TIBCO iWay® Service Manager

Protocol Guide

Version 8.0 and Higher
March 2021
DN3501359.0321

Contents

Chapter 1

iWay Protocol Adapters

This chapter introduces the various protocols supported by iWay Service Manager.

In this chapter:

Protocols Supported in iWay Service Manager

Protocols define how information is physically controlled during its movement through a process. This is different from format, which describes how data is stored and represents information. Protocol adapters are iWay adapters that enable messages to be accepted from and emitted onto transports such as files, HTTP, and IBM MQ Series, regardless of the meaning of the data carried by the message. iWay Protocol adapters enable iWay services to be applied to arbitrary messages regardless of the transport on which the messages move.

Protocol adapters are coupled with transformation EIS and format adapters to provide meaningful adapter services across protocols. Almost all integration activities require transformation from source format to target format. For process integration, this may require transformation between different API or message formats, whereas in data integration, the transformations are between different data repository schema and semantic definitions.

The base transformation process involves format modifications (both syntax and semantic mapping). Often more sophisticated transformation processes like message augmentation (or boosting) and data cleansing are required to handle the idiosyncrasies of application implementations. For more information on transformation capability, see the *iWay Transformer User's Guide*.

Message handlers deal with the discovery (parsing) of the in-bound message content, the optional conversion to a common internal format, and building the format expected by the target systems. Current implementations use XML as the internal format and increasingly, use XML for the out-bound message format.

Message routing is a key component of the Enterprise Application Integration (EAI) solution and includes the following:

- Managing the delivery of the messages over communication connections.

  Includes protocol conversion, flow control, guaranteed delivery, and connection optimization (for example, connection pooling).

- Multi-point message decomposition/recomposition.

  Enables one-to-many and many-to-one message routing.

- Routing definition for content-based routing.

  Includes associated directory-based or rules-based routing capabilities.

The following is a list of the available iWay Protocol adapters.

e-Business Adapters

- AS1, AS2 (EDIINT)
- Non-blocking AS2 (NAS2)

Application Adapters

- EDAAPI (CS3)
- RDBMS
- MySAP
- Schedule
- LDAP
- LDAP High Water Mark
- Relational Database High Water Mark
- Salesforce
- IEI
- Log Event

Transport Utility Adapters

- Email
- File
- File Transfer Protocol (FTP)
- SFTP
- FTP Server

- HTTP
- nHTTP
- SOAP
- TCP
- Telnet
- RVI

Queuing Adapters

- JMS
- MSMQ
- Oracle Advanced Queuing (AQ)
- Sonic MQ
- TIBCO Rendezvous
- RabbitMQ
- WebSphere MQ
- WebSphere MQJMS
- Internal and Ordered

Chapter 2

Configuring Channels

The concept of a channel has been introduced to assist in the construction of message flows in iWay Service Manager.

A channel serves as a container for all your components, simplifying the integration design process and improving organization, versioning, and troubleshooting. It contains the following conduits, which must be configured and associated with the channel.

- **Inlet.** Defines how a message enters a channel, along any number of protocols.
- **Route.** Defines the path a message takes through a channel, allowing you to apply business logic to the message.
- **Outlet.** Defines how a message leaves a channel, along any number of protocols.

In this chapter:

Defining an Inlet

Each inlet contains a sequence of listeners, optional decryptors, and optional preparsers.

Listeners are defined as protocol handlers and are responsible for startup, shutdown, and obtaining the incoming messages. Listeners receive the messages from the transport protocol, set special registers, such as header values and input source, and then pass the message to a decryptor.

Decryptors can apply a decryption algorithm to the incoming message and verify the security of the message. A decryptor can be used to verify that the sender is authorized, to check that the message has not been changed, and to decrypt any part of the message that has been encrypted. Finally the decryptor passes the message to a preparser.

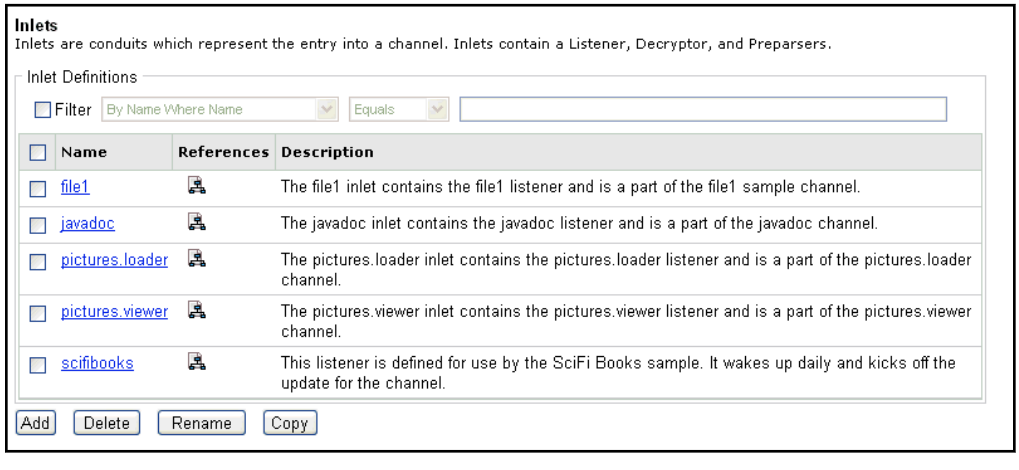
Preparsers convert transported messages into processable documents. For example, some preparsers convert non-XML messages into XML documents. Preparsers can be chained, so that the output of one preparser becomes the input of the next preparser. Input to the first preparser is a byte stream, and output is a properly encoded string. Subsequent preparsers accept and emit strings.

Procedure: How to Define an Inlet

To define an inlet using the iWay Service Manager Administration Console:

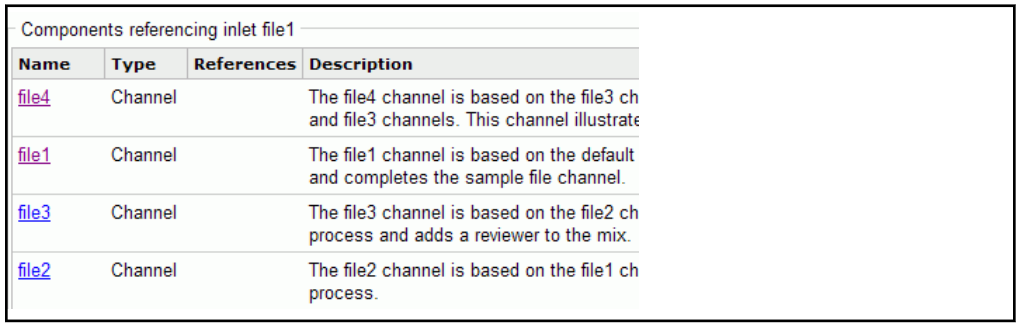
1. In the left console pane of the Registry menu, select *Inlets*.

The Inlets pane opens, as shown in the following image.



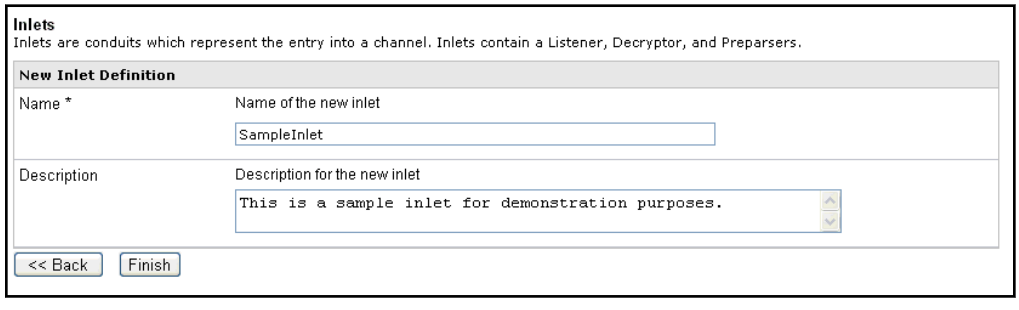
The table that is provided lists each inlet that is defined with a brief description. If you click the document schematic icon in the References column for a specific inlet, you will see which components are referencing that inlet.

The following image shows the result of clicking the schematic icon for the file1 inlet:



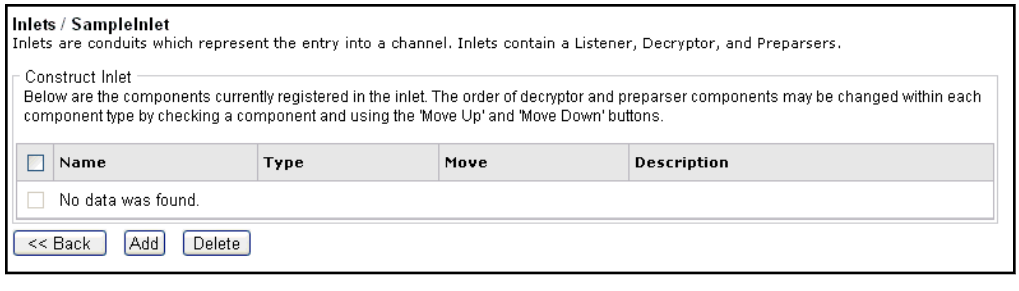
2. Click *Add*.

The New Inlet Definition pane opens, as shown in the following image.



3. Enter a name, for example, SampleInlet, and description for the inlet.
4. Click *Finish*.

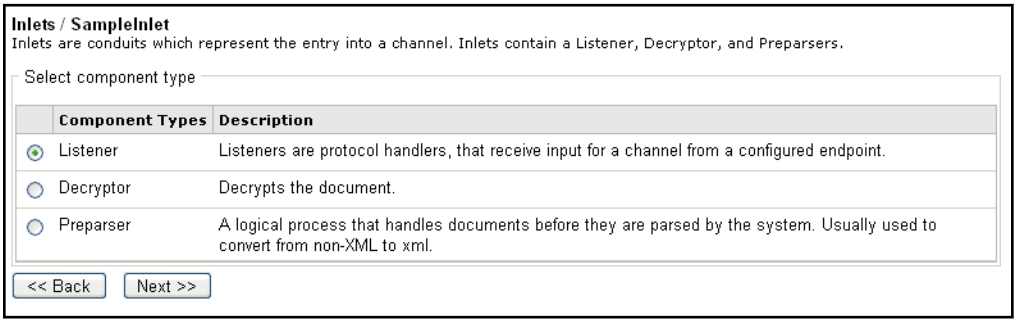
The Construct Inlet pane opens, as shown in the following image.



The table that is provided is used to list the components that are currently registered with the inlet.

5. Click *Add*.

   The Select component type pane opens, as shown in the following image.



   The table that is provided lists the component types you can select and register with the inlet you are defining.

   - **Listener.** Protocol handlers that receive input for a channel from a configured endpoint.
   - **Decryptor.** Used to decrypt a document.
   - **Preparser.** A logical process that handles documents before they are parsed by the system, for example, converting a non-XML document to XML.

   Note: Each inlet is required to have a registered listener. The remaining components are optional during inlet configuration.

Configuring Listeners

Service Manager allows you to configure listeners for many different protocols. The following procedures describe the steps required for creating a listener, using a file listener as an example, and the steps required to assign a listener to an inlet. After creating a listener, you must assign it to an Inlet, which you then assign to a channel.

The inlet to which you want to add the listener must be created before you assign a listener to it.

For more information on creating and configuring inlets and creating and configuring channels, see the *iWay Service Manager User's Guide*.

