	-	3
ACI Server Connection	ACI Server via EntireX Broker	x	-	-	1
COBOL Converter Connection	Integration Server (internal)	-	n/a	-	1
AS/400 Connection	AS/400	x	-	-	1,4
Adabas Replication Listener Connection	Adabas Replication Server via EntireX Broker	-	n/a	x	5
Adabas Replication Direct Listener Connection	Adabas Replication Server	-	n/a	x	5

Notes:

1. The Designer wizard additionally creates an EntireX Adapter service for each program of the IDL file; see ***EntireX Adapter Services***.
2. The Designer wizard additionally creates an EntireX Adapter listener; see also ***Listeners***.
3. The ApplinX Designer additionally creates an EntireX Adapter service for each ApplinX path procedure.
4. After installation of the EntireX Adapter, the AS/400 connection type is not visible and not usable. To access the AS/400 system you need the IBM ToolBox for Java (JTOpen). See ***Post-installation Steps for AS/400***.
5. These connection types are used by the product "Adabas Replication Service for webMethods Integration Server". For details see the documentation of this product.
6. For transaction support, refer to the respective connection parameters under ***Editing Adapter Connections*** and Appendix B in *webMethods Adapter Development Kit User's Guide Version 6.5*.

To create a connection

1 Start the Designer.

2 Navigate to the Software AG IDL file you want to use.

3 On the IDL file, right-click and choose **Generate webMethods IS Connection from Software AG IDL...**

4 Follow the wizard (Integration Server Wrapper), which guides you through the creation of the adapter connection and the adapter services or adapter listener.

As a result, the connection you created appears on the adapter's connections screen and in the Designer **Package Navigator** view.

Note: It is not possible to generate new connections using **Configure New Connection** on the **Connections screen** of the EntireX Adapter in the IS Administration Console because such connections would not have any metadata. Only connections generated by the Designer have the mandatory metadata. However, you can create a new connection by copying an existing connection on the **Connections screen**. This is useful for dynamically selected connections where the metadata is not generated from the interface definition (Software AG IDL file). To use dynamically selected connections with `$connectionName`, see chapter 8 in *webMethods Adapter Development Kit User's Guide Version 6.5*, see *https://empower.softwareag.com/Products/Documentation/default.asp* under "webMethods" > "Adapters".

Viewing Adapter Connection Parameters

You can view a connection's parameters using the IS Administration Console or the Designer.

To view the parameters for a connection using the IS Administration Console

1 In the **Adapters** menu in the IS Administration Console navigation area, click **EntireX Adapter**.

2 On the **Connections** screen, click the **View** icon for the connection you want to see. The **View Connection** screen displays the parameters for the connection. For descriptions of the connection parameters, see *Configuring Adapter Connections*.

3 Click **Return to EntireX Adapter Connections** to return to the main connections screen.

To view the parameters for a connection using the Designer

1 Start the Designer if it is not already running.

2 From the Designer **Package Navigator** view, open the package and folder in which the connection is located.

3 Click the connection you want to view. The parameters for the connection appear on the **Connection Information** tab. For descriptions of the connection parameters, see *Configuring Adapter Connections*.

Editing Adapter Connections

If the login information for a server changes, or if you want to redefine parameters that a connection uses when connecting to a server, you can update the connection's parameters. This section describes how to edit a connection and lists parameter settings for the connection types covered. The following topics are covered:

Editing a Connection

You edit (and modify) adapter connections using the IS Administration Console.

To edit a connection

1 In the **Adapters** menu in the IS Administration Console navigation area, click **EntireX Adapter**.

2 Make sure that the connection is disabled before editing it. See *Disabling Adapter Connections* for instructions.

3 On the **Connections** screen, click the **Edit** icon for the connection you want to edit. The **Edit Connection** screen displays the current parameters for the connection. Update the connection's parameters by typing or selecting the values you want to specify. The tables below give a complete listing of parameters for the following connection types:

- *RPC Connection*
- *Direct RPC Connection*
- *Reliable RPC Connection*
- *RPC Listener Connection and Reliable RPC Listener Connection*
- *Direct RPC and Direct Reliable RPC Listener Connection*

- *IMS Connect Connection*
- *CICS ECI Connection*
- *CICS Socket Listener Connection*
- *ApplinX Connection*
- *ACI Server Connection*
- *COBOL Converter Connection*
- *AS/400 Connection*

4 On the Connections screen, edit the connection management properties (connection pooling). The tables in *Common Connection Management Properties (for all Connection Types)* give a complete listing of parameters.

5 Click **Save Changes** to save the connection and return to the Connections screen.

Connection Parameters for RPC Connections

Parameter	Description	Default	Can be set from the Designer	Can be set dynamically with service
Broker ID	The ID of the broker you want to connect to. This ID consists of a host and an optional port. Default for the port is "1971".	localhost	Yes	No
Server Address	The address of the RPC server registered to the broker above. The address is given in the format `<class>/<server>/<service>`.	RPC/SRV1/CALLNAT	Yes	Yes
Logon User	The name of the user to log on to the broker.		Yes	Yes
Logon Password	The password for the user above.		Yes	Yes
Retype Logon Password	Verification of the password.		Not applicable	Not applicable
Encryption	Deprecated. For encrypted transport we strongly recommend using the Secure Sockets Layer/Transport Layer Security protocol. See *SSL/TLS, HTTP(S), and Certificates with EntireX* in the platform-independent Administration documentation.			
Encoding	The character encoding used for the RPC connection to the *EntireX Broker*.		Yes	No

Parameter	Description	Default	Can be set from the Designer	Can be set dynamically with service
	Default: the encoding of the Integration Server. Enable character conversion in the broker by setting the service-specific attribute `CONVERSION` to "`SAGTRPC`". See also *Configuring ICU Conversion* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation. More information can be found under *Internationalization with EntireX*.			
Compression Level	The level of broker data compression.	`NO_COMPRESSION`	No	No
Timeout	The number of seconds to wait for a response from the RPC server.	60S	No	No
RPC Library Name	The name of the RPC library. Default: the library name used to generate the connection. Mainly used to override the library name for Natural RPC servers. (1)		No	Yes
RPC User	The RPC user ID sent to the RPC server. (3)		No	Yes
RPC Password	The password for the user above. (3)		No	Yes
Retype RPC Password	Verification of the password above.		Not applicable	Not applicable
Metadata Information	**Caution:** Do not change this generated property. It contains timestamp, IDL file name, library name, and version of the Designer.		Generated	No
Transaction Type (2)	Transaction types "none" (no transactions) and "local" (local transactions) are supported.	none	No	No

Notes:

1. Depending on the server configuration, some Natural RPC servers need the RPC library name. See the documentation of the Natural RPC Server at *https://empower.softwareag.com/Products/Documentation/default.asp*.

2. The transaction type “local” of the Integration Server is mapped to conversational RPC. See Appendix B in *webMethods Adapter Development Kit User's Guide Version 6.5* for transactions of the Integration Server. XA transactions are not supported by the EntireX Adapter.
3. The RPC user ID/password pair is designed to be used by the receiving RPC server. This component's configuration determines whether the pair is considered or not. Useful scenarios are: credentials for Natural Security and Impersonation in the respective RPC Server documentation.

Connection Parameters for RPC and Reliable RPC Listener Connections

Parameter	Description	Default	Can be set from the Designer
Broker ID	The ID of the broker you want to connect to. This ID consists of a host and an optional port. Default for the port is "1971".	localhost	Yes
Server Address	The address of the RPC server registered to the broker above. The address is given in the format `<class>/<server>/<service>`.	`RPC/SRV1/CALLNAT`	Yes
Logon User	The name of the user to log on to the broker.		Yes
Logon Password	The password for the user above.		Yes
Retype Logon Password	Verification of the password.		Not applicable
Encryption	Deprecated. For encrypted transport we strongly recommend using the Secure Sockets Layer/Transport Layer Security protocol. See *SSL/TLS, HTTP(S), and Certificates with EntireX* in the platform-independent Administration documentation.	No_Encryption	No
Encoding	The character encoding used for the RPC connection to the *EntireX Broker*. Default: the encoding of the Integration Server. Enable character conversion in the broker by setting the service-specific attribute `CONVERSION` to `"SAGTRPC"`. See also *Configuring ICU Conversion* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation. More information can be found under *Internationalization with EntireX*.		Yes
Compression Level	The level of broker data compression.	`NO_COMPRESSION`	No
Metadata Information	**Caution:** Do not change this generated property. It contains timestamp, IDL file name, library name, and version of the Designer.		Generated

Connection Parameters for Direct RPC Connections

Parameter	Description	Default	Can be set from the Designer	Can be set dynamically with service
Server Address	The address of the RPC server registered to the broker above. The address is given in the format <class>/<server>/<service>.	RPC/SRV1/CALLNAT	Yes	Yes
Timeout	The number of seconds to wait for a response from the RPC server.	60S	No	No
RPC Library Name	The name of the RPC library. Default: the library name used to generate the connection. Mainly used to override the library name for Natural RPC servers. (2)		No	Yes
RPC User	The RPC user ID sent to the RPC server. (3)		No	Yes
RPC Password	The password for the user above. (3)		No	Yes
Retype RPC Password	Verification of the password above.		Not applicable	Not applicable
Metadata Information	**Caution:** Do not change this generated property. It contains timestamp, IDL file name, library name, and version of the Designer.		Generated	No
Transaction Type (1)	Transaction types "none" (no transactions) and "local" (local transactions) are supported.	none	No	No

Notes:

1. The transaction type "local" of the EntireX Adapter connection is mapped to conversational RPC. See Appendix B in *webMethods Adapter Development Kit User's Guide Version 6.5* for transactions of the Integration Server. XA transactions are not supported by the EntireX Adapter.
2. Depending on the server configuration, some Natural RPC servers need the RPC library name. See the documentation of the Natural RPC Server at *https://empower.softwareag.com/Products/Documentation/default.asp*.
3. The RPC user ID/password pair is designed to be used by the receiving RPC server. This component's configuration determines whether the pair is considered or not. Useful scenarios are: credentials for Natural Security and Impersonation in the respective RPC Server documentation.

Connection Parameters for Reliable RPC Connections

Parameter	Description	Default	Can be set from the Designer	Can be set dynamically with service
Broker ID	The ID of the broker you want to connect to. This ID consists of a host and an optional port. Default for the port is "1971".	localhost	Yes	No
Server Address	The address of the RPC server registered to the broker above. The address is given in the format `<class>/<server>/<service>`.	Yes	Yes	
Logon User	The name of the user to log on to the broker.		Yes	Yes
Logon Password	The password for the user above.		Yes	Yes
Retype Logon Password	Verification of the password.		Not applicable	Not applicable
Encryption	Deprecated. For encrypted transport we strongly recommend using the Secure Sockets Layer/Transport Layer Security protocol. See *SSL/TLS, HTTP(S), and Certificates with EntireX* in the platform-independent Administration documentation.	No_Encryption	No	No
Encoding	Deprecated. For encrypted transport we strongly recommend using the Secure Sockets Layer/Transport Layer Security protocol. See *SSL/TLS, HTTP(S), and Certificates with EntireX* in the platform-independent Administration documentation.		Yes	No
Compression Level	The level of broker data compression.	NO_COMPRESSION	No	No
RPC Library Name	The name of the RPC library. Default: the library name used to generate the connection. Mainly used to override the library name for Natural RPC servers. (2)		No	Yes
RPC User	The RPC user ID sent to the RPC server. (3)		No	Yes
RPC Password	The password for the user above. (3)		No	Yes

Parameter	Description	Default	Can be set from the Designer	Can be set dynamically with service
Retype RPC Password	Verification of the password above.		Not applicable	Not applicable
Metadata Information	**Caution:** Do not change this generated property. It contains timestamp, IDL file name, library name, and version of the Designer.		Generated	No
Transaction Type (1)	Transaction types "none" (no transactions) and "local" (local transactions) are supported.	none	No	No

Notes:

1. The transaction type “local” of the Integration Server is mapped to Reliable RPC with client-commit for Reliable RPC connections. See Appendix B in *webMethods Adapter Development Kit User’s Guide Version 6.5* for transactions of the Integration Server. XA transactions are not supported by the EntireX Adapter.
2. Depending on the server configuration, some Natural RPC servers need the RPC library name. See the documentation of the Natural RPC Server at *https://empower.softwareag.com/Products/Documentation/default.asp*.
3. The RPC user ID/password pair is designed to be used by the receiving RPC server. This component's configuration determines whether the pair is considered or not. Useful scenarios are: credentials for Natural Security and Impersonation in the respective RPC Server documentation.

Connection Parameters for Direct RPC and Direct Reliable RPC Listener Connections

Parameter	Description	Default	Can be set from the Designer
Server Address	The address of the RPC server registered to the broker above. The address is given in the format `<class>/<server>/<service>`.	`RPC/SRV1/CALLNAT`	Yes
Metadata Information	**Caution:** Do not change this generated property. It contains timestamp, IDL file name, library name, and version of the Designer.		Generated

Connection Parameters for Connections to IMS Connect

Parameter	Description	Default	Can be set from Designer	Can be set dynamically with service
Host	Hostname of IMS Connect.		Yes	No
Port	IMS Connect port.		Yes	No
IMS Connect Data Store ID	Name of the data store, as defined in the IMS Connect configuration member.		Yes	Yes
Encoding	Specify the appropriate EBCDIC encoding used by your IMS Connect.	cp037	Yes	No
Socket Timeout (1)		10000	No	No
Check for DFS Errors		true	No	No
Logical Terminal ID			No	Yes
Use old Exit		true	No	No
Exit Name		Default name for old exit is "*SAMPLE*", for new exit "*SAMPL1*"	No	No
RACF User ID	The name of the user to log on to IMS Connect.		Yes	Yes
RACF Password / PassTicket	The password of the user above.		Yes	Yes
Retype RACF Password / PassTicket	Verification of the password above.		Not applicable	Not applicable
RACF Group Name	Security setting.		No	Yes
RACF Application Name	Defined to RACF on the PKTDATA definition.		No	Yes
SSL Parameters	Truststore and optional certificate. Example: `trust_store=CACerts.jks` See also *Support for SSL/TLS*.		No	No
Metadata Information	**Caution:** Do not change this generated property. It contains timestamp, IDL file name, library name, and version of the Designer.		Generated	No
Use IDL program as transaction name	Automatically use the IDL program name as transaction name. If set to "true" or "yes", 10 bytes are used for the transaction name. If set to a	false	No	No

Parameter	Description	Default	Can be set from Designer	Can be set dynamically with service
	number, this number of bytes is used for the transaction name.			