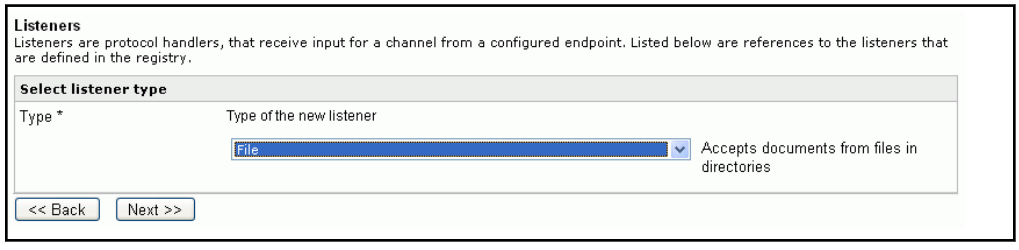
Procedure: How to Create a Listener

To create a listener:

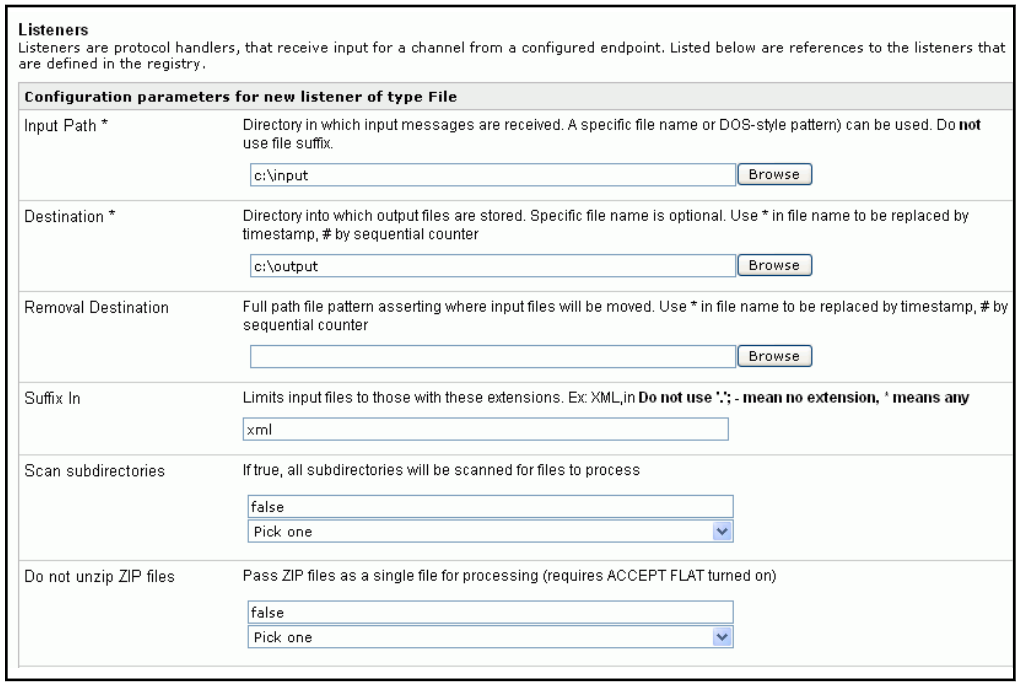
1. On the iWay Service Manager console home page, click *Registry*.

2. In the left pane of the console, click *Listeners* under Components.
3. In the Listeners pane that opens, click *Add*.

   The pane prompts you for the type of listener, as shown in the following image.

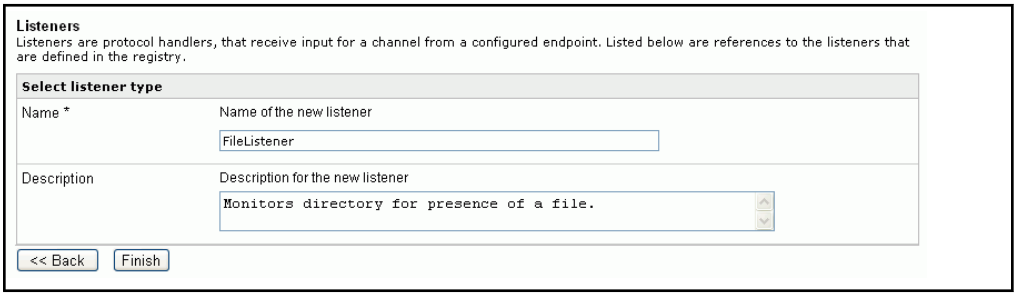


4. From the Select a type drop-down list, click the type of listener you want to create, for example, File, and then click *Next*.
   The configuration window for defining the properties of the new listener opens, as shown in the following image for a File listener. A property for which you must supply a value has an asterisk (*) next to it.

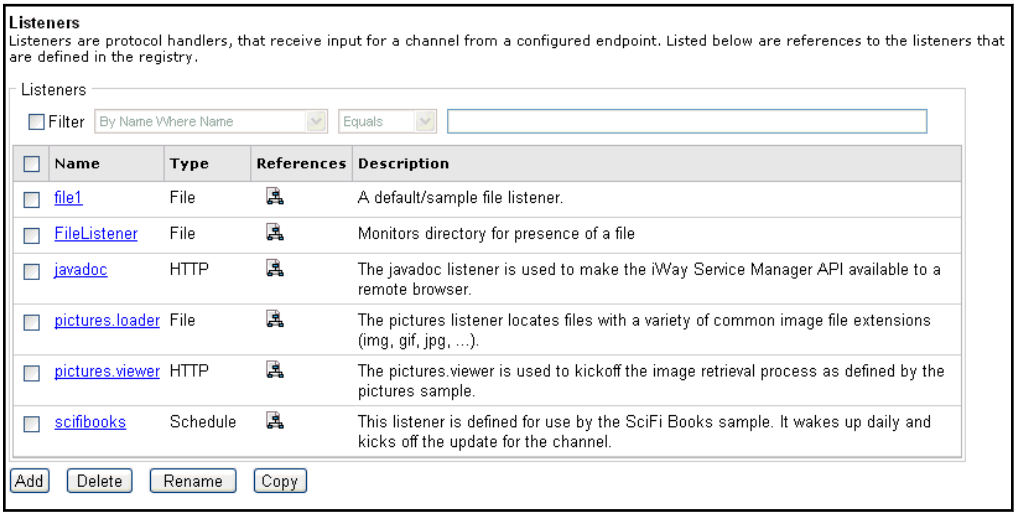


5. Enter or select values for the properties to describe the listener.
6. Click *Next* when you have finished entering values on this pane.

On the next pane, as shown in the following image, you are prompted to provide a name (required) and description for the new listener.



7. Enter a name and optional description then click *Finish*.
The listener is added, as shown in the following image.



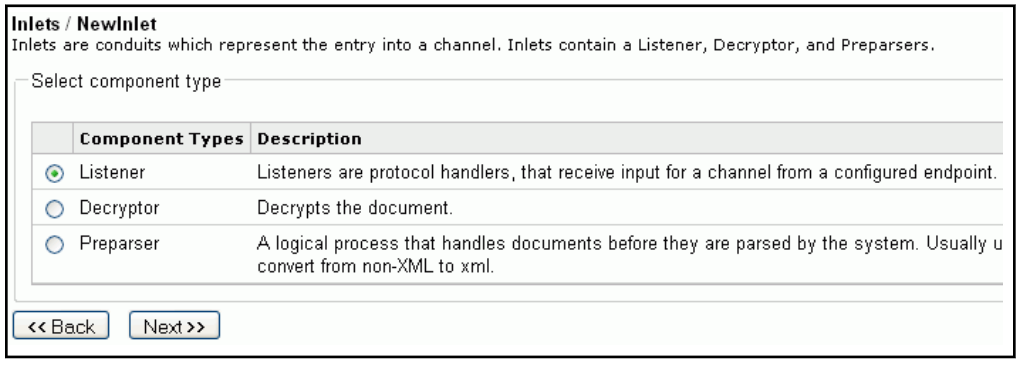
Procedure: How to Assign a Listener to an Inlet

1. On the Service Manager console, click *Registry*.
2. In the left pane, click *Inlets*.

   The Inlets pane appears on the right.
3. Click the name of the inlet to which you want to add the listener.

   The Construct Inlet pane appears.
4. Click *Add*.

The Select component type screen appears, as shown in the following image.



5. Select *Listener* and click *Next*.

   The Select a listener definition pane appears.

6. Select the listener you configured, and click *Finish*.

Defining a Route

Routes contain references to transforms, processes, and outlets. A route describes the path that a document takes during its passage through the system, after the inlet converts the input message to a document that can be processed. Multiple routes can be defined for the same channel if desired.

Transforms convert the document information to a common format suitable for general business processing. For example, similar messages from two trading partners might differ slightly in format; a common format is often desirable for business processing. Transforms are constructed using iWay Transformer. For more information, see the *iWay Transformer User's Guide*.

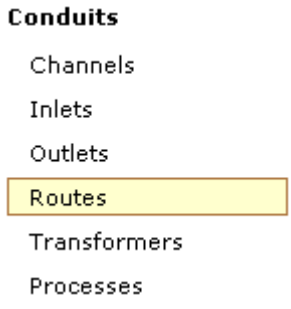
Processes perform the actual business operations on the document. A business process is composed of one or more services, with appropriate switching, testing, iteration, and error handling. Processes can call on other processes and web services, and in turn can be packaged as web services for external consumption. Business processes are constructed using iWay Designer. For more information, see the *iWay Designer User's Guide*.

Outlets pass the processed document to one or more designated recipients. They convert the document to a transport format and then emit the message.

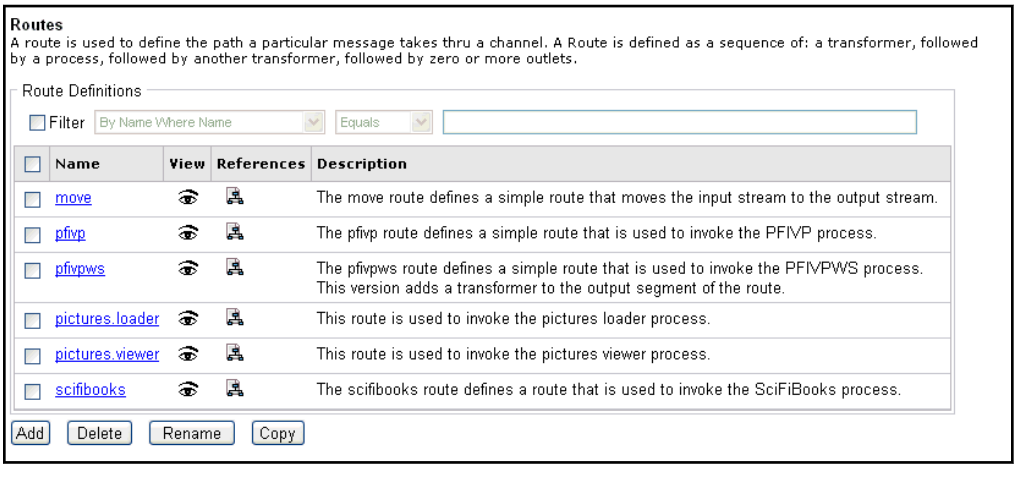
Procedure: How to Define a Route

To define a route using the iWay Service Manager Administration Console:

1. In the left console pane of the Registry menu, select *Routes*, as shown in the following image.



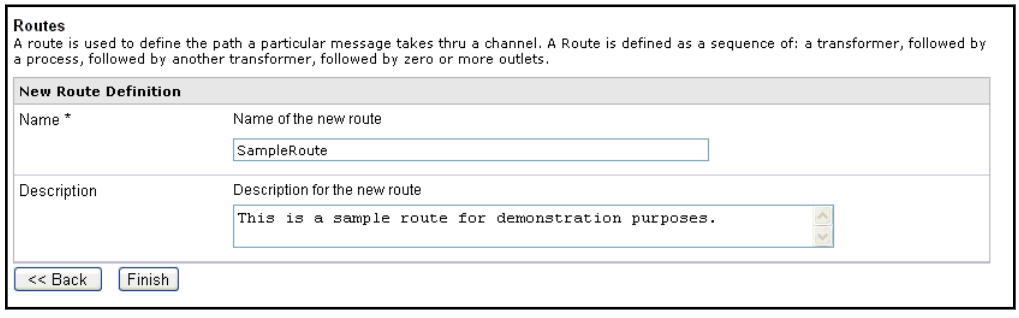
The Routes pane opens, as shown in the following image.



The table that is provided lists each route that is defined with a brief description. If you click the document schematic icon in the References column for a specific route, you will see which components are referencing that route. (If you click the eye icon in the View column for a specific route, and then the process icon between the two arrows, you will see a visual depiction of that route.)

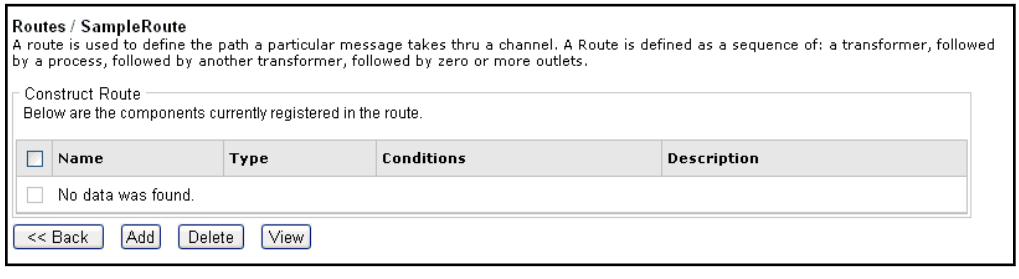
2. Click *Add*.