Notes:

1. The socket timeout value is used to set the IMS Connect timer value. The timer value specifies the delay that IMS Connect will wait until IMS returns data to IMS Connect, which in turn will be sent to the adapter. If the socket timeout is less than one minute, the IMS timer value is one second less than the socket timeout. If the socket timeout is less than one hour, the IMS timer value is one minute less than the socket timeout. The maximum IMS timer value is 60 minutes.

Connection Parameters for CICS ECI Connections

This connection type uses the External Call Interface (ECI) within CICS. For more information see *Preparing IBM CICS for ECI*.

Parameter	Description	Default	Can be set from Designer	Can be set dynamically with service
Host	Hostname of CICS.		Yes	No
Port	CICS port.		Yes	No
CICS Mirror Transaction ID	Name of the CICS mirror transaction. Default is "CPMI", which is the default dispatching transaction for ECI.	CPMI	Yes	Yes
Encoding	Specify the appropriate EBCDIC encoding used by your CICS ECI.	cp037	Yes	No
Socket Timeout	Socket timeout (in milliseconds).	10000	No	No
RACF User ID	The name of the RACF user to log on to CICS ECI.		Yes	Yes
RACF Password / PassTicket	The password for the user above. See **Note**.		Yes	Yes
Retype RACF Password / PassTicket	Verification of the password above.		No	No
SSL Parameters	Truststore and optional certificate. Example: `trust_store=CACerts.jks` See also *Support for SSL/TLS*.		No	No
Metadata Information	**Caution:** Do not change this generated property. It contains timestamp, IDL file name, library name, and version of the Designer.		Generated	No

Note: By default the password is translated to uppercase. To use mixed-case passwords, set `watt.com.softwareag.entirex.wmadapter.cics.eci.mixedcase.password=true` under **Settings** > **Extended** on the Integration Server administration page.

Connection Parameters for CICS Socket Listener Connections

This connection type uses the CICS Socket Listener. For more information see *Preparing for CICS Socket Listener*.

Parameter	Description	Default	Can be set from Designer	Can be set dynamically with service
Host	Hostname of CICS Socket Listener.		Yes	No
Port	CICS Socket Listener Port.		Yes	No
Encoding	Specify the appropriate EBCDIC encoding used by your CICS Socket Listener.	cp037	Yes	No
CICS Transaction ID	Name of the CICS transaction. Default is "XRFE", which is the default dispatching transaction for CICS Socket Listener.	XRFE	Yes	No
Socket Timeout	Socket timeout (in seconds).	10	No	No
RACF User ID	The name of the user to log on to CICS Socket Listener.		Yes	Yes
RACF Password	The password for the user above.		Yes	Yes
Retype RACF Password	Verification of the password above.		No	No
SSL Parameters	Truststore and optional certificate. Example: `trust_store=CACerts.jks` See also *Support for SSL/TLS*.		No	No
Application Name (1)	Application name used to generate the PassTicket.		No	No
Secured Signon Key (1)	Secured signon key used to generate the PassTicket.		No	No
Metadata Information	**Caution:** Do not change this generated property. It contains timestamp, IDL file name, library name, and version of the Designer.		Generated	No
Transaction Type (2)	Transaction types "none" (no transactions) and "local" (local transactions) are supported.	None	No	No
Sync on Return (3)	Optional. If `true`, the CICS program is called with the `CICS LINK` option `SYNCONRETURN`. Refer to the **IBM CICS Transaction Server for z/OS documentation** for more information on the `SYNCONRETURN` option. This option is only useful if the CICS program is defined as DPL. See note.	false	No	No
User Exit	Optional. Class name of user exit implementation. See *User Exit*.		No	No

Parameter	Description	Default	Can be set from Designer	Can be set dynamically with service
User Exit Class Path	Optional. URL of the classpath for user exit implementation, for example `file://myexit.jar` or `http://host/path/to/my/exit`.		No	No

Notes:

1. PassTicket is supported only when the CICS Socket Listener (remote connector) on z/OS is used. See ***Preparing for CICS Socket Listener*** and *Installing CICS Socket Listener* in the z/OS Installation documentation.
2. The transaction type "local" of the EntireX Adapter connection is mapped to conversational RPC. See Appendix B in *webMethods Adapter Development Kit User's Guide Version 6.5* for transactions of the Integration Server. XA transactions are not supported by the EntireX Adapter.
3. Setting this option to `true` with DPL impacts the syncpoint handling of the EntireX CICS Socket Listener. Usually the syncpoint is performed after a reply has been sent to the request. Using this option, the syncpoint is now performed after program execution. This means that if the reply fails, the syncpoint has already been performed. Using conversational requests (multiple requests with one syncpoint after the last request) the `SyncOnReturn` option is just ignored without further notice.

Connection Parameters for ACI Server Connections

Parameter	Description	Default	Can be set from the Designer
Broker ID	The ID of the broker you want to connect to. This ID consists of a host and an optional port. Default for the port is "1971".	localhost	Yes
Server Address	The address of the RPC server registered to the broker above. The address is given in the format `<class>/<server>/<service>`. The address may contain an asterix ('*') as a wildcard that is substituted by the IDL program name at runtime. This helps you to use only one connection for multiple IDL programs or adapter services.	RPC/SRV1/CALLNAT	Yes
Logon User	The name of the user to logon on to the broker.		Yes
Logon Password	The password for the user above.		Yes
Retype Logon Password	Verification of the password above.		Not applicable

Parameter	Description	Default	Can be set from the Designer
Encryption	Deprecated. For encrypted transport we strongly recommend using the Secure Sockets Layer/Transport Layer Security protocol. See *SSL/TLS, HTTP(S), and Certificates with EntireX* in the platform-independent Administration documentation.	No_Encryption	No
Encoding	The character encoding used for the ACI connection to the EntireX Broker. Default: the encoding of the Integration Server. Enable character conversion in the broker by setting the service-specific attribute `CONVERSION` to "`SAGTCHA`". See also *Configuring ICU Conversion* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation. More information can be found under *Internationalization with EntireX*.		No
Timeout	The number of seconds to wait for a response from the RPC server.	60S	No
Server Type	Defines how arrays of groups are marshalled. Allowed values are "COBOL" and "Natural". **Note:** See *Writing ACI Servers for the RPC-ACI Bridge in COBOL | Natural* for restrictions concerning ACI servers.	COBOL	No
Metadata Information	**Caution:** Do not change this generated property. It contains timestamp, IDL file name, library name, and version of the Designer.		Generated

Connection Parameters for ApplinX Connections

Parameter	Description	Default	Can be set from Designer
Host	Hostname of ApplinX server.		Yes
Port	Port of ApplinX server.		Yes
Secure connection	Set to "true" for a secure connection.	false	No
Metadata Information	**Caution:** Do not change this generated property. It contains timestamp, IDL file name, library name, and version of the Designer.		Generated

Connection Parameters for COBOL Converter Connections

Parameter	Description	Default	Can be set from Designer	Can be set dynamically with service
Encoding	The character encoding of the COBOL binary data.	Cp037	Yes	No
Metadata Information	**Caution:** Do not change this generated property. It contains timestamp, IDL file name, library name, and version of the Designer.		Generated	No

Connection Parameters for AS/400 Connections

Note: After installation of the EntireX Adapter, the AS/400 connection type is not visible and not usable. To access the AS/400 system you need the IBM ToolBox for Java (JTOpen). See *Post-installation Steps for AS/400*.

Parameter	Description	Default	Can be set from Designer	Can be set dynamically with service
Host	Hostname of the AS/400 (IBM i) system.		Yes	No
User ID	The user profile name to use to authenticate to the system.		Yes	No
Password	The user profile password to use to authenticate to the system.		Yes	No
Retype Password	Verification of the password above.			Not applicable
Encoding	The character encoding used for the RPC connection to the *EntireX Broker*. Default: the encoding of the Integration Server. Enable character conversion in the broker by setting the service-specific	cp037	Yes	No

Parameter	Description	Default	Can be set from Designer	Can be set dynamically with service
	attribute `CONVERSION` to `"SAGTRPC"`. See also *Configuring ICU Conversion* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation. More information can be found under *Internationalization with EntireX*.			
Timeout	Maximum time to run the program in seconds.	60		No
Program Path	The fully qualified integrated file system path name to the program. `%library%` is replaced by the IDL file library name and `%program%` is replaced by the IDL file program name. The library and program name must each be 10 characters or less.	/QSYS.LIB/%library%.LIB/%program%.PGM		No
Metadata Information	**Caution:** Do not change this generated property. It contains timestamp, IDL file name, library name, and version of the Designer.		Generated	Generated

Common Connection Management Properties (for all Connection Types)

Parameter	Description
Enable Connection Pooling	Enables the adapter to use connection pooling. Default: `true`. See *Connection Pooling* for more information on connection pooling.
Minimum Pool Size	If connection pooling is enabled, this field specifies the minimum number of connection objects that remain in the connection pool at all times. When the adapter creates the pool, it creates this number of connections. Default: `1`.
Maximum Pool Size	The maximum number of connection objects that can exist in the connection pool. The adapter will reuse any inactive connections in the pool or, if all connections are active and the connection pool has reached its maximum size, the adapter will wait for a connection to become available. Default: `10`.
Pool Increment Size	If connection pooling is enabled, this field specifies the number of connections by which the pool will be incremented if connections are needed, up to the maximum pool size. Default: `1`.
Block Timeout	If connection pooling is enabled, this field specifies the number of milliseconds that the Integration Server will wait to obtain a connection before it times out and returns an error. Default: `1000`.
Expire Timeout	If connection pooling is enabled, this field specifies the number of milliseconds that an inactive connection can remain in the pool before it is closed and removed from the pool. For example, to specify 10 seconds, specify 10000. Enter 0 to specify no timeout. Default: `1000`. **Note:** The adapter will never violate the `Minimum Pool Size` parameter. These connections remain in the pool regardless of how long they are inactive.
Startup Retry Count	The number of times that the system should attempt to initialize the connection pool at startup if the initial attempt fails. Default: `0`.
Startup Backoff Timeout	The number of seconds that the system should wait between attempts to initialize the connection pool.

Deleting Adapter Connections

If you no longer want to use a particular EntireX Adapter connection, you can delete it by following the instructions in this section. You delete adapter connections using the IS Administration Console. If you delete an EntireX Adapter connection, the adapter services that are defined to use the connection will no longer work. You can change the connection an adapter service uses. Therefore, if you delete an EntireX Adapter connection, you can assign a different connection to an adapter service and reuse the service.

To delete a connection

1 In the **Adapters** menu in the IS Administration Console navigation area, click **EntireX Adapter**.

2 Make sure that the connection is disabled before deleting. To disable the connection, click **Yes** in the Enabled column and click **OK** to confirm. The Enabled column now shows "No" (Disabled) for the connection.

3 On the Connections screen, click the **Delete** icon for the connection you want to delete.

As a result, the Integration Server deletes the adapter connection.

Enabling Adapter Connections

An EntireX Adapter connection must be enabled before you can configure any adapter service using the connection, or before an adapter service can use the connection at runtime. You enable adapter connections using the IS Administration Console.

Note: When you reload a package that contains enabled connections, the connections will automatically be enabled when the package reloads. If the package contains connections that are disabled, they will remain disabled when the package reloads.

To enable a connection

1 In the **Adapters** menu in the IS Administration Console navigation area, click **EntireX Adapter**.

2 On the **Connections** screen, click **No** in the **Enabled** column for the connection you want to enable.

As a result, the IS Administration Console enables the adapter connection and displays a tick (✓) and "Yes" in the **Enabled** column.

Note: The EntireX broker configured in the Connections does not need to be running to enable a connection. Only the parameters are checked for correctness while enabling a connection.

Disabling Adapter Connections

EntireX Adapter connections must be disabled before you can edit or delete them. You disable adapter connections using the IS Administration Console.

To disable a connection

1 In the **Adapters** menu in the IS Administration Console navigation area, click **EntireX Adapter**.

2 On the **Connections** screen, click **Yes** in the **Enabled** column for the connection you want to disable.

As a result, the adapter connection becomes disabled and you see a "No" in the **Enabled** column.

Connection Pooling

The Integration Server includes a connection management service that dynamically manages connections and connection pools based on configuration settings that you specify for the connection. All adapter services use connection pooling. A connection pool is a collection of connections with the same set of attributes. The Integration Server maintains connection pools in memory. Connection pools improve performance by enabling adapter services to reuse open connections rather than opening new connections.

Runtime Behavior of Connection Pools

When you enable a connection, the Integration Server initializes the connection pool, creating the number of connection instances you specified in the connection's `Minimum Pool Size` field. Whenever an adapter service needs a connection, the Integration Server provides a connection from the pool. If no connection is available in the pool, and the `Maximum Pool Size` has not been reached, the server creates one or more new connections (according to the number specified in `Pool Increment Size`) and adds them to the connection pool. If the pool is full (as specified in `Maximum Pool Size`), the requesting service will wait for the Integration Server to obtain a connection, up to the length of time specified in the `Block Timeout` field, until a connection becomes available. Periodically, the Integration Server inspects the pool and removes inactive connections that have exceeded the expiration period specified in `Expire Timeout`. You can enable the system to retry the initialization any number of times, at specified intervals.

5 EntireX Adapter Services

This chapter describes how to configure and manage EntireX Adapter services. EntireX Adapter services allow clients to interact with EntireX RPC servers, Natural RPC servers, ACI servers, ApplinX, IMS Connect, CICS ECI and AS/400 using a configured adapter connection. EntireX Adapter services are available for synchronous and asynchronous RPC calls (RPC and reliable RPC).

Before Configuring or Managing Adapter Services

To prepare configuration and management of EntireX Adapter services

1 Start your Integration Server and the IS Administration Console, if they are not already running.

2 Make sure you have webMethods administrator privileges so that you can access the EntireX Adapter's administrative screens. See the *Administering webMethods Integration Server* for information on setting user privileges.

3 Using the IS Administration Console, make sure the WmEntireX package is enabled. See ***Enabling and Disabling Packages*** for instructions.

4 Start the Designer if it is not already running.

5 Make sure you are viewing the Designer in the **Service Development** perspective, as described in *Switching Perspectives* in the Designer help.

6 Using the Designer, create a user-defined package to contain the service, if you have not already done so. Add a dependency on WmEntireX package (version *.*) to your user-defined package. When you configure adapter services, you should always define them in user-defined packages rather than in the WmEntireX package. See ***Package Management***.