The New Route Definition pane opens, as shown in the following image.



3. Enter a name, for example, SampleRoute, and description for the route.
4. Click *Finish*.

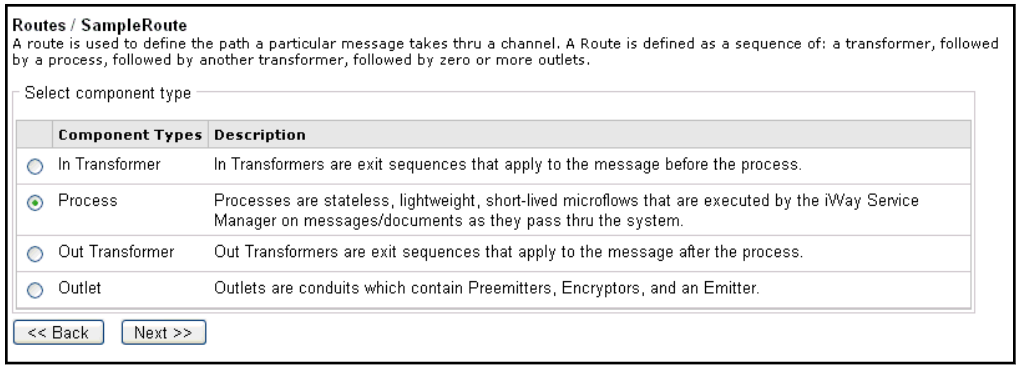
   The Construct Route pane opens, as shown in the following image.



   The table that is provided is used to list the components that are currently registered with the route.

5. Click *Add*.

   The Select component type pane opens, as shown in the following image.



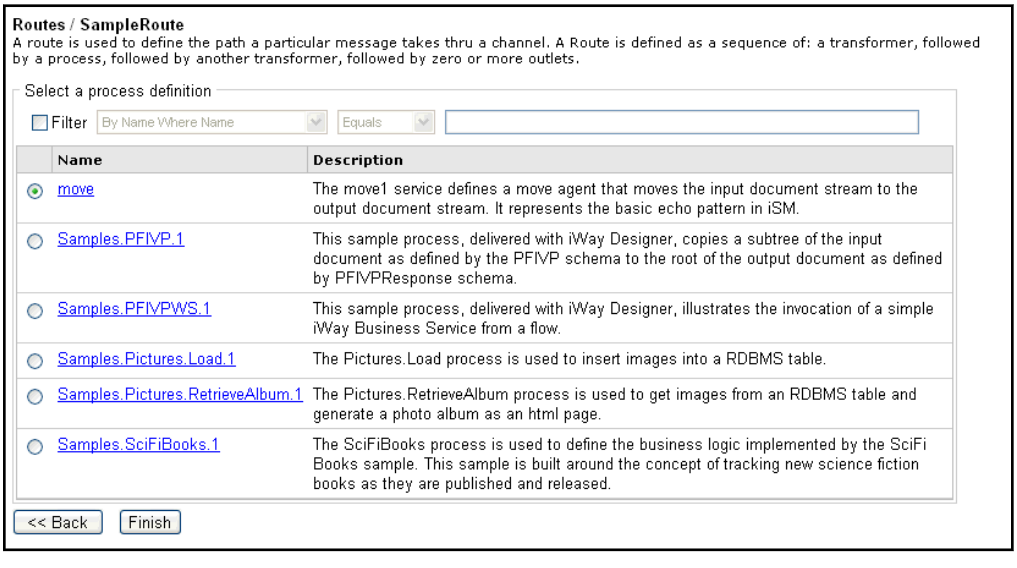
The table that is provided lists the component types you can select and register with the route you are defining.

- **In Transformer.** Exit sequences that apply to a message before processing occurs. For more information on creating transformations, see the *iWay Transformer User's Guide*.
- **Process.** Stateless, lightweight, and short-lived microflows that are executed by iWay Service Manager on messages and documents as they pass through the system. The simplest process contains a move service, which is first placed into a flow then added to a process. For more information on process flows, see the *iWay Designer User's Guide*.
- **Out Transformer.** Exit sequences that apply to the message after processing occurs. For more information on creating transformations, see the *iWay Transformer User's Guide*.
- **Outlet.** Conduits that consist of Preemitters, Encryptors, and Emitters. For more information, see the *iWay Service Manager User's Guide*.

Note: Each route that is being defined is required to have a registered process. The remaining components are optional during route configuration.

6. Select *Process* from the list of component types and click *Next*.

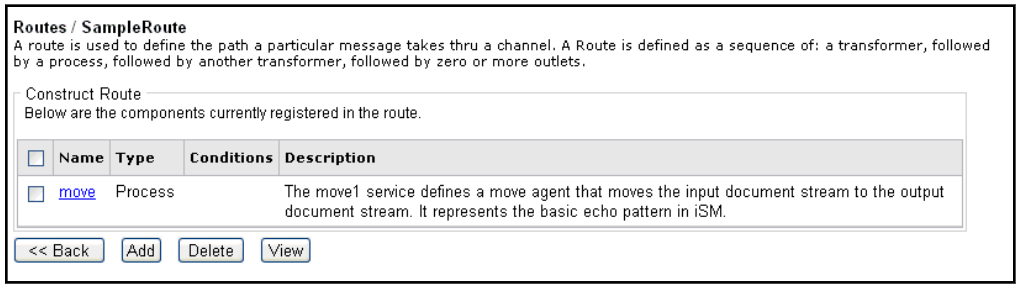
   The Select a process definition pane opens, as shown in the following image.



The table that is provided lists existing process flows you can select for the route you are defining.

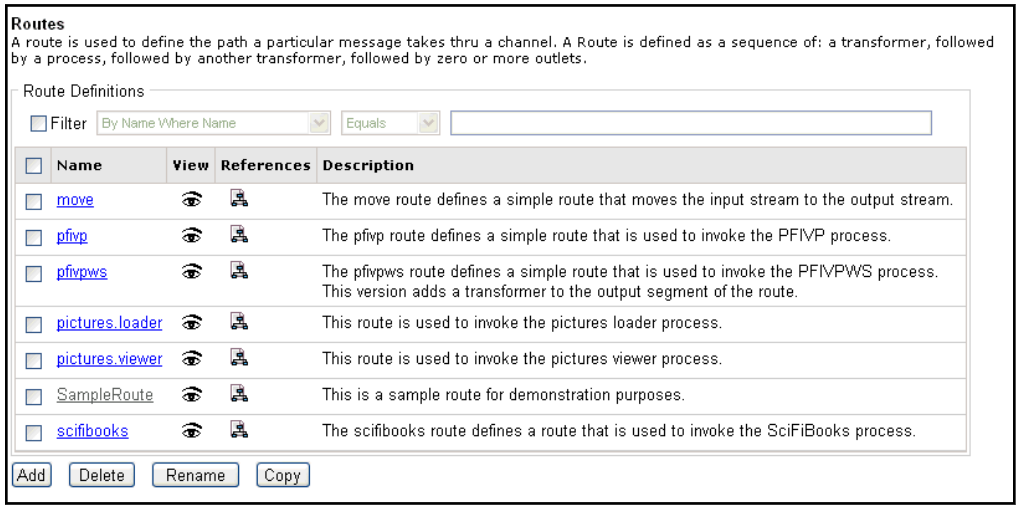
7. Select *move* and click *Finish*.

   As shown in the following image, you are returned to the Construct Route pane, which now includes the process (move) you registered with your route (SampleRoute).



   You can now add additional components, such as a transformer or an outlet.

   If you return to the main Routes pane, you will notice that the route you just defined (SampleRoute) has been added to the list, as shown in the following image.



Defining an Outlet

Outlets contain references to preemitters, encryptors, and emitters. Once a document has been processed, it must be sent to one or more designated recipients. This is the job of the outlet. The outlet is responsible for all aspects of preparing the document for emission and then emitting it.

Outlets contain a sequence of components tailored for this task. Multiple outlets can be configured for a single message flow. Each outlet incorporates all of the components needed to send the message to its destination.

A channel must contain an outlet. However, a default outlet, which contains no emitter, can be used. When you assign an empty outlet to a channel, the document output goes back to the listener assigned to the inlet and is emitted through whatever output is specified in the listener.

Preemitters convert the document from the internal format to an external format. This may include simply flattening XML or may be more complex, involving transformation logic. An example is converting the document to an EDI or HIPAA format. Preemitters can also be chained, so the output of one becomes the input to the next. The first preemitter receives the document in internal form, and transforms it to a message format. Subsequent preemitters can perform extra work on this message. Transformations are prepared using the iWay Transformer. For more information, see the *iWay Transformer User's Guide*.

Encryptors operate on the message that is ready for emitting. Parameters such as the location of encryption keys or certificate aliases can be stored by destination address in the optional iWay Trading Manager component. For more information, see the *iWay Trading Manager User's Guide*.

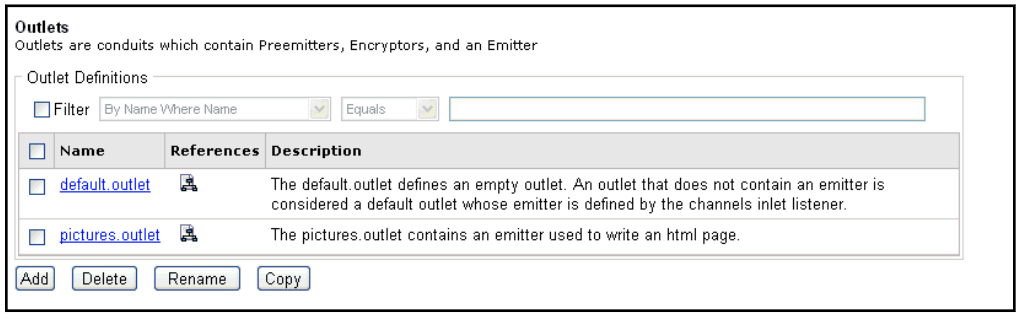
The emitter uses the appropriate transport protocol to send the document to its destination. Examples include JMS, HTTP, email, and AS2. Header information that has been prepared by processes and stored in special registers is applied to the message in a format-appropriate manner.

Procedure: How to Define an Outlet

To define an outlet using the iWay Service Manager Administration Console:

1. In the left console pane of the Registry menu, select *Outlets*, as shown in the following image.

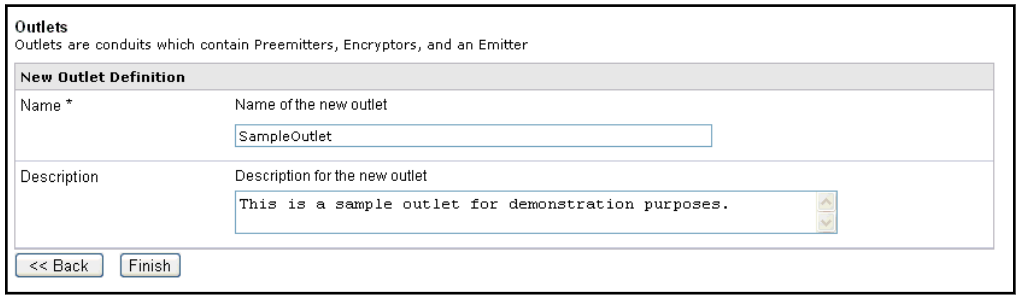
The Outlets pane opens, as shown in the following image.



The table that is provided lists each outlet that is defined with a brief description. If you click the document schematic icon in the References column for a specific outlet, you will see which components are referencing that outlet.

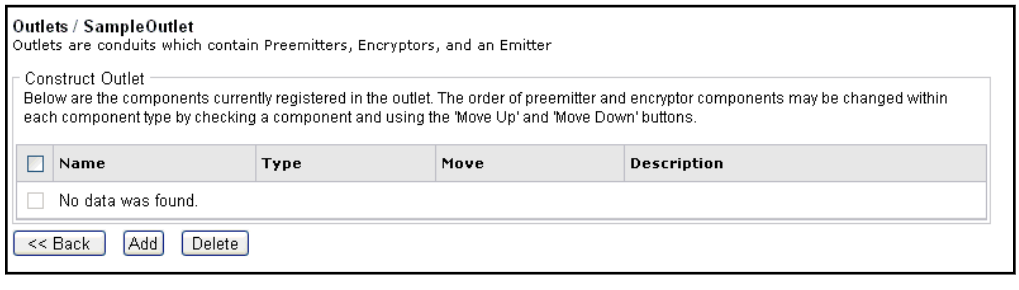
2. Click *Add*.

   The New Outlet Definition pane opens, as shown in the following image.



3. Enter a name, for example, SampleOutlet, and description for the outlet.
4. Click *Finish*.

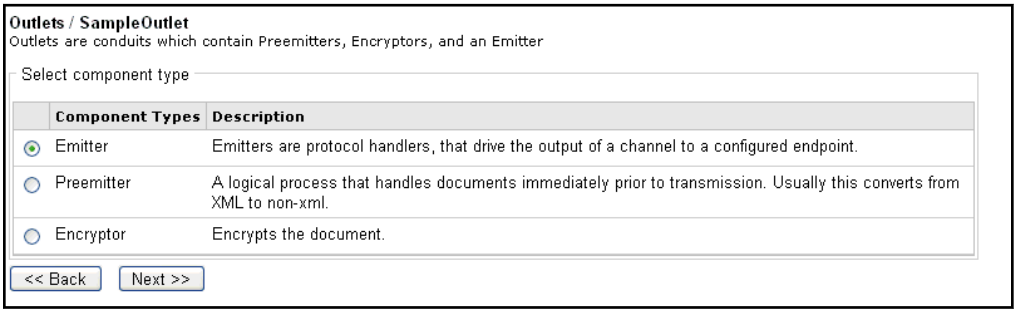
   The Construct Outlet pane opens, as shown in the following image.



The table that is provided is used to list the components that are currently registered with the outlet.

5. Click *Add*.

   The Select component type pane opens, as shown in the following image.



   The table that is provided lists the component types you can select and register with the outlet you are defining.

   - **Emitter.** Protocol handlers that send the output of a channel to a configured end point.
   - **Preemitter.** A logical process that handles documents immediately prior to transmission, for example, converting an XML document to non-XML.
   - **Encryptor.** Used to encrypt a document.

   An outlet that does not contain an emitter is considered a default outlet, whose emitter is defined by a channel's inlet listener. A default outlet defines an empty outlet.

Outlet Strategies

iWay Service Manager provides mechanisms to support various routing strategies when using outlets. This section describes how you can use iSM to execute business logic and route documents to a particular location. The following topics are provided:

- Adding conditions to outlets.
- Configuring run-time options for outlets (On Success, On Error).

Procedure: How to Add Conditions to Outlets

To add conditions to outlets:

1. Open the channel you want to edit.
2. Click the *add conditions*  icon in the outlet for which you want to set conditions.

   The Set Conditions pane appears.

3. Provide the condition, and then click *Back* to return to the channel pane.

   The icon changes to the Edit  icon to indicate that a condition has been set.

   If multiple outlets are defined for a channel and no conditions are added to the outlets, all of the outlets will be processed by iWay Service Manager.

Procedure: How to Configure Outlet Run-Time Options

To configure outlet run-time options:

1. Open the channel you want to edit.
2. Click the *check*

   icon for the outlet you want to modify to set the run time option from On Success to On Error  or the other way around, as appropriate.

Configuring Emitters

You can configure Service Manager to emit messages over many different protocols. After you configure the emitter, you assign it to an Outlet, which you then incorporate into a channel. The procedures in this section list the steps required to configure an emitter, using the file emitter as an example, and to assign an emitter to an outlet.

The outlet to which you want to add the emitter must be created before you assign an emitter to it.

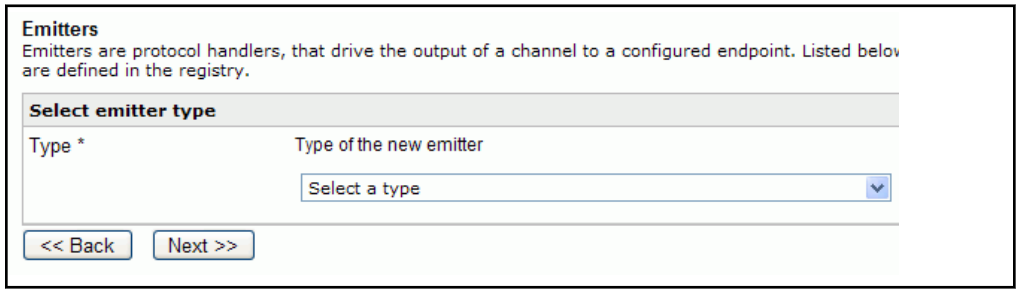
For more information on creating and configuring outlets and creating and configuring channels, see the *iWay Service Manager User's Guide*.

Procedure: How to Create an Emitter

To create an emitter:

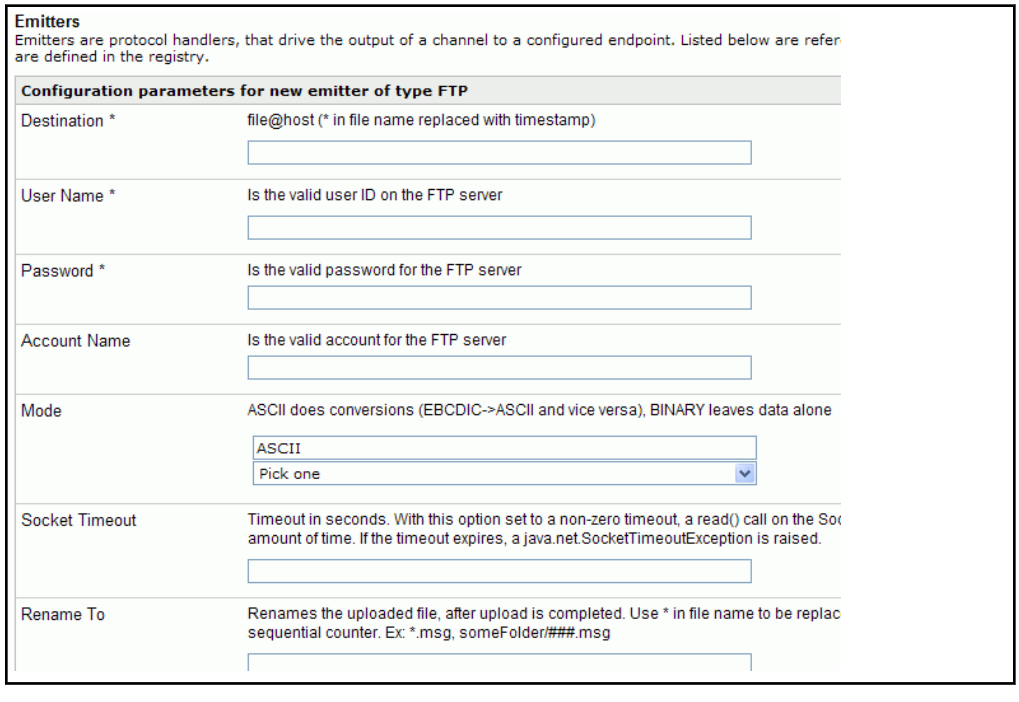
1. On the iWay Service Manager console home page, click *Registry*.
2. In the left pane of the console, under Components, click *Emitters*.
3. In the Emitters pane that opens on the right, click *Add*.

The pane prompts you for the type of emitter, as shown in the following image.



4. From the Select a type drop-down list, click the type of emitter you want to create, for example, FTP, and then click *Next*.

   The configuration window for the defining properties of the new emitter opens, as shown in the following image for the FTP emitter. An asterisk (*) denotes a required property.

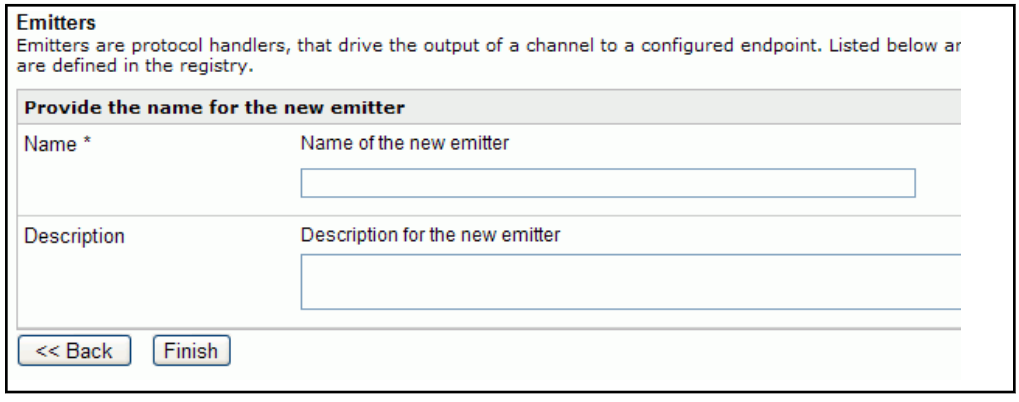


5. Enter or select values for the properties to describe the emitter.

   For more information on configuring an FTP emitter, see *FTP* on page 174.

6. Click *Next* when you have finished entering values on this pane.

On the next pane, you are prompted to provide a name (required) and description for the new emitter, as shown in the following image.



7. Enter a name and optional description and click *Finish*.

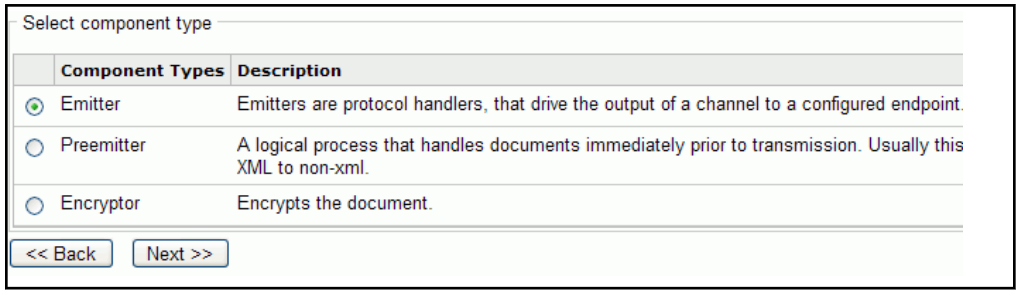
Procedure: How to Assign an Emitter to an Outlet

1. On the Service Manager console, click *Registry*.
2. In the left pane, click *Outlets*.

   The Outlets pane appears on the right.
3. Click the name of the outlet to which you want to add the emitter. For example, default_outlet.

   The Construct outlet pane appears.
4. Click *Add*.

   The Select component type screen appears, as shown in the following image.



5. Select *Emitter* and click *Next*.

   The select an emitter definition pane appears.