7 Generate the adapter connection with the Designer.

   Or:

   Using the IS Administration Console, configure an adapter connection to use with the adapter service. See ***EntireX Adapter Connections*** for instructions.

Configuring Adapter Services

The EntireX Adapter service enables you to send RPC calls to an EntireX RPC server or Natural RPC server. You configure EntireX Adapter services using a specific adapter connection. After you follow the steps in this section to configure the adapter service, you can invoke it from a flow or Java service. To use EntireX Adapter services, you provide values for the service's input signature in the pipeline. See ***Using Adapter Services***.

To create EntireX Adapter services

1 In the Designer, select the Software AG IDL file to use.

2 In the context menu, choose **Generate webMethods IS Connection from Software AG IDL...**.

Using Adapter Services

To enable the EntireX Adapter service to send an RPC call to an EntireX RPC server

- Provide values for the parameters in the service's input signature when configuring the adapter service within a flow service in the Designer. You can either map in values from the pipeline, or you can set constant values using the **Pipeline View**.

  This has the following effect on the input and output signature:

  - The input signature of an adapter service wraps all input parameters of the EntireX subprogram in the `inRec`.

    A parameter `throwException` is added to the signature. If this optional parameter is set to "true", errors are returned as exceptions and not by means of parameters in the output signature.

    A parameter `$connectionname` is added to the signature. If this optional parameter is set to a connection name, it overrides the default connection name of the service.

    Depending on the connection type, adapter services have optional parameters that can be used to overwrite the corresponding values of the connection. These optional parameters are:

    - **EntireX RPC Connection / EntireX Reliable RPC Connection**
      - Logon user
      - Logon password
      - RPC library name
      - RPC user
      - RPC password
      - Server address

      See *Connection Parameters for RPC Connections* | *Connection Parameters for Reliable RPC Connections*.

- **EntireX Direct RPC Connection**
  - RPC library name
  - RPC user
  - RPC password
  - Server address

  See *Connection Parameters for Direct RPC Connections*.
- **IMS Connect Connection**
  - Logical terminal ID
  - RACF user ID
  - RACF password/PassTicket
  - RACF group name
  - RACF application name
  - IMS data store ID
  - Maximum segment length

  See *Connection Parameters for Connections to IMS Connect*.
- **CICS ECI Connection**
  - RACF user ID
  - RACF password/PassTicket. See **Note**
  - CICS Mirror Transaction ID

  See *Connection Parameters for CICS ECI Connections*.
- **CICS Socket Listener Connection**
  - RACF user ID
  - RACF password
  - User transaction ID

  See *Connection Parameters for CICS Socket Listener Connections*.
- **EntireX ACI Connection**
  - Logon user
  - Logon password

  See *Connection Parameters for ACI Server Connections*.

- **EntireX COBOL Converter Connection**
  - cobolInput
  - cobolOutput

  See *Connection Parameters for COBOL Converter Connections* and *Converting IS Data Structures with the COBOL Converter* for details on the parameters.
- **AS/400 Connection**
  - User ID
  - Password
  - Program Path

  See *Connection Parameters for AS/400 Connections*.

- The output signature of an adapter service wraps all output parameter of the EntireX subprogram in the `outRec`. Three parameters `errorFlag`, `errorMessage` (optional), and `errorCode` are added to the signature. If `errorFlag` is true, `errorMessage` contains the error message. `errorCode` contains "00000000" for a successful call, otherwise the EntireX message class (4 digits) and message code (4 digits). Depending on the connection type, the parameters `messageID` and `correlationID` are added; they contain the message ID of the request and the correlation ID of the reply (if available).

Viewing Adapter Services

Use the Designer to view adapter services. Make sure you are viewing the Designer in the **Service Development** perspective, as described in *Switching Perspectives* in the Designer online help.

To view a service

1 In the Designer **Package Navigator** view, expand the package and folder that contain the service you want to view.

2 Select the service you want to view. The Designer displays the service in the **Service Editor**.

As a result, the Adapter Service Editor for the EntireX Adapter service appears.

Notes:

1. You can select the **Adapter Settings** tab at any time to confirm adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.
2. On the **RPC Call** tab, the EntireX Subprogram Name field holds the name of the IDL program called by this service. The tables **Input Parameter** and **Output Parameter** display names, the IDL types, and the parameter types in the pipeline.
3. The **Input/Output** tab shows the service's input and output signatures. To use the EntireX Adapter service, you provide values for the signature input within the service pipeline in the Designer. See ***Using Adapter Services*** for instructions on using the EntireX Adapter service.
4. For information on configuring the **Input/Output** and **Adapter Settings** tabs, see *webMethods Service Development Help*. These tabs apply to all services that you configure using the Designer. The information on **Audit** and **Permissions** appears in the **Properties** panel. For the results, see the **Results** panel.

Deleting Adapter Services

Use the Designer to delete adapter services. Make sure you are viewing the Designer in the **Service Development** perspective, as described in *Switching Perspectives* in the Designer online help.

To delete a service

1 In the Designer **Package Navigator** view, expand the package and folder that contain the service you want to delete.

2 Right-click the service and then click **Delete**.

Runtime Connection Allocation for Adapter Services

Adapter services of the EntireX Adapter are enabled for dynamically selecting a connection node. The connection name may be specified on the pipeline in the `$connectionName` field. This field has to be part of the services input signature. If `$connectionName` is not specified, the default connection name is used, usually the connection used to create the service.

See chapter 8 in *webMethods Adapter Development Kit User's Guide* , version 6.5 for more details on connection selection.

Connection selection is applicable to services that are generated with the EntireX Adapter (version 8.0 or higher) and helps to:

- **Switch between development and test environment**
  - Generate the connections and services as usual.
  - Design your services that call the adapter services to set `$connectionName` to one single connection.
  - Then you can switch to a different environment by changing this single connection to a different resource.
- **Reduce the number of connection pools**
  - Generate the connections and services as usual.
  - Then for each IDL library one connection is generated and one connection pool is used at runtime.
  - Design your services that call the adapter services to set `$connectionName` to one single connection.
  - This reduces the connection pools to a single pool associated with this connection.

Configure the Formatting of Decimal and Alphanumeric Parameters

Adapter services have two parameters to configure the formatting of decimal parameters (IDL data types `N`, `NU`, `P` and `PU`, and alphanumeric parameters of fixed length (IDL data types `A`, `K`, and `U`).

To set these options, use the Designer and open the adapter service for the EntireX Adapter. In **RPC Call**, select **Parameters for Decimal Numbers (N/P)** or **Parameters for fixed length Strings (A/K/U)**.

The following tables give an overview of parameters to be configured:

- **Parameters for Decimal Numbers (N/P)**
  (As an example, we use the value "3.14 " for a parameter defined as `N7.2` in the IDL.)

Parameter	Example
Remove leading zeros & keep decimal point (default)	3.14
Remove leading zeros & remove decimal point	314
Keep leading zeros & keep decimal point	0000003.14
Keep leading zeros & remove decimal point	000000314

- **Justify Decimal Numbers (N/P)**

  Note: This parameter is highly deprecated and should be only used for compatibility with webMethods Mainframe.

Parameter	Explanation
not justified (default)	The field formatting is unchanged.
left justified	The numeric field is left justified.
right justified	The numeric field is right justified.

- **Parameters for fixed length Strings (A/K/U)**

Parameter	Explanation
Remove whitespace characters (default)	Trim leading and trailing whitespace characters.
Keep whitespace characters	Format the string with the defined number of characters.

- **Justify fixed length Strings (A/K/U)**

  Note: This parameter is highly deprecated and should be only used for compatibility with webMethods Mainframe.

Parameter	Explanation
not justified (default)	The field formatting is unchanged.
left justified	The string field is left justified.
right justified	The string field is right justified.

Configuring Null Value Suppression

Null value suppression (NVS) allows you to suppress parameters with no specified value. Suppressed parameters will not appear in the result of the adapter service call. Null value suppression is available for the following IDL data types:

- all string data types
- groups and structures
- arrays

To activate NVS in the context of adapter services, modify each adapter service separately, using Designer. For each adapter service, select **Null Value Suppression** in **RPC Call**. You can fine-tune NVS with the following suppression modes:

Suppression Mode	Explanation
No suppression (default)	No suppression, all parameters are present in the output structure.
Suppress null elements	String parameters with empty strings are suppressed, empty structures and arrays are suppressed.
Suppress null elements and trim array cells at end	Same as above. In addition, all empty array elements at the end of the array will be removed from the array.
Suppress null elements and trim array cells at end containing null elements and zero numbers	Same as above. In addition, numbers of numeric/decimal type (I1, I2, I4, N, P) with the value zero are treated as an empty value.

A string parameter is suppressed if the assigned string is:

- of type variable length (AV/KV/UV) and the string is empty (same for variable length strings with a maximum length).
- of type fixed length (A/K/V) and the parameter **Remove whitespace characters** is enabled, and trimming the leading and the trailing whitespace characters would result in an empty string.

6 Listeners

Introduction

Listeners are used for inbound connections to the Integration Server. They are needed for the following connection types:

- RPC Listener connections
- Reliable RPC Listener connections
- Direct RPC Listener connections
- Direct Reliable RPC Listener connections

Use the Integration Server Wrapper to create:

- RPC Listener and Direct RPC Listener connections with listeners
- Reliable and Direct Reliable RPC Listener connections with listeners, notifications, triggers and document types

Creating and Updating Listeners

Use the Integration Server Wrapper to create or update listeners. See *Step 3: Create or Update an Adapter Connection* and *Step 4c: Create a Connection and Related Adapter Listener* in the Integration Server Wrapper documentation.

Depending on the connection type, the following objects will be generated:

- **RPC Listener and Direct RPC Listener connections**
  - The connection and the listener.
  - For all services specified in the "Service Names" field that do not exist, an empty flow service with the corresponding signature is created.
- **Reliable RPC Listener and Direct Reliable RPC Listener connections**
  - The connection and the listener.
  - A listener notification, a messaging trigger and a notification publish document.
  - For all services specified in the "Service Names" field that do not exist, an empty flow service with the corresponding signature is created
  - For all services specified in the "Service Names" field that do exist, and if the input signature of the service does not include the notification publish document, a wrapper flow service is generated which implements the signature mapping.

Configuring Listeners

You change the configuration of EntireX Adapter Listeners using the IS Administration Console.

To configure a listener

1 In the **Adapters** menu in the IS Administration Console's navigation area, click **EntireX Adapter**.

2 In the **EntireX Adapter** menu, click **Listeners**.

3 The **Listeners** screen shows all existing listeners. Only disabled listeners can be configured. If a listener is enabled, select **Disabled** in the **State** column. Then click **Edit**.

4 On the **Listener** screen, in the EntireX Adapter section, the following fields can be changed:

Field	Description
Connection Name	This setting should not be changed. Otherwise the listener may no longer work.
Retry Limit	The number of times the adapter tries to reconnect if the adapter fails to connect, or loses connection with the EntireX Broker or Direct RPC. Default: `10`.
Retry Backoff Timeout	The number of seconds that elapse between each of the retries specified in the retry limit. Default: `20`.
EntireX Subprogram Names	A list of subprogram names, separated by blanks. This list and the following list of service names is generated by the Designer.
Service Names	A list of Integration Server services, separated by blanks. Each service corresponds to a subprogram name. For the subprogram the service at the same position in the list is called. The lists must have the same number of names.
Minimum Listener Threads	The number of worker threads for this listener that should be started when the listener is enabled. If set to `0`, worker threads are started on demand. Default: `1`.
Maximum Listener Threads	The maximum number of listener threads to start. Default: `1`. If the listener is using one of the reliable listener connections, the value cannot be greater than 1.
Entry Handler Service	The name of an Integration Server service that will be called in addition before the listener calls a service listed in **Service Names**. The specification `pub.wmentirex.listener:entryExitHandlerSpecification` describes the interface of the handler service.
Exit Handler Service	The name of an Integration Serverservice that will be called in addition after the listener has called a service listed in **Service Names**. The specification `pub.wmentirex.listener:entryExitHandlerSpecification` describes

Field	Description
	the interface of the handler service. If the listener is using one of the reliable listener connections, this field cannot be specified.
Execute Service with Client Credentials	If enabled, the RPC client has to provide RPC user ID and RPC password. These credentials are then used to execute the Integration Server service. ■ For how to send the RPC user ID/password pair from an RPC client, see *Using the Broker and RPC User ID/Password* (.NET Wrapper | Java Wrapper | C Wrapper | PL/I Wrapper | DCOM Wrapper | Web Services Wrapper | IDL Tester | Listener for XML/SOAP | Listener for IBM MQ). ■ For the COBOL Wrapper, refer to *Using Broker Logon and Logoff* and *Using RPC Authentication (Natural Security, Impersonation, Integration Server)*. ■ For non-RPC clients, see *Using the Broker and RPC User ID/Password* under *EntireX XML Tester* in the XML/SOAP Wrapper documentation. Default: `false`.
Parameters for Fixed-length Strings	See ***Configure the Formatting of Decimal and Alphanumeric Parameters***.
Null Value Suppression	See ***Configuring Null Value Suppression***.

5 Save the listener.

6 Enable the listener if needed.

7 Settings and Information

This chapter describes information and tracing screens in the IS Administration Console.

Adapter Settings

Changing Settings

EntireX Tracing	
Level of EntireX Tracing	1 (trace calls)
Tracefile Location (Folder)	C:\SoftwareAG910\IntegrationServer\instances\default\logs

Date and Time Patterns	
Date Pattern	MMM d, yyyy
Time Pattern	MMM d, yyyy h:mm:ss a
Locale for Date/Time Pattern	en

The first part of this screen shows the current level of EntireX Tracing.

The second part shows the setting for the Date and Time format patterns. You can change these values with the **Change Settings** link. This has immediate effect. For the Date and Time format patterns, the syntax of the Java class `java.text.SimpleDateFormat` is used. Localized date and time pattern strings are also supported. In these strings, the pattern letters described above may be replaced with other, locale-dependent, pattern letters using the specified locale. The default value appears when a field is left empty.

Further Setting Options for Tracing

The adapter uses the logging facility of the Integration Server. The facility code is "0800" for the EntireX Adapter. The logging level specifies the amount of data logged. These log entries are written to the server log file.

Connections Information

This screen displays the following sections about connections and listeners.