6. Select the emitter you configured, and click *Finish*.

Constructing a Channel

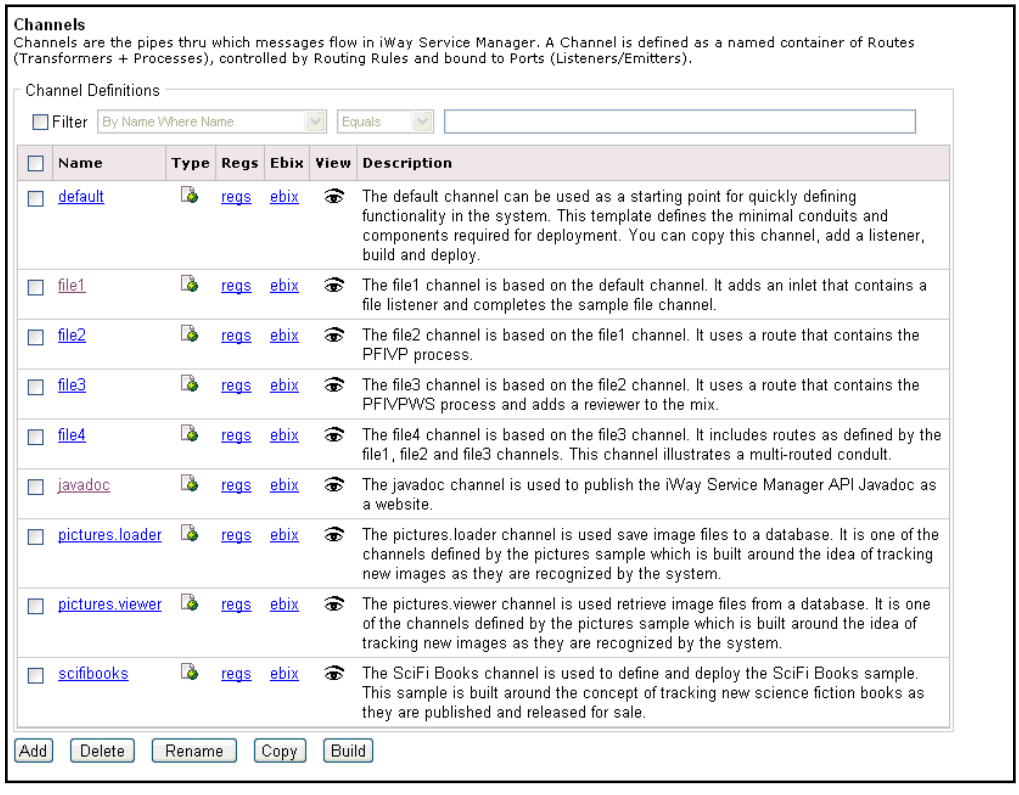
After you have defined the necessary channel components (inlet, route, and outlet), you can combine these components and construct a channel using the iWay Service Manager Administration Console. Every channel is required to have an inlet, a route, and an outlet.

Procedure: How to Construct a Channel

To construct a channel using the iWay Service Manager Administration Console:

1. In the left console pane of the Registry menu, select *Channels*, as shown in the following image.

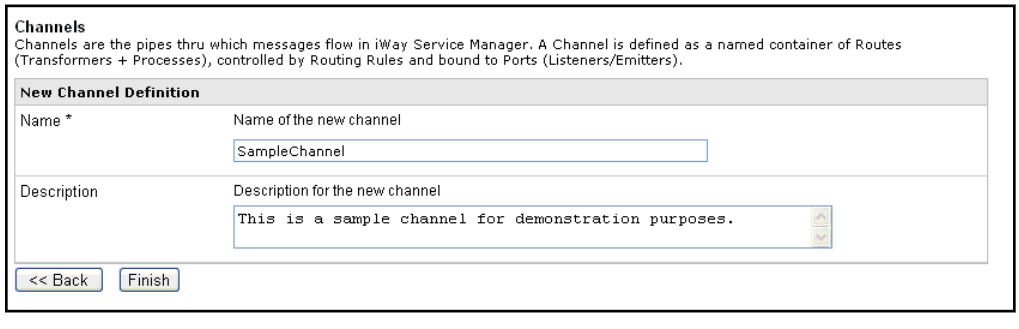
The Channels pane opens, as shown in the following image.



The table that is provided lists each channel that is defined with a brief description.

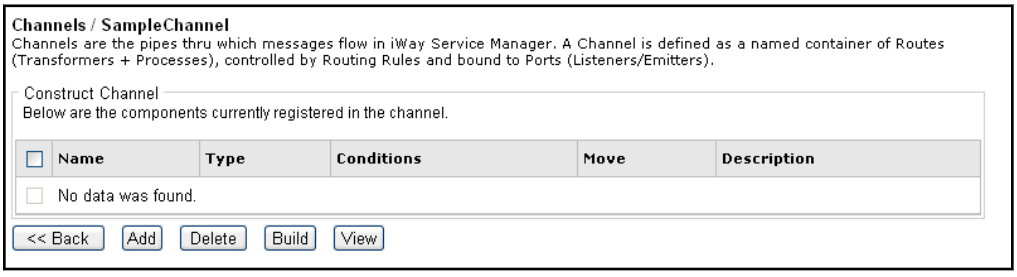
2. Click *Add*.

   The New Channel Definition pane opens, as shown in the following image.



3. Enter a name, for example, SampleChannel, and description for the channel.
4. Click *Finish*.

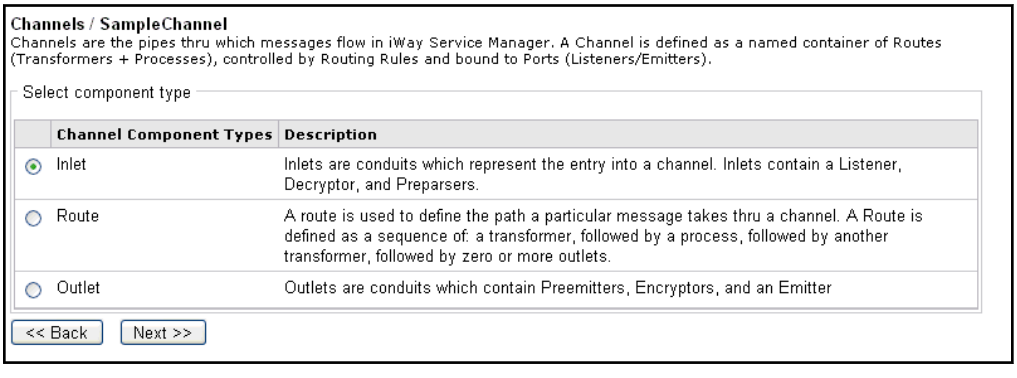
The Construct Channel pane opens, as shown in the following image.



The table that is provided is used to list the components that are currently registered with the channel.

5. Click *Add*.

The Select component type pane opens, as shown in the following image.

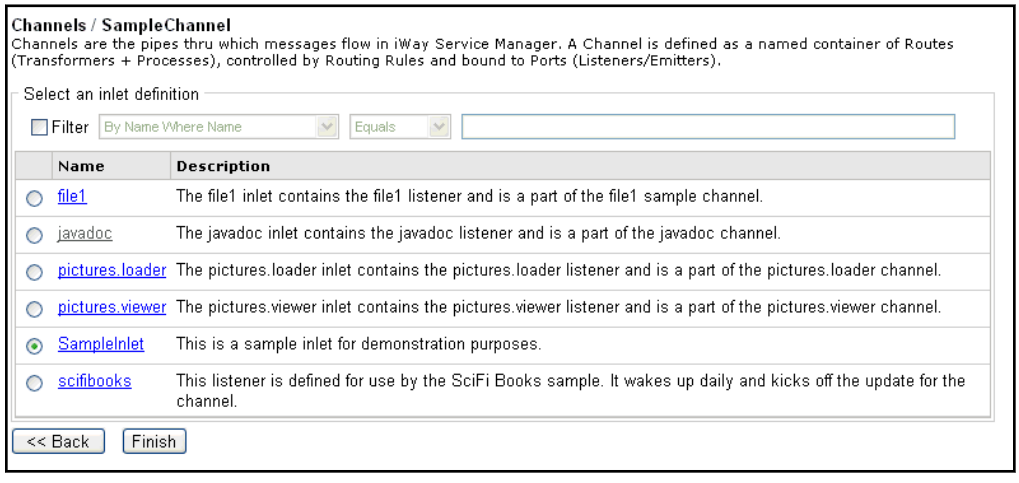


The table that is provided lists the component types you can select and register with the channel you are defining.

- **Inlet.** Conduits that represent the entry into a channel.
- **Route.** Used to define the path a particular message takes through a channel.
- **Outlet.** Conduits that consist of Preemitters, Encryptors, and Emitters.

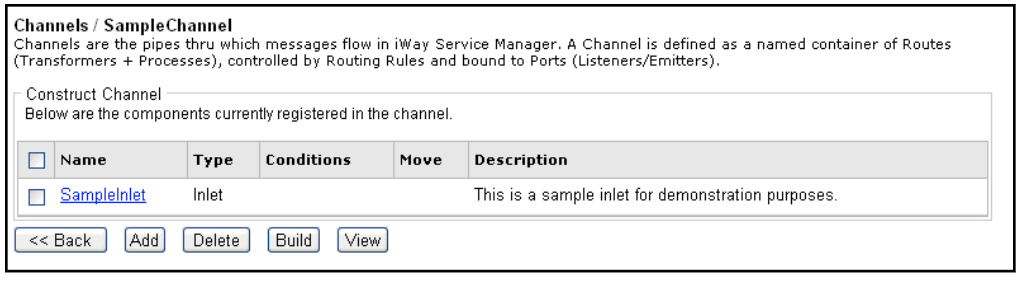
6. Select *Inlet* from the list of component types and click *Next*.

The select an inlet definition pane opens, as shown in the following image.



7. Select an available inlet, for example, SampleInlet, from the list and click *Finish*.

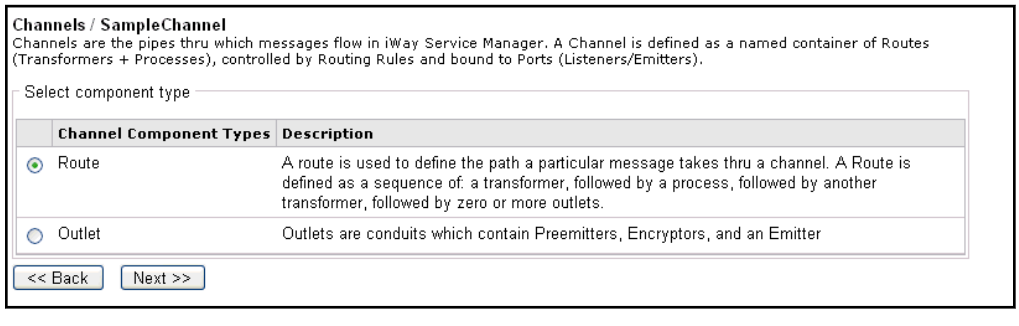
   As shown in the following image, you are returned to the Construct Channel pane, which now includes the inlet (SampleInlet) you defined earlier.



   You are now ready to add a route to the channel.

8. Click *Add*.

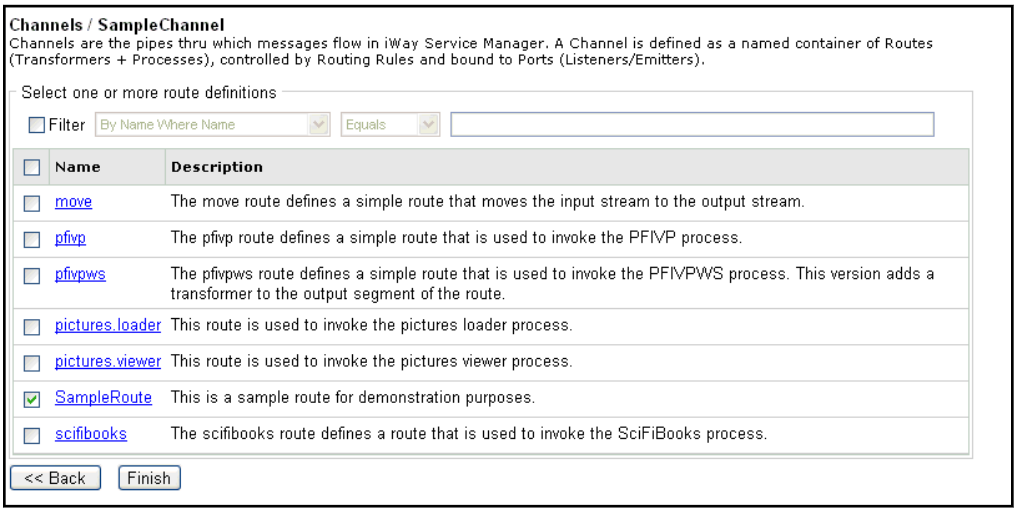
   The Select component type pane opens, as shown in the following image.



Notice that only the Route and Outlet component types are listed, since you have already added an inlet to the channel.

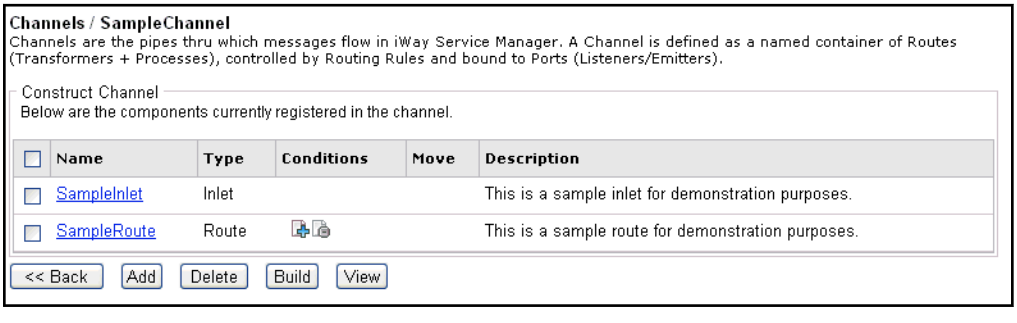
9. Select *Route* from the list of component types and click *Next*.

   The select one or more route definitions pane opens, as shown in the following image.



10. Select an available route, for example, SampleRoute, from the list and click *Finish*.

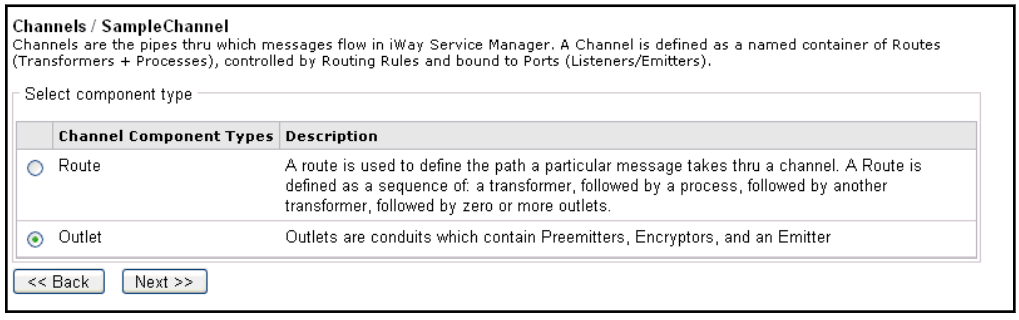
    As shown in the following image, you are returned to the Construct Channel pane, which now includes the inlet (SampleInlet) and route (SampleRoute) you defined earlier.



    You are now ready to add an outlet to the channel.

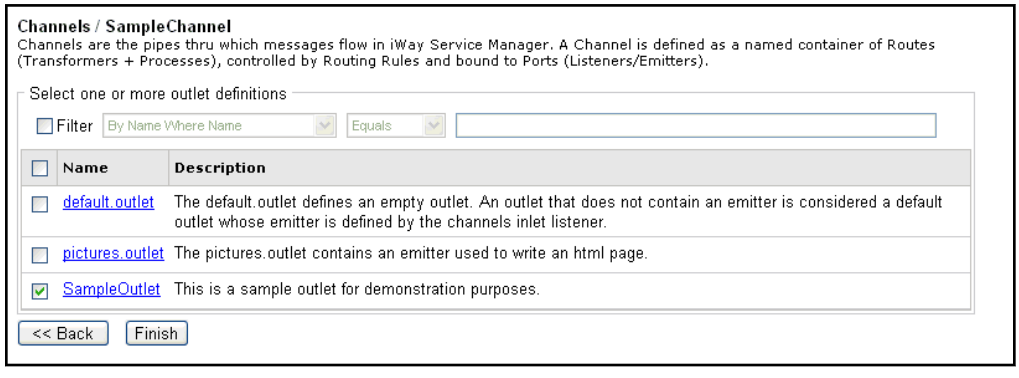
11. Click *Add*.

The Select component type pane opens, as shown in the following image.



12. Select *Outlet* from the list of component types and click *Next*.

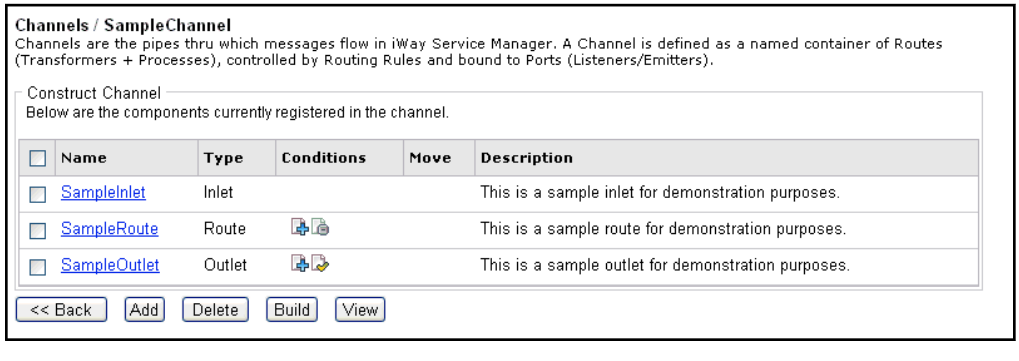
    The select one or more outlet definitions pane opens, as shown in the following image.



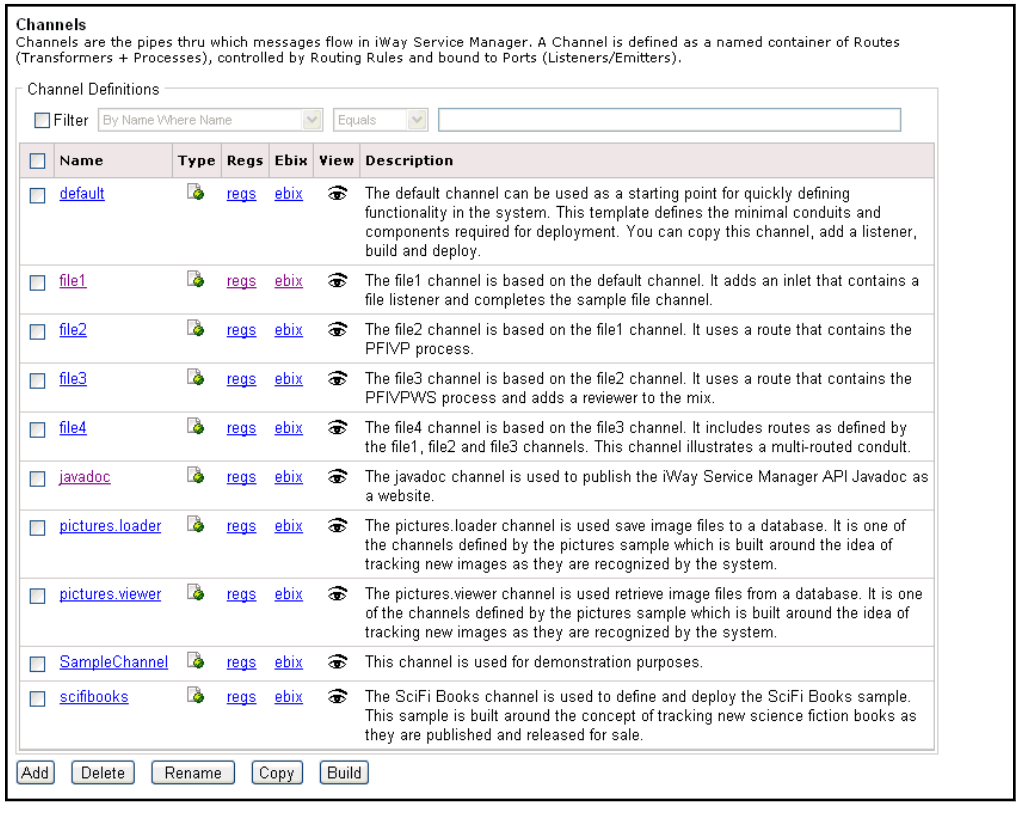
13. Select an available outlet, for example, SampleOutlet, from the list and click *Finish*.

    You can assign multiple routes and multiple outlets to the channel.

    As shown in the following image, you are returned to the Construct Channel pane, which now includes the inlet (SampleInlet), route (SampleRoute), and outlet (SampleOutlet) you defined earlier.



If you return to the main channels pane, you will notice that the channel you just constructed (SampleChannel) has been added to the list, as shown in the following image.



After you have designed your channel, you are ready to build and deploy it into a run-time environment.

Building a Channel

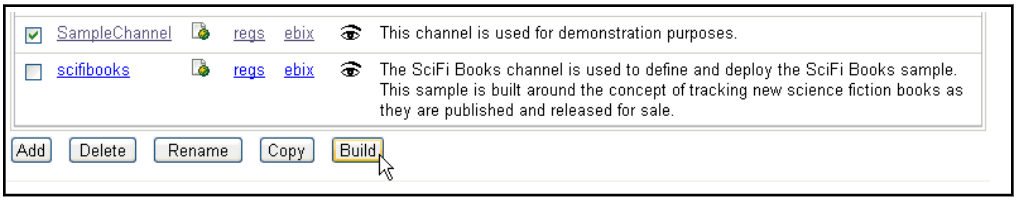
After constructing a channel, building a channel is the first stage in channel management. This process compiles all the registered channel components (inlet, route, and outlet) and validates the combination of components you have selected.

Procedure: How to Build a Channel

To build a channel:

1. In the left console pane of the Registry menu, select *Channels*.

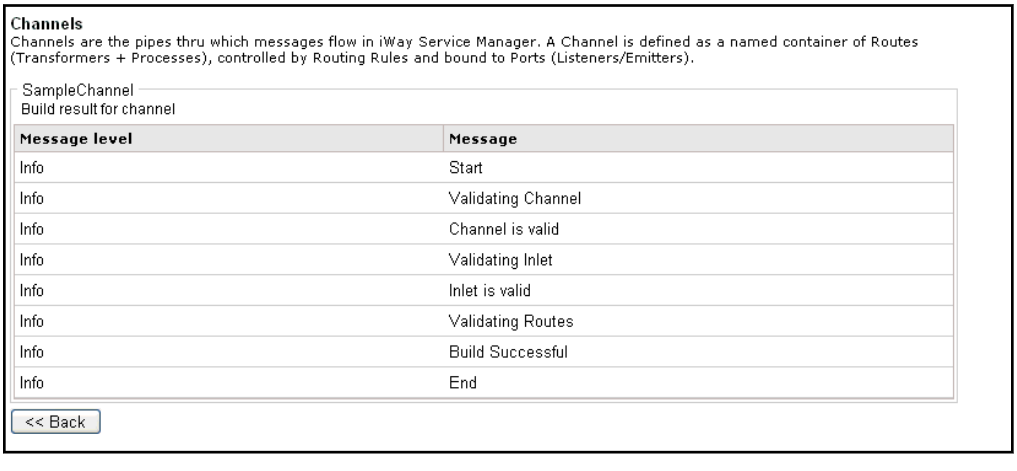
The Channels pane opens, as shown in the following image.



The table that is provided lists each channel that is defined with a brief description.

2. Select the check box next to the channel you want to build, for example, SampleChannel, and click *Build*.

The build result pane for the channel opens, as shown in the following image.



Each validation step is listed in the table and includes the final build result. If no errors are listed, you have successfully built a channel, which is now ready to be deployed.

Tip: To build more than one channel at once, you can select multiple check boxes in the Channels pane and click *Build*.

3. Click *Back* to return to the Channels pane.

For information on testing a channel, see the *iWay Service Manager User's Guide*.

Registering Library Files and Setting JVM Options

Many listeners and emitters require that you register library files with the system you are integrating or specifying JVM options. On Windows, the Path Settings function enables you to include JAVA class and JAR files in the class path. Use this function to add third-party drivers, such as those for IBM WebSphere MQ, Oracle AQ, and JDBC. You also can specify additional library directories that may be required when the third-party Java classes require dynamic link libraries or shared objects (depending on the platform in use). WebSphere MQ, for example, requires this type of setting.

See the section for the listener or emitter you are configuring for any library file requirements.

Note: For the change in these values to take effect, you must completely stop Service Manager and then start it again.

The location for registering libraries depends on the operating system. You must edit the UNIX startservice.sh or iway70.sh script to include additional Java libraries on that platform. For information on registering libraries, see the *iWay Service Manager User's Guide*.