- **Broker Information**
  The status of all brokers used in the connections. A green dot shows that a connection to the broker is successful. As status, the version and the operating system platform is displayed. A red dot shows that a connection to the broker is not possible. As status, the error message is displayed.
- **Connections for Services**
  The status of the external servers such as RPC Server, IMS Connect and CICS ECI used by the connections for adapter services. A green dot shows that connection to the RPC server is successful and the status information from the server is displayed. A red dot shows that a connection to the RPC server or to the Broker is not possible. As status, the error message is displayed. The current number of connection instances is shown in column **Count**.
- **Connections for Listeners**
  For each Listener the corresponding name of the connection, Broker ID, and server address is displayed.

Built-in Services for Connections Information

The information on connections and listeners shown on the IS Administration Console's page is available with built-in services. The following services are available in the WmEntireX package. Parameters are either input or output (I/O).

Service	Parameter	I/O	Description
`pub.wmentirex.connectioninfo:brokerInfo` The service retrieves the status of the broker given in the broker ID. Use `pub.wmentirex.connectioninfo:brokerList` before to get the list of broker IDs used in all connections.	`brokerID String`	I	ID of the broker to be pinged.
	`refreshData String`	I	The number of milliseconds to refresh the data. If `refreshData` is null or not a number, the default of 10 seconds is used.
	`errorFlag String`	O	"true" if an error occurred, "false" if the call succeeded.
	`message String`	O	The result of the `ping` call to the broker.
`pub.wmentirex.connectioninfo:brokerList`	`refreshData String`	I	The number of milliseconds to refresh the data. If `refreshData` is null or not a

Service	Parameter	I/O	Description
The service retrieves the list of all broker IDs configured in the connections.			number, the default of 10 seconds is used.
	brokerID String[]	O	The array of broker IDs.
pub.wmentirex.connectioninfo:listenerList The service retrieves name, broker ID and server address for each listener.	refreshData String	I	The number of milliseconds to refresh the data. If refreshData is null or not a number, the default of 10 seconds is used.
	listeners IData[]	O	The array of IData objects containing three strings name, brokerID, serverAddress for each listener.
pub.wmentirex.connectioninfo:serviceInfo The service retrieves the result of the ping call for the connection.	connection String	I	The name of the connection to ping, e.g. "folder:name".
	refreshData String	I	The number of milliseconds to refresh the data. If refreshData is null or not a number, the default of 10 seconds is used.
	errorFlag String	O	"true" if an error occurred, "false" if the call succeeded.
	message String	O	The result of the ping call to the connection.
pub.wmentirex.connectioninfo:serviceList The service retrieves the connection names. One of these names can be used as input for pub.wmentirex.connectioninfo:serviceInfo	refreshData String	I	The number of milliseconds to refresh the data. If refreshData is null or not a number, the default of 10 seconds is used.
	connections String[]	O	The array of connection names in the format folder:connection.

Support Information

Connections, adapter services and adapter listeners contain metadata that is generated during deployment of connections and services. Use the service `pub.wmentirex.supportinfo:createInfo` to extract this metadata for support purposes and error diagnosis.

Service	Parameter	I/O	Description
`pub.wmentirex.supportinfo:createInfo` The service extracts the generated metadata from all connections and adapter services in a package and writes this to a file in the folder *<IntegrationServer_instance>/packages/WmEntireX/resources*.	`packageName` `String`	I	The name of a package. The generated metadata for all connections, adapter services and adapter listeners in this package is written to a file.
	`result` `String`	O	File <*`filename`*> created on host <*`hostname`*>.

Services Information

This screen shows statistics about the adapter services of the EntireX Adapter. Only services that have been executed at least once during the runtime of the Integration Server are shown.

The following information is shown for each service:

- name of the service
- corresponding program name
- number of successful calls
- number of erroneous calls
- average processing time in milliseconds (only successful calls)
- time of last access
- processing time in milliseconds for the last call
- status of last call (detailed error message if the last call failed)

Note: All these items except the service name can be reset with the link "Reset Statistics".

Listeners Information

This screen shows statistics about the listeners of the EntireX Adapter. For each listener, all services that have been invoked at least once during the runtime of the Integration Server are shown.

The following information is shown for each service invoked by a listener:

- name of the listener
- subprogram name and service name separated by '/'
- number of successful calls
- number of erroneous calls
- average processing time in milliseconds (only successful calls)
- time of last access
- processing time in milliseconds for the last call
- status of last call (detailed error message if the last call failed)

Note: All these items except the listener and subprogram/service name can be reset with the link **Reset Statistics**.

License Information

EntireX License Information

The first part of this screen shows information on the EntireX license used by the EntireX Adapter.

- Valid until: values are a date specification or "Unlimited".
- License for IMS Connect connections: "Yes" or "No".
- License for CICS ECI connections: "Yes" or "No"
- License for CICS Socket Listener connections: "Yes" or "No"
- License for Direct RPC connections: "Yes" or "No".
- License for COBOL Converter connections: "Yes" or "No".
- License for AS/400 connections: "Yes" or "No".
- License for EntireX Broker connections: "Yes" or a message that indicates a possible license coverage mismatch.

- Error message (only if the license is not granted).

Notes:

1. This license file for the EntireX Adapter is always named "license.entirex.xml" and resides in directory *IntegrationServer/instances/<instance_name>/config*.
2. If this file does not exist and an EntireX license file is available in the same suite installation the EntireX Adapter will copy the EntireX license file during startup to the location mentioned in [(1)].
3. You can specify the location of a license file (file name with folder) with the **Change Settings** link. The EntireX Adapter will copy this license file to the location mentioned in [(1)].
4. If a license file is copied to the location mentioned in [(1)], it is renamed to "license.entirex.xml" if necessary.

Adabas Replication License Information

The second part of this screen shows information on the Adabas Replication license used by the EntireX Adapter.

- Valid until: values are a date specification or "Unlimited".
- License for Adabas Replication connections: "Yes" or "No".
- Error message (only if the license is not granted).

Notes:

1. This license file for the EntireX Adapter is always named "license.reptor_entirex.xml" and resides in directory *IntegrationServer/instances/<instance_name>/config*.
2. You can specify the location of a license file (file name with folder) with the **Change Settings** link. The EntireX Adapter will copy this license file to the location mentioned in [(1)].
3. If a license file is copied to the location mentioned in [(1)], it is renamed to "license.reptor_entirex.xml" if necessary.

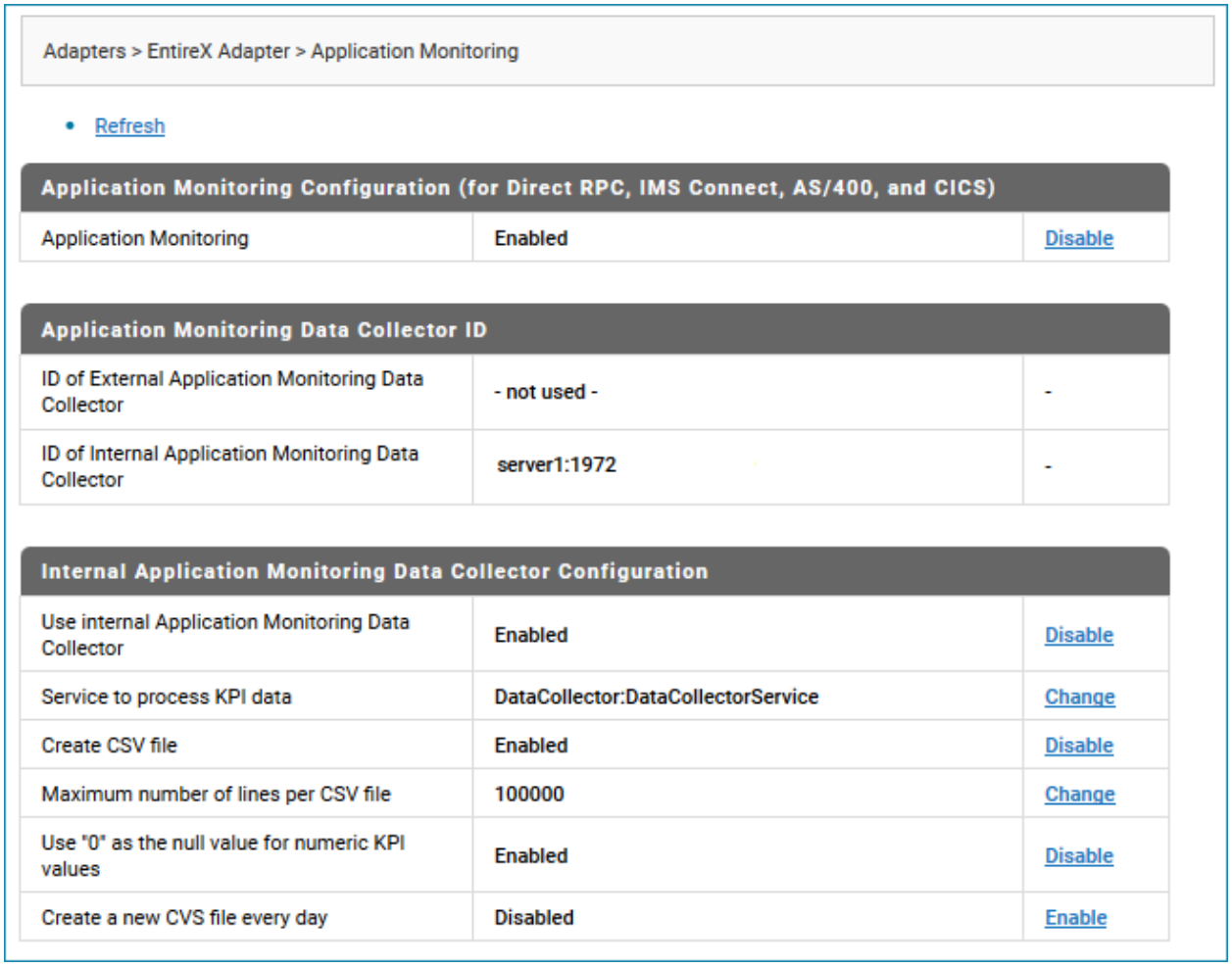
Adabas Replication Wizards

The menu items **Adabas Replication Wizard** and **Create Document Type from Adabas File** are used by the product "Adabas Replication Service for webMethods Integration Server". For details see the documentation of this product.

8 Application Monitoring

Application Monitoring is an EntireX feature that enables you to monitor the response times in your distributed applications, and it also enables you to monitor certain error situations. See the separate Application Monitoring documentation. To configure it, from the administration menu of the EntireX Adapter choose **Application Monitoring**.

You can either use the internal Data Collector of the Adapter (parameter `Use internal Application Monitoring Data Collector` is enabled), or an external Data Collector running outside of the Integration Server (parameter `Use internal...` is disabled).



Parameter	Description	Default	Note
Application Monitoring Configuration			
`Application Monitoring`	Enable or disable Application Monitoring for the EntireX Adapter. Applies to the following connection types: ■ EntireX Direct RPC Connection ■ EntireX Direct RPC Listener Connection ■ IMS Connect Connection	disabled	For connection types EntireX RPC Connection and EntireX RPC Listener Connection, configure the

Parameter	Description	Default	Note
	■ AS/400 Connection ■ CICS ECI Connection ■ CICS Socket Listener Connection		used broker instance. See *Setting up EntireX Broker* in the Application Monitoring documentation.
Application Monitoring Data Collector ID			
`ID of External Application Monitoring Data Collector`	Required if using an External Collector. Must be in format *`host-name:port-number`*, where *`host-name`* is the host on which the Application Monitoring Data Collector is running, and *`port-number`* is the port number of the Data Collector.		
`ID of Internal Application Monitoring Data Collector`	You cannot change the address of the EntireX Adapter's internal Data Collector. It is always the hostname of the machine where the Integration Server is running and the TCP/IP port number of the Direct RPC component.		
Internal Application Monitoring Data Collector Configuration			
`Use Internal Application Monitoring Data Collector`	Enable this parameter to use the internal Data Collector. This collector runs as a server in the Direct RPC component and is started and stopped automatically when the Direct RPC component is started or stopped.	disabled	
`Service to process KPI data`	Enter the full name of the service to consume the KPIs. This service is called with the available KPI values when the Data Collector receives a monitoring event. To define the signature of such a service, use a reference to the specification `pub.wmentirex.appmon:ApplicationMonitoringKPIs`.		The monitoring KPIs can be consumed by an Integration Server service or written to a CSV file (or both).
`Create CSV file`	If this parameter is enabled, the CSV files are stored in the subfolder *appmondc* of the resources folder of the package *WmEntireX*, for example *.../IntegrationServer/instances/default/packages/WmEntireX/resources/appmondc*.	enabled	
`Maximum number of lines per CSV file`	The maximum number of rows per CSV data file. If the limit is reached, a new file is created.	100000	
`Use "0" as the null value for numeric KPI values`	Use "0" instead of an empty entry as the null value for all numeric KPI values in the CSV file.	disabled	
`Create a new CSV file every day`	A new CVS data file is created automatically every day.	disabled	

9 Built-in Services for Creating Document Types, Flows and IDL Files

The following built-in services are available in the WmEntireX package. They can be used to create document types, flows and IDL files from EntireX Adapter objects and/or IDL files. Parameters are either input or output (I/O).

Service	Parameter	I/O	Description
pub.wmentirex.listener:createDocumentTypes This service creates document type objects, using the metadata stored in the listener, connection or adapter service object. Only objects created by the EntireX Adapter are supported. You can also use a Software AG IDL file as input. A document type named <programname>Request is created for the input parameters of each program. A document type named <programname>Response is created for the output parameters of each program.	input	I	Name of a listener, adapter service, connection, or pathname of an IDL file.
	packageName	I	Name of the package where the created document types will be stored.
	namespace	I	Namespace of the folder where the created document types will be stored.
	mapToString	I	Defines how IDL data types are mapped to Integration Server data types. See *Mapping IDL Data Types to IS Data Types* in the Integration Server Wrapper documentation. Only used when a connection or IDL file is used as input.
	result	O	Either a success message if the creation was successful (which includes the number of created document types), or an error message.
pub.wmentirex.listener:createFlow	listener	I	Name of the listener.
	packageName	I	Name of the package where the created flow(s) will be stored. If this

Service	Parameter	I/O	Description
This service creates flow objects for the specified EntireX Adapter listener object. For each service name specified in the definition of an EntireX Listener, an empty flow service with the appropriate input and output signature is created if this service does not exist. If the service exists it will remain unchanged.			parameter is not specified, the flow(s) will be stored in the package of the listener.
	`result`	O	Either a success message if the creation was successful (which includes the number of created flow objects), or an error message.
`pub.wmentirex.listener:createIDL` This service creates an IDL file, using the metadata which is stored in the listener, connection or adapter service object. Only objects created by the EntireX Adapter are supported.	`input`	I	Name of a listener, connection or adapter service.
	`filename`	I	The name of the IDL file (with extension ".idl"). You cannot specify a folder name. The file is created in the folder *<IntegrationServer_instance>/packages/WmEntireX/resources*.
	`result`	O	Either a success message if the creation was successful (which includes the full pathname of the IDL file), or an error message.