Configuring Local Transaction Management

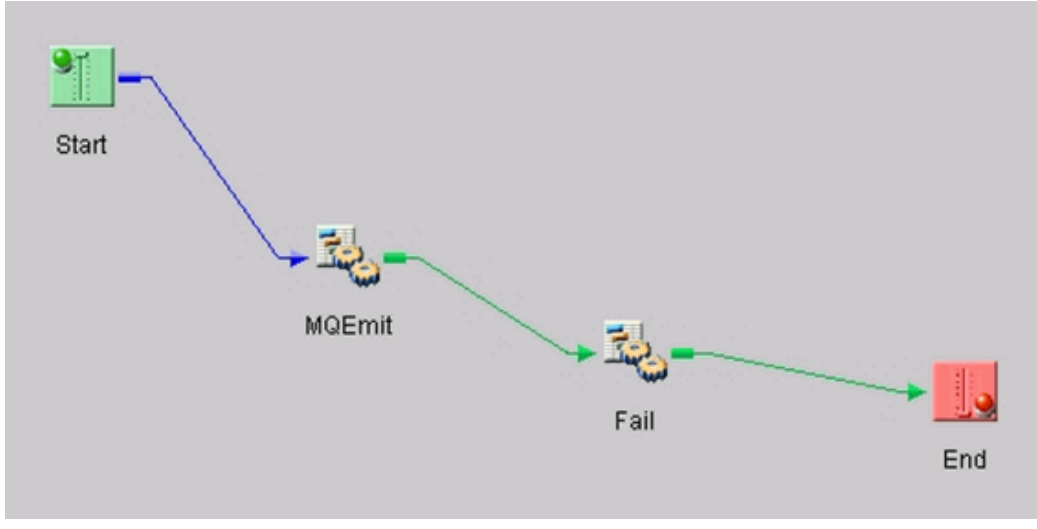
Many listeners in iWay Service Manager can be configured to function as transaction managers. To enable this functionality, ensure that the Listener is Transaction Manager parameter is set to *true* when you configure the listener parameters. When transaction management is enabled, any services are run within a local transaction that is managed by the listener.

You can configure a process flow to serve as the transaction boundary. In this case, a rollback can be triggered by adding a Fail service object to the process flow. A message on the queue is considered to be acquired and does not roll back to the original source. An error message is then sent to the configured channel outlet.

To demonstrate local transaction management, perform the following steps:

1. Create a channel using iWay Service Manager that contains a File listener as an inlet.
2. Set the Listener is Transaction Manager parameter to *true* for the File listener.
3. Create a process flow using iWay Designer that contains an object that will fail.

Consider the following process flow as an example:



In this process flow, the MQEmit service object is configured to emit a message to a valid MQ queue. The Join Local Transaction? parameter is set to *true* for the MQEmit service object. The Fail service object is set to *fail* and starts the rollback.

As a result, the message that is placed on the MQ queue by the MQEmit service object is removed and the following error message is sent to the configured channel outlet:

```
<eda>
  <error timestamp="2010-12-06T13:58:14Z" code="6" stage="AGENT"
source="Fail">
       XD[FAIL] cause: 0 subcause: 0 message: failure!
          <data type="xml">&lt;test/&gt;</data>
  </error>
</eda>
```

Chapter 3

e-Business Protocol Adapters

This section describes how to configure and use the iWay e-Business protocol adapters. It is intended for those developing e-commerce projects that require secure message and transaction processing over the Internet.

In this chapter:

AS1 and AS2 Standards

To offer an alternative to the proprietary Value Added Networks (VAN) protocol, the Internet Engineering Task Force (IETF) developed and approved the Electronic Data Interchange-Internet Integration (EDIINT) specification for conducting secure business transactions over the Internet. EDIINT uses the open Internet as the backbone and provides the following message-handling elements under an agreed upon standard named Applicability Statement 1 and 2 (AS1 and AS2):

- Guaranteed delivery
- On-time delivery
- Non-repudiation
- Authentication

Through implementation of these standard protocols, cooperating business enterprises can eliminate the requirement for the expensive VAN connectivity.

AS1 uses Simple Mail Transfer Protocol (SMTP) with S/MIME providing encryption and security by requiring authentication, message integrity, and originating non-repudiation. AS2 is a modification of AS1 that provides S/MIME support and uses direct HTTP or HTTP/S as its transport protocol.

The AS2 specification supports any data type transmission using the Internet over HTTP. AS2 governs data transport, not specific data-type validation or document processing. It designates the means by which to connect, deliver, validate, and acknowledge transport in a secure and reliable manner.

For information on configuring and using listeners for AS1 and AS2, see the *iWay Adapter for EDIINT User's Guide*.

Non-Blocking AS2 Adapter

The Non-Block AS2 (NAS2) adapter is a new nonblocking AS2 with improved performance, connection management, and various other security features.

The NAS2 adapter provides extensive flexibility by exposing an array of configurable parameters for the security providers, Message Disposition Notification (MDN) handling, CRL checking, and so on. The following sections describe some of the features that have been added as part of the improvement to the NAS2 adapter.

iWay Providers

You can configure multiple security providers and use them as named providers as part of the NAS2 configuration. For more information on configuring security providers, see the *iWay Service Manager User's Guide*.

- **KeyStore Provider**

  You can configure multiple KeyStore providers which can be used as keystores or truststores by the NAS2 adapter.

- **SSL Context Provider**

  You can configure multiple SSL Context providers and refer to one of them by name in the NAS2 adapter. This simplifies the SSL configuration by grouping all SSL parameters in one place. The SSL Context Provider simplifies the SSL configuration further by referring, by name, to previously configured keystore and certstore providers.

- **Directory CertStore Provider**

  This provider can be configured to point to a file system directory where peer certificates and CRLs are stored in files. You can configure multiple Directory Certstore providers and refer to them in the NAS2 adapter. CertStores are used to complete certificate chains and to retrieve CRLs during certificate verification.

- **Directory Provider**

  Can be configured to point to an LDAP system which can be used as a named certstore provider in the NAS2 configuration.

Features

- **LDAP Certificate Support.** Retrieval of partner certificates from the LDAP system as part of the certificate store configuration to complete the signature chain validation.
- **Signer Certificate Chain.** Option to not include the signer certificate when sending an AS2 message or replying with an MDN. This allows the user to minimize the message size for enhanced performance.
- **Certificate Revocation List Checking Option.** Allows the configuration of NAS2 to validate if the message being processed is signed using a revoked certificate. If the option for CRL checking is selected, it will require a configured certificate store on the NAS2 component which can point either to a list of named keystore providers, directory CertStore providers, and directory Providers (LDAP) where the revoked certificates are located.
- **Key Alias Selection.** On the S/MIME and SSL components, new parameters are exposed, which allow the user to specify the key alias with the keystore/truststore. This allows the user to pick which key to use for various security operations, such as signatures, decryptions, and so on.
- **Persistent Connection Support.** The NAS2 adapter supports persistent connections, which allows improved connection handling and management.
- **Ordering of Signature and Compression.** A new feature to allow the selection of compression and signature ordering is available. Now you can configure if the message should be signed and then compressed or compressed then signed.
- **Delayed MDN.** The NAS2 adapter also supports the new feature which is not typical to the standard AS2 processing, but allows a great degree of flexibility when it comes to MDN processing. When a message is received on the NAS2 listener, a user may configure the MDN to be delayed until the business processing of the message is completed. If the Delayed MDN option is selected, it is the responsibility of the user to invoke the corresponding MDN send service as part of the business processing that will send the MDN as requested by the originator of the message.
- **Safe Store for Messages.** The Safe Store option on the NAS2 component will safe store the message before performing any further processing to the message. This will prevent any message loss. After the message has been processed, it will be removed from the safe store. In the event that the system goes down, all the messages in the safe store are processed after the system is back on line.

- **Large File Limit.** The NAS2 adapter contains various internal improvements to handle large file sizes. A new option has been exposed on the NAS2 inbound processing that allows the user to limit the message size accepted by the NAS2 adapter.

Configuring NAS2 Listeners

A listener is a component that is responsible for receiving inbound messages through an assigned listener protocol. After a listener is created, it must be added to an inlet configuration. An inlet will become part of the final channel configuration that will consist of an inlet, route, and an outlet. For more information on configuring channels, see the *iWay Service Manager User's Guide*.

Prerequisites

Before using NAS2 you must first download and extract the jar files to the proper directory.

You can download the files at: *http://java.sun.com/javase/downloads/index.jsp*

Once the files are downloaded, extract them to \jre\lib\security. You will have to override the jars that are currently there.

Once this is done, you are able to use NAS2.

Procedure: How to Configure a NAS2 Listener

To configure a NAS2 listener:

1. Ensure that iWay Service Manager is running.

   On Windows, you can start iWay Service Manager by clicking *Start*, selecting *Programs*, *iWay 7.0 Service Manager*, and then *Start Service Manager* for the configuration you are currently using.

   For more information on starting and stopping iWay Service Manager, see the *iWay Service Manager User's Guide*.

2. Open a browser window and point to the following URL:

   ```
   http://host:port/ism
   ```

   where:

   host

   Is the host machine on which iWay Service Manager is installed.

   port

   Is the port on which iWay Service Manager is listening. The default port is 9999.

On Windows, alternatively, you can click *Start*, select *Programs*, *iWay 7.0 Service Manager*, and then click *Console*.

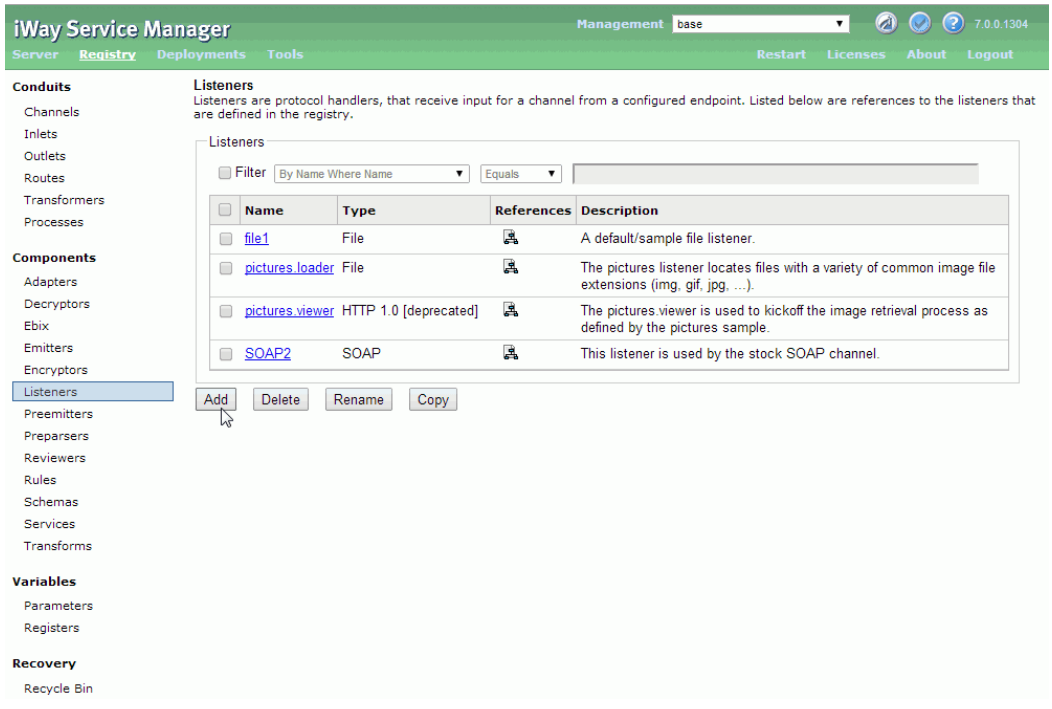
A login dialog box opens.

3. Type a user name and password for the configuration you are using, and click *OK*.

   The iWay Service Manager Administration Console opens.

4. Click *Registry* in the top pane, and then click *Listeners* in the left pane.

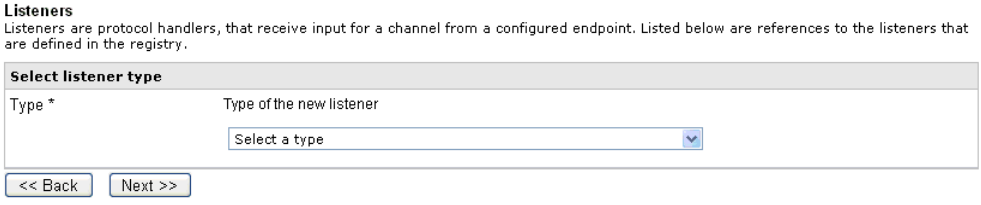
   The Listeners pane opens.



The table that is provided lists all the previously configured listeners and a brief description for each.