10 Direct RPC

Direct RPC is a component that enables RPC clients and RPC servers to connect directly to the EntireX Adapter without the EntireX Broker.

Configuring Direct RPC

To enable Direct RPC, choose **Direct RPC Administration** in the EntireX Adapter's administration menu and follow the wizard. The table below gives an overview of parameters to be specified:

Name	Values	Default	Action	Description
Status	Stopped, Running	Stopped	Start or stop.	Current status of the Direct RPC component.
Logging Level	Fatal: Fatal errors, Direct RPC cannot work correctly. Error: Severe errors, Direct RPC continues to work correctly. Warning: Errors with low severity. Info: To follow the flow of requests between RPC clients, servers and Direct RPC. Debug: All communication buffers will be written to the log file. Trace: Internal use only (Software AG Support).	Fatal	Change logging level.	Logging level. Log file *wmentirex.directrpcYYYYMMDD.log* is located in folder *logs* in the Integration Server installation folder. To change the default location, see ***Adapter Settings***. It is not recommended to use "Info", "Debug" or "Trace" in a production environment.
TCP Port Number	Port number	1971	Change port number.	TCP listening port for Direct RPC.
SSL Port Number	Port number	-	Change port number.	SSL/TLS listening port for Direct RPC.
Keystore Alias (for SSL)	The alias name of a keystore defined in the Integration Server.	-	Change alias name.	Only needed for SSL. Defined in the IS Web Administration **Security** > **Keystore**.
FIPS Mode	Enabled, Disabled	Disabled	Enable or disable	Running in FIPS 140-2 compliant security mode.
Auto Start	Enabled, Disabled	Disabled	Enable or disable.	Automatic start of Direct RPC during startup of the Adapter.

Encoding for RPC Clients and Servers (Default and Available Codepages)

The default encoding of an RPC client or server is the same as the default encoding of the platform used. The table below gives an overview of default and available codepages:

Platform	Default Codepage	Available Codepages (single byte only)
Windows/UNIX/Linux	ISO 8859-1	ASCII codepages
IBM mainframe (z/OS, z/VSE)	CP037	EBCDIC codepages
Fujitsu mainframe (BS2000)	CP273	EBCDIC codepages

To specify the encoding of an RPC client or server, see *Configuration and Usage* for links to the respective sections of the documentation.

Monitoring

The following table gives an overview of monitored objects:

Item	Description
Servers	Number of registered servers.
RPC Requests	Number of RPC requests.
Socket Connections	Number of plain socket connections (TCP) from RPC clients or servers.
Secure Socket Connections	Number of secure socket connections (SSL/TLS) from RPC clients or servers.
Direct Connections	Number of direct connections from Adapter services or Adapter Listeners.
Calls	Number of calls (including calls such as `logon`).
Conversations	Number of conversational calls.

Notes:

1. For each item the following information is provided: Current (number of items currently active), maximum (high watermark) and total (number of activations so far).
2. If Direct RPC connections are enabled, the value of `Direct Connections` can be greater than zero even if Direct RPC is stopped.

Services

The following information is displayed for each service:

Name	The name of the service
Type	"External RPC Server" or "Direct RPC Server".
Instances	Number of registered replicates.
Wait for Server	Number of client requests that are waiting for a free server to be processed. The percentage (number of calls waiting compared to the total number of calls) is helpful in deciding if further server replicates are necessary.
Action	Shutdown (only for external RPC servers).

Note: As a prerequisite, at least one server must be registered or listener enabled.

Servers

The following information is displayed for each service:

Service Name	The name of the service
Type	"Listener" or "RPC Server".
Name	Name of the Direct Listener or the external RPC server.
Host	Host where the external RPC server is running.
Version	Version of the external RPC server.
Start Time	Start time of the RPC server.
Action	Shutdown (only for external RPC servers).

Note: `Host`, `Name` and `Version` can only be displayed if the RPC server provides this information.

Built-in Services for Direct RPC

The functionality of Direct RPC as shown on the IS Administration Console's page is available with built-in services in the WmEntireX package. They correspond to the parameters on the IS Administration Console's page. When a service is executed, the signature of the input and output parameters is shown. The following services are available:

- `pub.wmentirex.directrpc.admin:getAutoStart`
- `pub.wmentirex.directrpc.admin:getIbmMainframeDefaultCharset`
- `pub.wmentirex.directrpc.admin:getKeyStoreAlias`
- `pub.wmentirex.directrpc.admin:getLoggingLevel`
- `pub.wmentirex.directrpc.admin:getSSLPortNumber`
- `pub.wmentirex.directrpc.admin:getSiemensMainframeDefaultCharset`
- `pub.wmentirex.directrpc.admin:getStatus`
- `pub.wmentirex.directrpc.admin:getTCPPortNumber`
- `pub.wmentirex.directrpc.admin:getWindowsUnixDefaultCharset`
- `pub.wmentirex.directrpc.admin:setAutoStart`
- `pub.wmentirex.directrpc.admin:setIbmMainframeDefaultCharset`
- `pub.wmentirex.directrpc.admin:setKeyStoreAlias`
- `pub.wmentirex.directrpc.admin:setLoggingLevel`
- `pub.wmentirex.directrpc.admin:setSSLPortNumber`
- `pub.wmentirex.directrpc.admin:setSiemensMainframeDefaultCharset`
- `pub.wmentirex.directrpc.admin:setTCPPortNumber`
- `pub.wmentirex.directrpc.admin:setWindowsUnixDefaultCharset`
- `pub.wmentirex.directrpc.admin:start`
- `pub.wmentirex.directrpc.admin:stop`

In addition, there are 8 services for advanced configuration of the SSL transport:

- `pub.wmentirex.directrpc.admin:getSSLClientAuthentication`
- `pub.wmentirex.directrpc.admin:getTrustStoreAlias`
- `pub.wmentirex.directrpc.admin:getTLSProtocols`
- `pub.wmentirex.directrpc.admin:getCipherSuites`
- `pub.wmentirex.directrpc.admin:setSSLClientAuthentication`
- `pub.wmentirex.directrpc.admin:setTrustStoreAlias`

- `pub.wmentirex.directrpc.admin:setTLSProtocols`
- `pub.wmentirex.directrpc.admin:setCipherSuites`

Note: Use the `setSSLClientAuthentication` service to switch on SSL client authentication. Client authentication requires that a truststore alias has been set with `pub.wmentirex.directrpc.admin:setTrustStoreAlias`. The truststore alias is defined in the IS web admin **Security** > **Keystore**. Note that the names of the keystore alias and the truststore alias must be different.

Use the `setTLSProtocols` service to change the set of enabled TLS protocols. The default is defined by the Java Runtime. Only protocols that are supported by the Java Runtime can be used. The set of enabled and supported protocols are shown in the log file if the logging level is set to DEBUG or TRACE. Note that the `getTLSProtocols` service returns the value previously set by the `setTLSProtocols` service.

Use the `setCipherSuites` service to change the set of enabled cipher suites. The default is defined by the Java Runtime. Only cipher suites that are supported by the Java Runtime can be used. The set of enabled and supported cipher suites are shown in the log file if the logging level is set to DEBUG or TRACE. Note that the `getCipherSuites` service returns the value previously set by the `setCipherSuites` service.

Limitations of Direct RPC

- Reliable RPC is supported using the Direct RPC Reliable Listener Connection. Other use cases are not supported. See *Connection Parameters for Direct RPC and Direct Reliable RPC Listener Connections*.
- Command and Information Services (CIS) are not supported. Using this API can result in errors, or the data returned is incomplete or missing.
- An adapter listener and an external RPC server cannot be registered for the same service.
- Replicates of external RPC servers must use the same encoding.
- EntireX Security is not supported.
- ACI applications are not supported. For Natural RPC clients or servers, Broker ACI is only supported for versions higher than 1. See the documentation on the `ACIVERS` parameter in *Parameter Reference* in the latest *Natural for Mainframe* documentation under *https://empower.softwareag.com/Products/Documentation/default.asp*.

11 IDL Extraction from Integration Server

Integration Server services (e.g. flow services) can be called from an RPC client, using an adapter listener. To create a listener you need a Software AG IDL file, which describes the call interface (signature) of the Integration Server service. This IDL file can be created automatically using the IDL Extractor for Integration Server, a component of the Designer that reads a package from the Integration Server and generates an IDL file from:

- all services contained in the package (if you are using an EntireX Adapter version 10.3 or lower)
- all services contained in the package or a subset thereof (if you are using an EntireX Adapter version 10.5 or above)

As an alternative, you can create an IDL file from a specified subset of Integration Server services, using service `pub.wmentirex.listener:generateIDLfromService` from the WmEntireX package. This alternative is described below, along with an overview of data type mapping relevant for both approaches.

Using the Service pub.wmentirex.listener:generateIDLfromService

Service `pub.wmentirex.listener:generateIDLfromService` generates an IDL file for a given list of Integration Server services. For each service, a program in the IDL file is created. If the IDL file already exists, it is extended with the newly created program definitions. Duplicate definitions for the same program are possible. In this case, the IDL file has to be corrected manually. Note that this service creates/updates only the IDL file. Creating or updating a listener with the definitions of the IDL file has to be done with *Integration Server Wrapper* of the Designer. The following parameters are relevant. Parameter direction can be In or Out:

Parameter	I/O	Description
`serviceName` *`String`*	I	The full name of the Integration Server service (e.g. *`folder1`*`.`*`folder2`*`:`*`service`*). Multiple services can be specified using a semicolon (;) as delimiter.
`fileName` *`String`*	I	The name of the IDL file (with extension ".idl"). You cannot specify a folder name. The file is created in the folder *<IntegrationServer_instance>/packages/WmEntireX/resources*.
`libraryName` *`String`*	I	The library name to be used in the IDL file. If the IDL file already exists, this parameter is ignored.
`stringType` *`String`*	I	The IDL data type which is used for string data types. Possible values are `AV`, `AVn`, or `An`. This parameter is optional, default is `AV`.
`language` *`String`*	I	The target language for the RPC clients. Possible values are `COBOL`, `Natural`, `PL/I`, or `Other`.
`result` *`String`*	O	Either a success message if the generation was successful (which includes the full pathname of the IDL file) or an error message.

Integration Server Data Types to IDL Mapping

The signature of an Integration Server service specifies the data types for the parameters of the service. There are three data types: `String`, `Record` and `Object`. Parameters of type `String` are mapped to an IDL alphanumeric data type, parameters of type `Record` to IDL groups. The mapping of parameters of type `Object` has been enhanced in recent versions of the EntireX Adapter. For details see table below.

Integration Server Object Type Java Wrapper Types	Software AG IDL Data Type			
	Adapter Version 10.5	Adapter Version 10.3	Adapter Version 9.12 Fix 1 or 10.1	Adapter Version 9.12 or lower
`java.lang.String`	`A`*n*`, AV, AV`*n*[(1)(3)]	`A`*n*`, AV, AV`*n*[(1)]	`A`*n*`, AV, AV`*n*[(1)]	`A`*n*`, AV, AV`*n*[(1)]
`java.lang.Long`	`N19`	`N19`	`N19`	error
`java.lang.Integer`	`I4`	`I4`	`I4`	error
`java.lang.Short`	`I2`	Input parameter: ignored; Output parameter: `A`*n*`, AV, AV`*n*[(1)]	error	error
`java.lang.Byte`	`I1`	Input parameter: ignored; Output parameter: `A`*n*`, AV, AV`*n*[(1)]	error	error
`java.lang.Float`	`F4`	`F4`	`F4`	error
`java.lang.Double`	`F8`	`F8`	`F8`	error
`java.lang.Boolean`	`L`	`L`	`L`	error
`java.util.Date`	A24 (format 2017-08-15T10:53:35.087Z)	`A`*n*`, AV, AV`*n*[(1)]	error	error
`java.math.BigInteger`	`A`*n*`, AV, AV`*n*[(1)]	Input parameter: ignored; Output parameter: `A`*n*`, AV, AV`*n*[(1)]	error	error
`java.math.BigDecimal`	`A`*n*`, AV, AV`*n*[(1)]	Input parameter: ignored; Output parameter:`A`*n*`, AV, AV`*n*[(1)]	error	error
`byte[]`	`BV`, `BV256` (depending on target language)	`BV`, `BV256` (depending on target language)	`BV`, `BV256` (depending on target language)	error

Integration Server Object Type Java Wrapper Types	Software AG IDL Data Type			
	Adapter Version 10.5	Adapter Version 10.3	Adapter Version 9.12 Fix 1 or 10.1	Adapter Version 9.12 or lower
All other wrapper types or record without subcomponents	Input parameter: ignored; output parameter: `An`, `AV`, `AVn`[1]	Input parameter: ignored; output parameter: `An`, `AV`, `AVn`[1]	error	error
No wrapper type	`BV`, `BV256` if extracting from EntireX Adapter Service [4]; otherwise see above	`BV`, `BV256` if extracting from EntireX Adapter Service [4]; otherwise see above	error; or `BV`, `BV256` if extracting from EntireX Adapter Service [4]	error; or `BV`, `BV256` if extracting from EntireX Adapter Service [4]

Parameters of type `String` may have an associated context type which is specified in the constraints of the parameter's properties. This content type may influence the data type mapping. The following table shows the mapping.

Integration Server Content Type	Software AG IDL Data Type	
`boolean`	`L`	Logical
`decimal`	`Nx`, `Nx.y` or `An`, `AV`, `AVn`	Unpacked decimal or Alphanumeric [5]
`float`	`F4`	Floating point (small)
`double`	`F8`	Floating point (large)
`Date`	`An`, `AV`, `AVn`	Alphanumeric [1]
`dateTime`	`An`, `AV`, `AVn`	Alphanumeric [1]
`base64Binary`	`An`, `AV`, `AVn`	Alphanumeric [1]
`byte`	`I1`	Integer (small)
`unsignedByte`	`NU3`	Unpacked decimal unsigned
`short`	`I2`	Integer (medium)
`unsignedShort`	`NU5`	Unpacked decimal unsigned
`int`	`I4`	Integer (large)
`unsignedInt`	`NU10`	Unpacked decimal unsigned
`long`	`N19`	Unpacked decimal
`unsignedLong`	`NU20`	Unpacked decimal unsigned
`integer`	`N29`	Unpacked decimal
`positiveInteger`	`NU29`	Unpacked decimal unsigned
`nonPositiveInteger`	`N29`	Unpacked decimal
`negativeInteger`	`N29`	Unpacked decimal
`nonNegativeInteger`	`NU29`	Unpacked decimal unsigned

Integration Server Content Type	Software AG IDL Data Type	
`string`	A*n*, AV, AV*n*	Alphanumeric [2]
`all others`	A*n*, AV, AV*n*	Alphanumeric [1]

Notes:

1. All parameters of type `String` are mapped to an IDL alphanumeric data type, available as variable (`AV`, `AVn`) or fixed (`An`) length. Which alphanumeric type is used is specified in the wizard page of the IDL Extractor for Integration Server or as a parameter of the generation service.
2. If the content type specifies a length `len`, then `Alen` is used. If the content type specifies a maximum length `max`, then `AVmax` or `Amax` is used, see [1] for the choice between `A` versus `Av`. If neither length nor maximum length is specified, the choice is the same as in [1].
3. If an input parameter of a service has a default value, this value is shown in the extracted IDL file as a comment. When the Adapter Listener calls this service and no value (or only a value with blanks) is specified by the RPC client, the default value is used as the parameter value.
4. A parameter of an EntireX Adapter service of type `Object` is mapped to an IDL binary data type. Depending on the target language, it is either mapped to `BV` (for `Natural` and `Other`) or to `BV256` (for `COBOL` and `PL/I`).
5. If the content type specifies a total number of digits and a number of fraction digits, then `Nx.y` is used. If only a total number of digits is specified, then `Nx` is used. If no total number of digits is specified, or the total number exceeds the maximum allowed value, or the number of fraction digits exceeds the maximum allowed value, then a string according to [1] is used. The maximum allowed value for the total number of digits is 29, for COBOL it is 31. The maximum allowed value for the number of fraction digits is 7, for COBOL it is 31 and for Natural it is 29.

12 SSL/TLS and Security Support

Security Support for Adapter Services

The EntireX Adapter uses adapter services to send synchronous or asynchronous requests to various back-end systems. Access to these systems can be secured in the following ways:

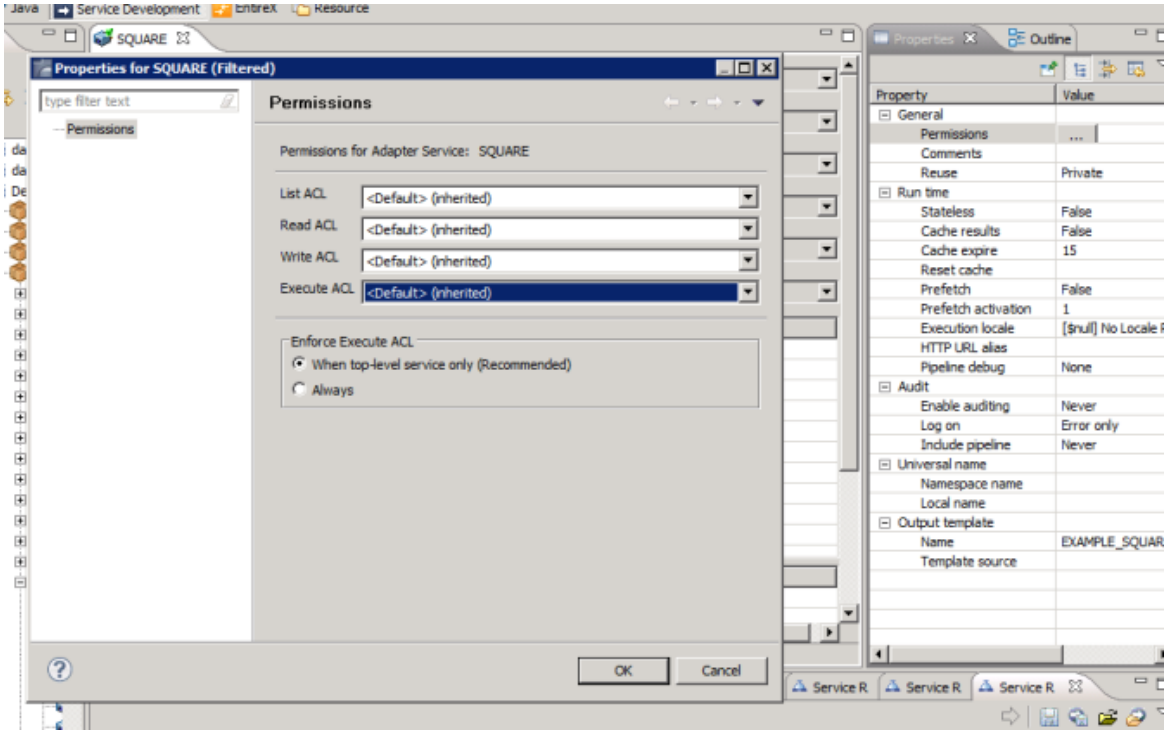
- **Provide security credentials in the adapter connection for security-protected back-end systems**
  If access to a back-end system is security protected, security credentials (user ID and password) have to be provided in the corresponding adapter connection. This can be done when the connection is configured or edited, see *EntireX Adapter Connections*. This security support is optional and not available for all connection types, see the overview of available connection parameters in *Editing Adapter Connections*.

  The credentials can also be provided dynamically when calling the adapter service, see *EntireX Adapter Services*.

- **Set access control permissions for the adapter service**
  Using the Designer you can set access control permissions to restrict the right to execute an adapter service to a particular user group of the Integration Server:

  To set access control permissions using the Designer

  1 Choose **Service Development** > **Package Navigator** > **Permissions**.



2 Set **Enforce Execute ACL** to **When top-level service only** to select the group that is specified on the top-level service. This way, access control is already checked when executing the top-level service.

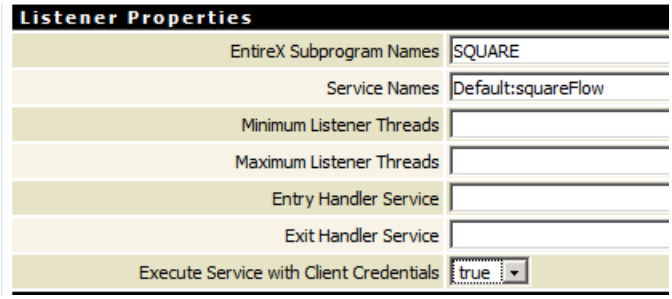
Or:

Set **Enforce Execute ACL** to **Always** and select a user group in **Execute ACL**.

Security Support for Adapter Listeners

Adapter listeners receive synchronous or asynchronous requests from external RPC clients. If the RPC request is sent to the EntireX Adapter using a broker that is configured to use security, the credentials of the RPC clients are checked to see if the client is permitted to call the adapter listener.

By enabling the listener property **Execute Service with Client Credentials**, an adapter listener can be configured to use the credentials (user ID and password) provided by the RPC client to execute the Integration Server service:



Then the RPC client has to provide credentials (RPC user ID and RPC password) which are used to execute the Integration Server service.

Support for SSL/TLS

Communication between the Adapter and the back-end systems is done using TCP/IP or, for secure communication, SSL/TLS. The configuration of SSL depends on the connection type:

Connection Type	Description	Notes
Connection types using the EntireX Broker	Use the URL-style syntax by which Broker IDs are defined to select SSL as transport method and to define the SSL parameters. For example: `ssl://host:1958?definition_of_sslparameters`. The syntax for defining SSL parameters is described in the notes below.	1,2,3,4,5
IMS Connect, CICS ECI and CICS Socket Listener	Specify the SSL port in the **Port** property and the SSL parameters in the **SSL Parameters** property as described in *Editing a Connection*. The syntax for defining SSL parameters is described in the notes below.	1,3
Direct RPC	Specify the SSL port and SSL parameters in the configuration wizard for *Direct RPC*.	1,5

Notes

1. To operate with SSL, certificates need to be provided and maintained. Depending on the platform, Software AG provides default certificates, but we strongly recommend that you create your own. See *SSL/TLS Sample Certificates Delivered with EntireX* in the EntireX Security documentation.
2. Specify Broker ID and SSL parameters.

   SSL transport will be chosen if the Broker ID starts with the string `ssl://`. Example of a typical *URL-style Broker ID*:

   ```
   ssl://host:1958?definition_of_sslparameters
   ```

   If no port number is specified, port 1958 is used as default.
3. If the SSL client checks the validity of the SSL server only, this is known as *one-way SSL*. The mandatory `trust_store` parameter specifies the file name of a keystore that must contain the list of trusted certificate authorities for the certificate of the SSL server. By default a check is made that the certificate of the SSL server is issued for the hostname specified in the Broker ID. The common name of the subject entry in the server's certificate is checked against the hostname. If they do not match, the connection will be refused. You can disable this check with SSL parameter `verify_server=no`.

   If the SSL server additionally checks the identity of the SSL client, this is known as *two-way SSL*. In this case the SSL server requests a client certificate (the parameter `verify_client=yes` is defined in the configuration of the SSL server). Two additional SSL parameters must be specified on the SSL client side: `key_store` and `key_passwd`. This keystore must contain the private key of the SSL client. The password that protects the private key is specified with `key_passwd`.

   The ampersand (&) character cannot appear in the password.

   SSL parameters are separated by ampersand (&). See also *SSL/TLS Parameters for SSL Clients*.

   Example of one-way SSL:

```
ssl://host:1958?trust_store=/temp/ExxCACert.jks&verify_server=no
```

Example of two-way SSL:

```
ssl://host:1958?trust_store=/temp/ExxCACert.jks&key_store=/temp/ExxJavaAppCert.jks&key_passwd=ExxJavaAppCert
```

4. Make sure the SSL server to which the EntireX Adapter (service or listener) connects is prepared for SSL connections as well. The SSL server can be EntireX Broker or Broker SSL Agent. See:
   - *Running Broker with SSL/TLS Transport* in the platform-specific Administration documentation
   - Broker SSL Agent in the UNIX | Windows Administration documentation
5. For information on how to configure other EntireX components using SSL/TLS, see *Using SSL/TLS with EntireX Components*.

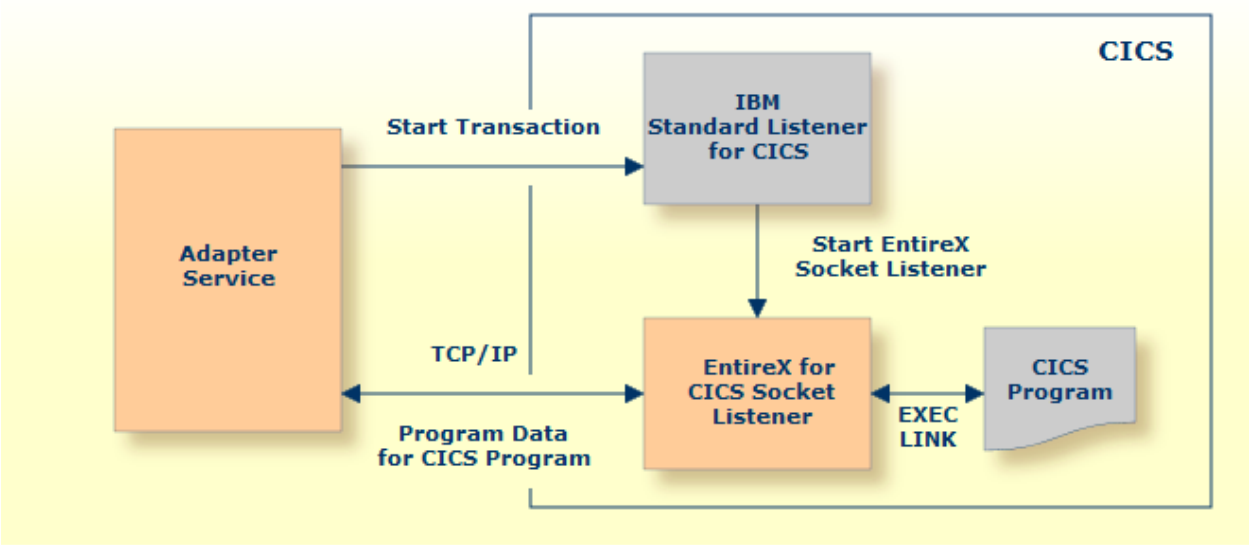
Built-in Service to Generate a RACF PassTicket

Service	Parameter	I/O	Description
`pub.wmentirex.RACFPassTicket:generate` This service generates a RACF PassTicket. The generated PassTicket can be used in all service calls which require a RACF password. The PassTicket is specified in the password parameter of the service call. Errors in the generation are returned as an exception.	userId	I	The RACF User ID used to generate the PassTicket.
	applicationName	I	Application name used to generate the PassTicket.
	securedSignonKey	I	Secured signon key used to generate the PassTicket.
	passTicket	O	The generated PassTicket.

13 Preparing for CICS Socket Listener

Overview

The CICS Socket Listener is used by the EntireX Adapter for CICS Socket Listener Connections. Apart from installation there is no configuration necessary in CICS. Configuration is done with the Adapter Connection. See *Connection Parameters for CICS Socket Listener Connections*.



The implementation for CICS is based on the CICS standard listener provided by IBM. With this listener you can launch CICS transaction via TCP/IP. The launched transaction takes the TCP connection and continues the communication with the launching process.

Depending on your platform, more information on configuring the IBM standard listener for CICS can be found under the following IBM documentation:

- *z/OS Communications Server: IP CICS Socket Guide*
- *z/VSE TCP/IP Support*

Installing the CICS Socket Listener

The CICS Socket Listener is installed

- together with the RPC Server for CICS, see *Installing the RPC Server for CICS*, or
- separately, see *EntireX CICS Socket Listener* (z/OS | z/VSE)

Configuring the IBM Standard Listener

Depending on your platform, more information on configuring the IBM standard listener for CICS can be found under the following IBM documentation:

- *z/OS Communications Server: IP CICS Socket Guide*
- *z/VSE TCP/IP Support*

To start/stop the IBM standard listener

- Use the CICS supplied transaction EZAO. The listener is automatically started/stopped when CICS is started or stopped.

To configure the IBM standard listener

- Use the CICS-supplied transaction `EZAC,ALT,LISTENER`.
  - For `SECEXIT`, define `EXXRFECS`.
  - Make sure the `PORT` number of the IBM standard listener corresponds to the configuration parameter `Port`. See ***Connection Parameters for CICS Socket Listener Connections***.

Automatic Syncpoint Handling

After each non-conversational RPC request (or the end of a series of conversational RPC requests), an implicit `CICS COMMIT` (or `CICS ROLLBACK` if an error occurs) is executed by the CICS Socket Listener. If you are running with user transaction support, syncpoint handling is executed by CICS itself when the user transaction terminates.

User Exit

To enable a user exit, use the property `cics.sl.userexit` to specify the class name of the user exit implementation. The class will be loaded using the standard classpath. You can specify a separate classpath with the property `cics.sl.userexit.classpath`. Note that for the classpath, a file or HTTP URL must be specified, for example `file://myexit.jar` or `http://host/path/to/my/exit`.

Your user exit class must implement the Java interface `com.softwareag.entirex.cics.socketlistener.IUserExit`. This Java interface has the following method:

```
/**
   * Read and modify the CICS Socket Listener request payload before sending the ↩
request.
   * Get access to the ConfigurationParameters.
   *
   * @param requestPayload CICS Socket Listener Payload to be send to CICS, ↩
containing a java.nio.ByteBuffer
   * @param properties CICS Socket Listener Configuration Parameters with access ↩
to the user transaction id and several
   * Bridge framework parameters, like IDL program name, IDL library name, target ↩
program name etc.
   */
  void beforeSend(Payload requestPayload, ConfigurationParameters properties);
```

14 Preparing IBM CICS for ECI

This chapter describes how to set up the External Call Interface (ECI) within CICS.

Important: If the terms and concepts in this chapter are unfamiliar to you, ask an appropriate CICS system programmer. Only authorized personnel should make changes to mainframe computer systems.

Defining an ECI Service

The DFH$SOT group contains three TCP/IP services. For our purposes, the pertinent service is ECI. The ECI service has, defined in it, the TCP/IP port number through which the CICS region listens to the ECI. By default, IBM predefines the TCP/IP port number as "1435". If this port is already reserved for another CICS region, you may have to define a different port number in the ECI service. For more information, see your IBM documentation.

You must specify SOCKETCLOSE(NO) on the TCPIPSERVICE. If you need to specify a timeout for a task initiated using ECI over TCP/IP, specify an RTIMOUT value on the mirror transaction. Note that the standard mirror, CPMI, is defined with profile DFHCICSA, which specifies RTIMOUT(NO). This means that long running mirrors will wait indefinitely for data, unless you specify a different RTIMOUT for the mirror transaction.

To define an ECI service without security

1 Use CEDA ALTER to supply the ECI TCPIPService with a unique TCP/IP port number.

2 Use CEDA ALTER to set the ECI ATTACHSEC to "LOCAL"

To define an ECI service with security

1 Copy the ECI TCPIPService to an ECIS TCPIPSERVICE.

2 Use CEDA ALTER to supply the ECIS TCPIPSERVICE with a unique TCP/IP port number.

3 Use CEDA ALTER to set the ECIS ATTACHSEC to "VERIFY".

Installation Verification

A successful installation should pass the following verification tests:

1. CEMT INQUIRE TCPIPSERVICE(*) should now display your services with status OPE. A status of CLO might indicate an already used (not unique) TCP/IP port number.
2. CEMT I TCPIPSERVICE(ECI) to verify TCPIPSERVICE in CICS.
3. CEMT I TRA(CIEP) to verify transaction CIEP in CICS.

4. `CEMT I PROGRAM(DFHIEP)` to verify program `DFHIEP` in CICS.
5. `CEMT I TRAN(CPMI)` to verify transaction `CPMI` in CICS.
6. `CEMT I PROGRAM(DFHMIRS)` to verify program `DFHMIRS` in CICS.
7. `CEMT I TD(CIEO)` to verify program `TQ` queue `CIEO`.
8. Verify that the groups `DFHISC DFHDCTG` and `DFHIPECI` are added to the active autoinstall.

Tip: You can check the mentioned programs and transactions and TD queues by using `CEMT INQUIRE TCPIPSERVICE(*)`.

Error Handling

This table describes the handling of errors in the CICS ECI connection or the RPC Server for CICS ECI.

Problem	Handling
A CICS program sends abend code in response.	The CICS session is closed and the next call opens a different session.
The TCP/IP connection is lost with a `SocketTimeoutException`.	The CICS session is closed and the next call opens a different session.
The TCP/IP connection is lost with an `EOFException`.	■ The TCP/IP socket is closed and the next call opens a different session. ■ There are no further attempts to send bytes on the TCP/IP connection that received the `EOFException`.

15 Preparing for IMS Connect

This chapter describes how to prepare for connections to IMS.

Important: If the terms and concepts in this chapter are unfamiliar to you, ask an appropriate IMS system programmer. Only authorized personnel should make changes to mainframe computer systems.

Extracting from Message Format Service (MFS)

To extract interface definitions from Message Format Service (MFS) with MID and MOD definitions, use a command-line extractor. Run the extractor with the following command:

```
java -classpath <suite installation ↩
folder>\IntegrationServer\packages\WmEntireX\code\jars\entirex.jar ↩
com.softwareag.entirex.ims.extractor.MFSExtractor <inputfile>
```

The input file must be an MFS source with MID and MOD definitions. The output is an IDL file with the same name as the input file and suffix ".idl". Use the IDL file to generate connections and adapter services with the Integration Server Wrapper.

Related Literature

- Extracting from a COBOL source is described under *IMS MPP Message Interface (IMS Connect)* in the IDL Extractor for COBOL documentation.
- For troubleshooting, see *Message Class 2011 - Connections to IMS Connect* in the Error Messages and Codes documentation.

16 Converting IS Data Structures with the COBOL Converter

Adapter services for the connection type COBOL Converter are used to convert Integration Server data structures from/to a byte array representing the COBOL binary data. The IS data structure is described by a *Software AG IDL File* in the IDL Editor documentation (with an optional server mapping file, see *Server Mapping Files for COBOL*), which is created using the IDL Extractor for COBOL. The interface type COBOL Converter has to be used for this purpose. See *COBOL Converter* (In same as Out, In different to Out).

Note: For interface type COBOL Converter, COBOL input and output is either described by the same layout ("In same as Out") or the input is overlaid by a different output layout ("In different to Out"). See *COBOL Mapping Editor*.

The COBOL Converter functionality can be used in scenarios where COBOL binary data is either already available in IS, or required by an IS service. Typical examples are:

- working with files that contain COBOL binary data
- COBOL binary data with a complex structure consumed or produced by components such as the WebSphere MQ Adapter
- COBOL binary data from EntireX Adapter connections such as AS/400, CICS ECI, IMS Connect, CICS Socket Listener Connections

The conversion depends on the signature of the generated adapter service:

- **Input Only**
  If the adapter service has only input parameters (structure `inRec`) they are converted to a COBOL binary data structure (byte array) and stored in the output parameter `cobolOutput`.
- **Output Only**
  If the adapter service has only output parameters (structure `outRec`) the COBOL binary data structure (byte array) contained in the input parameter `cobolInput` is converted to the output parameters.

- **Input and Output**
  If the adapter service has both input and output parameters (structures `inRec` and `outRec`), input parameter `cobolInput` is checked:
  - If `cobolInput` is *empty*, the structure `inRec` is converted to a COBOL binary data structure (byte array) and stored in the output parameter `cobolOutput`.
  - If `cobolInput` is *not empty*, the contained COBOL binary data structure (byte array) is converted to the output parameters.

17 Extracting IDL using the REST API

Introduction

You can call the IDL Extractor for Integration Server from a command line, using the REST API. A simple service performs the extraction only; an advanced service also creates a listener connection and listener services. The returned IDL file representation is part of the JSON response in standard Base64 encoding, so you can easily save the content to disk after decoding. Base64 encoding/decoding is used to keep the source file formatting inside the JSON format. See *Base64 Resources* for information on Base64 encoding/decoding tools.

To extract IDL

- Send the request to the Integration Server to the following URL:

```
http://integrationServerHostName:5555/restv2/wmentirex/extractIdl
```

Example:

```
http://localhost:5555/restv2/wmentirex/extractIdl
```

To extract IDL and create listener connection service

- Send the request to the Integration Server to the following URL:

```
http://integrationServerHostName:5555/restv2/wmentirex/extractIdlAndCreateListenerConnection
```

Example:

```
http://localhost:5555/restv2/wmentirex/extractIdlAndCreateListenerConnection
```

Extracting IDL

- Response Codes

Configuring your Input Parameters

To configure your input parameters

- Use a JSON request document. The following parameters are available:

Parameter	Opt/ Req	Description
`sourcePackageName`	R	The package name from you want to extract.
`services`	O	An array of full names of Integration Server services available in the source package, for example `pub.string:toLower`.
`stringType`	O	The IDL data type that is used for string data types. Possible values: `AV` (default), `AV`*n*, `A`*n*.
`language`	R	The target language for the RPC clients. Possible values: `COBOL`, `Natural`, `PL/I`, `Other`.

Sample request document `example.json`:

```
{
     "sourcePackageName": "EntireXDemo",
     "services": [
          "folder1.folder2:service1",
          "folder1.folder2:service2"
     ],
     "language": "Natural"
}    ↩
```

Calling the REST Service

To call the REST service

- Provide the host name, port and credentials for the Integration Server.

  This example uses the executable `curl`:

```
curl -u <username>:<password> -X POST -H "Content-Type: application/json"
 -d @.\example.json ↩
http://<integrationServerHostName>:5555/restv2/wmentirex/extractIdl
```

Response Codes

The HTTP response is in JSON format.

- **Successful Request**
  If a request is successful, the following parameters are returned and `error` is empty. Example:

```
{
     ↩
"idlSourceBase64":"LyogR2VuZXJhdGVkIGJ5IFNvZnR3YXJlIEFHLCBJREwgRW5OaXJlWCAuLi4=",
     "mapToString":"false",
     "error":null
}
```

  where `idlSourceBase64` is the Base64-encoded source, and

  `mapToString` indicates whether all data items were mapped to `String`. Valid values: `true`, `false`.

  The HTTP response code is 200 (OK).

- **Unsuccessful Request**
  If a request is not successful, the `error` parameter contains the reason for the failure. Example:

```
{
  "idlSourceBase64": null,
  "error": "Required parameter language is missing"
}
```

  The HTTP response code is 500 (internal server error).

- **Invalid Syntax**
  If the JSON input file has a syntax error (invalid input), the error parameter contains the reason for the JSON parser failure:

```
{
  "idlSourceBase64": null,
  "error": {
  "$errorDump": "com.fasterxml.jackson.core.JsonParseException:Unexpected character
  (';' (code 59)): was expecting comma to separate Object entries\n at [Source:
  (com.wm.net.HttpInputStream); line: 1, column: 36]",
...
}
```

The HTTP response code is 400 (bad request).

Extracting IDL and Creating a Listener Connection Service

Configuring your Input Parameters

To configure your input parameters

- Use a JSON request document. The following parameters are available:

Parameter	Opt/Req	Description	Note
`sourcePackageName`	R	The package name from which you want to extract.	
`services`	O	An array of full names of Integration Server services available in the source package, for example `pub.string:toLower`.	
`stringType`	O	The IDL data type that is used for string data types. Possible values: `AV` (default), `AVn`, `An`.	
`language`	R	The target language for the RPC clients. Possible values: `COBOL`, `Natural`, `PL/I`, `Other`.	
`targetPackageName`	R	The package name to store the connection and listener services.	
`connectionType`	R	The type of the listener connection. Possible values: `RpcListenerConnection`, `DirectRpcListenerConnection`, `ReliableRpcListenerConnection`, `DirectReliableRpcListenerConnection`.	Reliable RPC does not allow output parameters inside the extracted IDL.
`folderName`	R	The name of the folder inside the target package.	
`connectionName`	R	The name of the listener connection.	
`listenerName`	R	The name of the listener service.	
`serverAddress`	R	The address of the RPC server. The address is given in the format `<class>/<server>/<service>`.	
`brokerID`	R	The Broker ID.	Not applicable to Direct RPC Listener Connections.

Parameter	Opt/Req	Description	Note
`userid`	O	The user ID for secured communication.	
`password`	O	The password for secured communication.	
`encoding`	R	The data encoding.	

Sample request document `example.json`:

```
{
     "sourcePackageName": "EntireXDemo",
     "services": [
          "folder1.folder2:service1",
          "folder1.folder2:service2"
     ],
     "language": "Natural",
     "targetPackageName": "CustomerPackage",
     "connectionType": "DirectRpcListenerConnection",
     "folderName": "folder3.folder4",
     "connectionName": "connection1",
     "serverAddress": "RPC/JSON1/CALLNAT",
     "listenerName": "listener1",
     "encoding": "UTF-8"
}
```

Calling the REST Service

To call the REST service

- Provide the host name, port and credentials for the Integration Server.

  This example uses the executable `curl`:

```
curl -u <username>:<password> -X POST -H "Content-Type: application/json"
-d @.\example.json ↩
http://<integrationServerHostName>:5555/restv2/wmentirex/extractIdl
```

Response Codes

The HTTP response is in JSON format.

- **Successful Request**
  If a request is successful, the following parameters are returned and `error` is empty. Example:

```
{
     ↩
"idlSourceBase64":"LyogR2VuZXJhdGVkIGJ5IFNvZnR3YXJlIEFHLCBJREwgRW5OaXJlWCAuLi4=",
     "mapToString":"false",
     "infos":[
          "Connection folder3.folder4:connection1 created"
     ],
     "error":null
}
```

where `idlSourceBase64` is the Base64-encoded source,

`mapToString`	indicates whether all data items were mapped to `String` (valid values: `true`, `false`), and
`infos`	is an array of status information, for example `Connection abc:def created`.

The HTTP response code is 200 (OK).

- **Unsuccessful Request**
  If a request is not successful, the `error` parameter contains the reason for the failure. Example:

```
{
  "idlSourceBase64": null,
  "info": null,
  "error": "Required parameter connectionName is missing"
}
```

The HTTP response code is 500 (internal server error).

- **Invalid Syntax**
  If the JSON input file has a syntax error (invalid input), the error parameter contains the reason for the JSON parser failure:

```
{
  "idlSourceBase64": null,
  "error": {
  "$errorDump": "com.fasterxml.jackson.core.JsonParseException:Unexpected character
  (';' (code 59)): was expecting comma to separate Object entries\n at [Source:
  (com.wm.net.HttpInputStream); line: 1, column: 36]",
...
}
```

The HTTP response code is 400 (bad request).

18 Creating or Updating Connections using the REST API

Introduction

You can create or update connections, adapter services and listeners, using the REST API.

To create a connection, adapter service or listener

- Send the request to the Integration Server to the following URL:

  ```
  http://hostname:port/restv2/wmentirex/createConnection
  ```

  Example:

  ```
  http://localhost:5555/restv2/wmentirex/createConnection
  ```

To update a connection, adapter service or listener

- Send the request to the Integration Server to the following URL:

  ```
  http://hostname:port/restv2/wmentirex/updateConnection
  ```

  Example:

  ```
  http://localhost:5555/restv2/wmentirex/updateConnection
  ```

Configuring your Input Parameters

To configure your input parameters

- Use a JSON request document. Parameters valid for all connection types and parameters for specific connection types are described in the sections below.

  Sample JSON file *createConnection.json*:

```
{
     "connectionType": "RpcConnection",
     "folderName": "rpcConnection",
     "connectionName": "RpcConnection",
     "packageName": "Default",
     "brokerID": "localhost:1971",
     "serverAddress": "RPC/Test/CALLNAT",
     ↩
"idlSourceBase64":"KioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKi
                          ↩
oqKioqKioqKioqKioqKioqDQoqKgOKKiogVGhpcyBJREwgZmlsZSBjb25OYWlucyBOaGUg
                          ↩
SURMIGRlZmluaXRpb25zIGZvciBOaGUNCioqIEVudGlyZVggYmFzaWMgUlBDIGV4YW1wbG
                          ↩
UuDQoqKgOKKiogQ29weXJpZ2hOIChjKSAxOTk3ICOgMjAxOSBTb2ZOd2FyZSBBRywgRGFy
                          ↩
bXNOYWROLCBHZXJtYW55IGFuZC9vciANCioqICBTb2ZOd2FyZSBBRyBVUOEsIEluYy4sIF
                          ↩
Jlc3RvbiwgVkEsIFVuaXRlZCBTdGFOZXMgb2YgQW1lcmljYSwgDQoqKiAgYW5kL29yIHRo
                          ↩
ZWlyIGxpY2Vuc29ycy4NCioqDQoqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKi
                          ↩
oqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioNCgOKTGlicmFyeSAnRVhB
                          ↩
TVBMRScgSXMNClByb2dyYWOgJONBTEMnIElzDQpEZWZpbmUgRGFOYSBQYXJhbWVOZXINCi
                          ↩
AgICAgIDEgT3BlcmFOaW9uICAgICAgIChBMSkgIEluDQogICAgICAxIE9wZXJhbmQxICAg
                          ↩
ICAgICAoSTQpICBJbgOKICAgICAgMSBPcGVyYW5kMiAgICAgICAgKEkOKSAgSW4NCiAgIC
                          ↩
AgIDEgRnVuY3Rpb25fUmVzdWxOIChJNCkgIE91dAOKICAgIEVuZC1EZWZpbmUNCgOKICBQ
                          ↩
cm9ncmFtICdTUVVBUkUnIElzDQogICAgRGVmaW5lIERhdGEgUGFyYW1ldGVyDQogICAgIC
                          ↩
AxIE9wZXJhbmQgICAgICAgICAoSTQpICBJbgOKICAgICAgMSBSZXN1bHQgICAgICAgICAg
                          KEkOKSAgT3VODQogICAgRW5kLURlZmluZQkgIA=="
}
```

Connection Parameters for All Connection Types

The following parameters apply to all connection types:

Parameter	Opt/ Req	Description		Note
`connectionType`	R	Must be one of the following connection types:		See *EntireX Adapter Connections* for a detailed description of all connection types.
		`RpcConnection`	RPC Connection	
		`DirectRpcConnection`	Direct RPC Connection	
		`ReliableRpcConnection`	Reliable RPC Connection	
		`RpcListenerConnection`	RPC Listener Connection	
		`DirectRpcListenerConnection`	Direct RPC Listener Connection	
		`ReliableRpcListenerConnection`	Reliable RPC Listener Connection	
		`DirectReliableRpcListenerConnection`	Direct Reliable RPC Listener Connection	
		`ImsConnection`	IMS Connect Connection	
		`CicsConnection`	CICS ECI Connection	
		`CicsSocketListenerConnection`	CICS Socket Listener Connection	
		`ACIConnection`	ACI Server Connection	

Parameter	Opt/ Req	Description	Note
		`CobolConverterConnection` COBOL Converter Connection `AS400Connection` AS/400 Connection `ApplinXConnection` ApplinX Connection	
`packageName`	R	The package must exist in the Integration Server.	
`folderName`	R	If the folder does not exist, it will be created.	
`connectionName`	R	The name of the connection to be created.	
`idlSourceBase64`	R	The Base64-encoded IDL source.	See ***Base64 Resources*** for information on Base64 encoding/decoding tools.
`serverMappingBase64`	O	The Base64-encoded CVM source.	
`mapToString`	O	Map all IS data types to string. Valid values: `true` (default), `false`.	
`resourceName`	O	Name of a REST resource to create in the format `"folderName:RESTResourceName"`.	Only for connections with adapter services. See *Step 4b: Create or Update a REST Resource* in the Integration Server Wrapper documentation.

Connection Parameters for RPC and Reliable RPC Connections

Note: Parameters marked R are required when you create a connection and optional for updates.

Parameter		Opt/ Req	Description
`brokerID`	Broker ID	R	The ID of the broker you want to connect to. This ID consists of a host and an optional port. Default for the port is "1971".
`serverAddress`	Server Address	R	The address of the RPC server registered to the broker above. The address is given in the format `<class>/<server>/<service>`.
`userid`	Logon User	O	The name of the user to log on to the broker.
`password`	Logon Password	O	The password for the user above.
`encoding`	Encoding	O	The character encoding used for the RPC connection to the *EntireX Broker*. Default: the encoding of the Integration Server. Enable character conversion in the broker by setting the service-specific attribute `CONVERSION` to "`SAGTRPC`". See also *Configuring ICU Conversion* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation. More information can be found under *Internationalization with EntireX*.

Connection Parameters for RPC and Reliable RPC Listener Connections

Note: Parameters marked R are required when you create a connection and optional for updates.

Parameter		Opt/ Req	Description
`brokerID`	Broker ID	R	The ID of the broker you want to connect to. This ID consists of a host and an optional port. Default for the port is "1971".
`serverAddress`	Server Address	R	The address of the RPC server registered to the broker above. The address is given in the format `<class>/<server>/<service>`.
`userid`	Logon User	O	The name of the user to log on to the broker.
`password`	Logon Password	O	The password for the user above.

Parameter		Opt/ Req	Description
`encoding`	Encoding	O	The character encoding used for the RPC connection to the *EntireX Broker*. Default: the encoding of the Integration Server. Enable character conversion in the broker by setting the service-specific attribute `CONVERSION` to "`SAGTRPC`". See also *Configuring ICU Conversion* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation. More information can be found under *Internationalization with EntireX*.
`listenerName`	Listener Name	R	Name of the generated listener object.
`serviceNames`	Service Names	O	Comma-separated list of IS services. ■ If *not* set, service names will be generated as: `<library name>:<program name>Service`. Example: Library `EXAMPLE` and subprogram `CALC` will be generated to service `EXAMPLE:CALCService`. ■ If set, the number of elements must be identical to the number of subprograms in the library.

Connection Parameters for Direct RPC Connections

Note: Parameters marked R are required when you create a connection and optional for updates.

Parameter		Opt/ Req	Description
`serverAddress`	Server Address	R	The address of the RPC server registered to the broker above. The address is given in the format `<class>/<server>/<service>`.

Connection Parameters for Direct RPC and Direct Reliable RPC Listener Connections

Note: Parameters marked R are required when you create a connection and optional for updates.

Parameter		Opt/ Req	Description
`serverAddress`	Server Address	R	The address of the RPC server registered to the broker above. The address is given in the format `<class>/<server>/<service>`.
`serviceNames`	Service Names	O	Comma-separated list of IS services. ■ If *not* set, service names will be generated as: `<library name>:<program name>Service`. Example: Library `EXAMPLE` and subprogram `CALC` will be generated to service `EXAMPLE:CALCService`. ■ If set, the number of elements must be identical to the number of subprograms in the library.

Connection Parameters for Connections to IMS Connect

Note: Parameters marked R are required when you create a connection and optional for updates.

Parameter		Opt/ Req	Description
`host`	Host	R	Hostname of IMS Connect.
`port`	Port	R	IMS Connect port.
`datastoreid`	IMS Connect Data Store ID	R	Name of the data store, as defined in the IMS Connect configuration member.
`encoding`	Encoding	O	Specify the appropriate EBCDIC encoding used by your IMS Connect.

Connection Parameters for COBOL Converter Connections

Parameter		Opt/ Req	Description
`encoding`	Encoding	O	The character encoding of the COBOL binary data.

Connection Parameters for CICS Socket Listener Connections

Note: Parameters marked R are required when you create a connection and optional for updates.

Parameter		Opt/Req	Description
`host`	Host	R	Hostname of CICS Socket Listener.
`port`	Port	R	CICS Socket Listener Port.
`transactionName`	CICS Transaction ID	R	Name of the CICS transaction. Default is "XRFE", which is the default dispatching transaction for CICS Socket Listener.
`encoding`	Encoding	O	Specify the appropriate EBCDIC encoding used by your CICS Socket Listener.
`userid`	RACF User ID	O	The name of the user to log on to CICS Socket Listener.
`password`	Password	O	The password for the user above.

Connection Parameters for CICS ECI Connections

Note: Parameters marked R are required when you create a connection and optional for updates.

Parameter		Opt/Req	Description
`host`	Host	R	Hostname of CICS.
`port`	Port	R	CICS port.
`transactionName`	CICS Mirror Transaction ID	R	Name of the CICS mirror transaction. Default is "CPMI", which is the default dispatching transaction for ECI.
`encoding`	Encoding	O	Specify the appropriate EBCDIC encoding used by your CICS ECI.
`userid`	RACF User ID	O	The name of the RACF user to log on to CICS ECI.
`password`	RACF Password	O	The password for the user above.

Connection Parameters for ACI Server Connections

Note: Parameters marked R are required when you create a connection and optional for updates.

Parameter		Opt/ Req	Description
`brokerID`	Broker ID	R	The ID of the broker you want to connect to. This ID consists of a host and an optional port. Default for the port is "1971".
`serverAddress`	Server Address	R	The address of the RPC server registered to the broker above. The address is given in the format `<class>/<server>/<service>`. The address may contain an asterix ('*') as a wildcard that is substituted by the IDL program name at runtime. This helps you to use only one connection for multiple IDL programs or adapter services.
`userid`	Logon User	O	The name of the user to logon on to the broker.
`password`	Logon Password	O	The password for the user above.
`encoding`	Encoding	O	The character encoding used for the ACI connection to the EntireX Broker. Default: the encoding of the Integration Server. Enable character conversion in the broker by setting the service-specific attribute `CONVERSION` to "`SAGTCHA`". See also *Configuring ICU Conversion* under *Configuring Broker for Internationalization* in the platform-specific Administration documentation. More information can be found under *Internationalization with EntireX*.

Connection Parameters for AS/400 Connections

Note: Parameters marked R are required when you create a connection and optional for updates.

Parameter		Opt/ Req	Description
`host`	Host	R	Hostname of the AS/400 (IBM i) system.
`encoding`	Encoding	O	Specify the appropriate EBCDIC encoding which corresponds to the CCSID (Coded Character Set Identifier) of your AS/400 system.
`userid`	User ID	O	The user profile name to use to authenticate to the system.
`password`	Password	O	The user profile password to use to authenticate to the system.

Connection Parameters for ApplinX Connections

Note: Parameters marked R are required when you create a connection and optional for updates.

Parameter		Opt/ Req	Description
`host`	Host	R	Hostname of ApplinX server.
`port`	Port	R	Port of ApplinX server.

Response Codes

The HTTP response is in JSON format and contains two parameters, `infos` and `error`.

- **Successful Request**
  If a request is successful, `infos` contains a comma-separated list of objects that were created or updated, and the `error` parameter is empty. Example:

```
{
  "infos": [
    "Connection rpc:RpcConnection created",
    "Adapter Service rpc:CALC created",
    "Adapter Service rpc:SQUARE created"
  ],
  "error": null
}
```

  The HTTP response code is 200 (OK).

- **Unsuccessful Request**
  If a request is not successful, the `infos` parameter is empty, and the `error` parameter contains the reason for the failure. Example:

```
{
  "infos": null,
  "error": "Required parameter connectionName is missing"
}
```

  The HTTP response code is 500 (internal server error).

- **Invalid Syntax**
  If the JSON input file has a syntax error (invalid input), the error parameter contains the reason for the JSON parser failure:

```
{
  "idlSourceBase64": null,
  "error": {
  "$errorDump": "com.fasterxml.jackson.core.JsonParseException:Unexpected character
  (';' (code 59)): was expecting comma to separate Object entries\n at [Source:
  (com.wm.net.HttpInputStream); line: 1, column: 36]",
...
}
```

  The HTTP response code is 400 (bad request).

Base64 Resources

Base64 encoding/decoding is used to keep the source file formatting inside the JSON format. Base64 encoding/decoding tools are available for several environments, for example:

- **Integration Server**
  `pub.string:base64Encode` and
  `pub.string:base64Decode` in package `WmPublic`
- **Java**
  Class `java.util.Base64` (nested classes `java.util.Base64.Encoder` and `java.util.Base64.Decoder`)
- **Online Tools**
  - **https://www.base64encode.org**
  - **https://www.base64decode.org**

"

19 Post-installation Steps for AS/400

After installation of the EntireX Adapter, the AS/400 connection type is not visible and not usable. To access the AS/400 system you need the IBM ToolBox for Java (JTOpen).

To download and install the IBM ToolBox for Java

1 From **https://sourceforge.net/projects/jt400/**, click **Downloads**. From **https://sourceforge.net/projects/jt400/**, select the latest version.

2 From the downloaded zip file, extract the file *lib/java8/jt400.jar*.

3 Copy the file *jt400.jar* to directory *IntegrationServer/instances/<instance_name>/packages/WmEntireX/code/jars*.

4 Reload the WmEntireX package.

Or:

Restart the Integration Server instance.