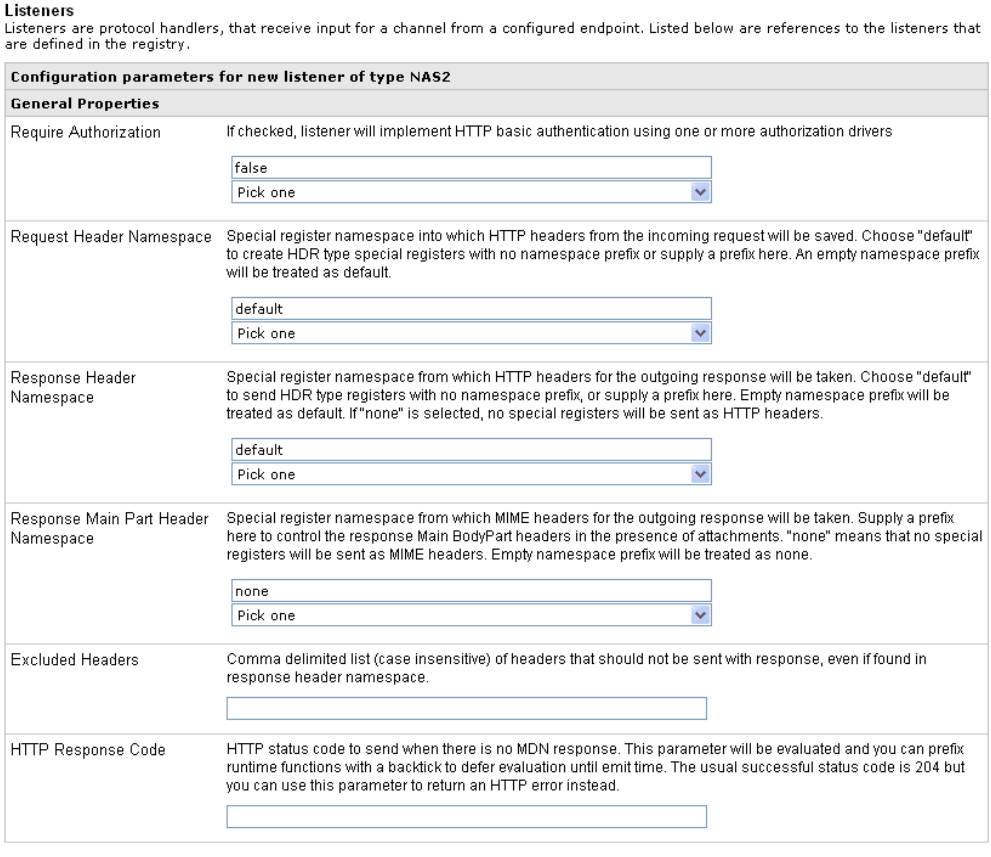
5. Click *Add*.

The Select listener type pane opens.



6. Select *NAS2* from the Type drop-down list and click *Next*.

The configuration parameters for the NAS2 listener opens.

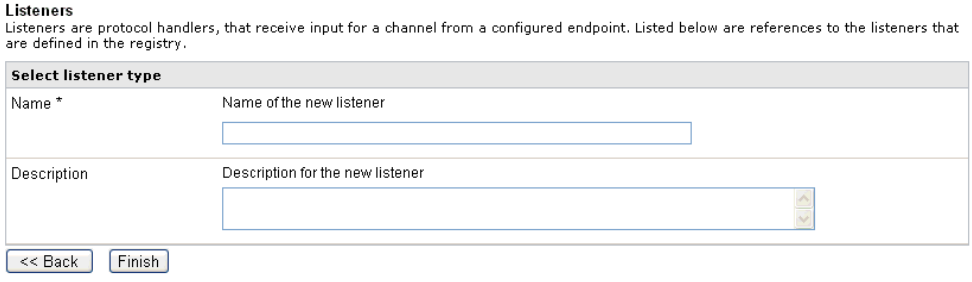


Note: The parameters prefixed with an asterisk (*) in the listener configuration pane are required.

7. Provide the appropriate values for the NAS2 listener parameters.

For more information, see *NAS2 Listener Configuration Parameters* on page 49.

8. Click *Next*.
   You are returned to the Select listener type pane.



9. Enter a name for the NAS2 listener and description (optional).
10. Click *Finish*.
    You can now use this listener as part of your channel configuration where the business logic is applied to the received messages.

Reference: NAS2 Listener Configuration Parameters

The following table lists and describes parameters for the NAS2 listener.

Parameter	Description
General Properties	
Authentication Scheme	The scheme to apply when authenticating HTTP requests.
Authentication Realm	If authentication is required, then the name of the configured Realm provider is used.
Request Header Namespace	The special register namespace to which HTTP headers from the incoming requests are saved. The Default Namespace option creates HDR type special registers without a namespace prefix.
Response Header Namespace	The special register namespace from which HTTP headers for the outgoing response are taken. The Default Namespace option sends HDR type registers with no namespace prefix. If None is selected, then no special registers are sent as HTTP headers.

Parameter	Description
Response Main Part Header Namespace	The special register namespace from which MIME headers for the outgoing response are taken. Provide a prefix to control the response Main BodyPart headers in the presence of attachments. Selecting none means that no special registers are sent as MIME headers.
Excluded Headers	A comma delimited list (case insensitive) of headers that should are not sent with the response, even if they are found in the response header namespace.
HTTP Response Code	An HTTP status code to send when there is no MDN response. This parameter will be evaluated and you can prefix run time functions with a backtick to defer its evaluation until emit time. The usual successful status code is 204, but you can use this parameter to return an HTTP error instead.
Use Safestore?	If set to *true*, the listener persists incoming messages after handling any protocol-related packaging. Messages are removed from the safestore upon completion of processing or on error if the HTTP response has not yet been returned to the client. Messages remaining in the safestore are processed at listener startup.
Maximum Request Entity Size	When a request is received that is larger than the maximum, the listener will return a 413 HTTP status code and close the connection. Leave this field blank or set a value of zero to have no maximum size limit. The default value is 256KB.
Compress Response	If set to *true*, the response is compressed with gzip or deflate compression when the client indicates that it can accept compressed transfer encoding.
IP Properties	
Port	The TCP port for receipt of HTTP requests.
Local bind address	The local bind address for multi-homed hosts. This parameter value is usually not specified.

Parameter	Description
Persistence	If set to *true*, the connection is maintained when the client requests to do so. Otherwise, the connection is closed.
Maximum Connections	This parameter defines the maximum number of simultaneous connections that are allowed. When this threshold is reached, new connections are not accepted until the current connections are closed and the total number of connections is below the limit. Leave this field blank (default) or set a value of zero to have no maximum limit of connections.
Persistence Timeout value in Minutes	The maximum length of time that a connection can persist with no activity.
Set Response NoDelay	If set to *true*, it disables the Nagle Algorithm on the response. This will result in a faster line turnaround at the expense of an increased number of packets.
Reuse Address	If set to *true*, when a connection is closed, it immediately makes the address available, bypassing TCP defaults.
Allowable Clients	If supplied, then only messages from this list of fully qualified host names and/or IP addresses are accepted. Accepts comma-separated list or use the FILE() function.
Secure Connection (SSL)	
Secure Connection	If set to *true*, a connection over HTTPS is made.
SSL Context Provider	The named iWay Security provider for SSL Context.
S/MIME	
S/MIME Keystore Provider	The name of an iWay KeyStore provider used to decrypt incoming messages and sign receipts.
S/MIME Truststore Provider	The name of an iWay KeyStore provider containing the S/MIME certificate authorities.

Parameter	Description
S/MIME Certificate Store Providers	The Comma-separated list of Keystore, Directory Certstore or LDAP providers for the certificate stores used to complete signer certificate chains when the signed message contains fewer certificates than needed.
S/MIME JCE Cryptography Provider	The JCE provider for S/MIME cryptography services.
S/MIME PKIX JCE Provider	The JCE provider for S/MIME PKIX services.
S/MIME Decryption Key Alias	The private key alias used to decrypt incoming messages.
S/MIME Decryption Key Password	The password for the Description Private key. If left blank, then the password for accessing the keystore is used.
Enforce KeyUsage Extension	If set to *true*, verify certificates used for signing allow the digital signature KeyUsage extension, and certificates used for encryption allow the keyEncipherment KeyUsage extension.
Enable Certificate Revocation	If set to *true*, use the CRLs from the CertStores to check whether the certificate signer has been revoked.
Unrecognized Certs Location	The directory to store unrecognized certificates found in S/MIME messages.
Payload Header Namespace	The special register namespace to which any headers on the extracted body part are stored as HDR registers. If no value is supplied, then the body part headers are saved in the default namespace.
Keep Message Flat	If set to *true*, the body of the message will be kept as an array of bytes.
MDN (Receipt)	
Delayed MDN	If set to *true*, MDN is delayed until after the request is processed. If specified as delayed, then the MDN must be sent from the process that handles this message. Failure to do so will result in an HTTP204.

Parameter	Description
MDN Header Namespace	A special register namespace from which MIME headers for the multipart/report entity is taken. This namespace is different than the Response Header Namespace when the MDN is wrapped by an external signature. The default value is mdnhdr.
MDN Field Namespace	A special register namespace where special registers are used to override or add MDN field values. The default value is mdn.
MDN Reporting User Agent	The value of the Reporting-UA field in the MDN. The default value is AS2 Server.
SMTP Host	The host name of the SMTP server. Used for asynchronous MDN through email.
SMTP User	The user name to access SMTP server.
SMTP Password	The password to access SMTP server.
From	The email address used in the From field of the receipt message.
HTTP Client Provider	The HTTP Client provider that manages outgoing connections for asynchronous MSNs.
HTTP Version for Asynchronous MDN	The HTTP Version used to send asynchronous MDNs over HTTP or HTTPS.
Compress Asynchronous MDN	If set to *true*, asynchronous MDNs over HTTP or HTTPS are compressed using one of the following encoding options: ❑ deflate {deflate} ❑ gzip {gzip} ❑ none {none} The content-encoding header is set accordingly.
MDN S/MIME Keystore Provider	The provider for the keystore used to sign receipts. Defaults to the value assigned to the S/MIME Keystore Provider.

Parameter	Description
MDN S/MIME JCE Cryptography provider	The JCE Provider for MDN S/MIME cryptography service. Defaults to the value assigned to the S/MIME JCE Provider.
MDN S/MIME Signature Key Alias	The private key alias used to sign receipts.
MDN S/MIME Signature Key password	The password for signature private key. Defaults to *password* for accessing the keystore.
Include Certificate Chain	Determines how much of the signer certificate chain is included in a signed receipt. Options include: ❑ *Complete Certificate Chain* ❑ *Signer Certificate only* ❑ *No Certificates*
Other	
Optimize Favoring	The selection of memory is useful for large input documents.
Multithreading	The number of documents that can be processed in parallel.
Execution Time Limit	The time limit for document execution (in seconds) before it is terminated.
Default Java File Encoding	The default encoding if incoming message is not self-declaring.
Agent Precedence	The changes in order by which iSM selects agents. This is normally set to Document overrides listener.
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined replies.
Error Documents treated normally	If set to *true*, error documents are processed by any configured pre-emitters.
Listener is Transaction Manager	If set to *true*, agents run within a local transaction managed by the listener.

Parameter	Description
Record in Activity Log(s)	If set to *true*, activity on this channel will be recorded in the activity logs. If set to *false*, the activity will not be recorded.

Reference:

Special Registers for the NAS2 Listener

The following table lists and describes the special registers for the NAS2 listener.

Special Register	Level	Description
...	Header	Each header value from the message.
action	Document	The action field of the post.
as2from	Document	The AS2-From header.
as2to	Document	The AS2-To header.
asxDocType	Document	The payload for an AS2 request or MDN for a receipt.
dispositionFileName	Header	The FileName parameter extracted from the Content-Disposition header.
ip	System	The IP of the sending system.
iwayconfig	System	The current active configuration name.
mdnDisposition	Document	The field extracted from the received MDN.
mdnFinalRecipient	Document	The field extracted from the received MDN.
mdnOriginalMessageId	Document	The field extracted from the received MDN.
mdnOriginalRecipient	Document	The field extracted from the received MDN.
mdnReceivedContentMIC	Document	The field extracted from the received MDN.
mdnReportingUA	Document	The field extracted from the received MDN.
mdnRequested	Document	If set to *true*, a receipt is requested. If set to *false*, no receipt is requested.

Special Register	Level	Description
mdnSent	Document	If set to *true*, the MDN was already sent. If it is set to *false*, the MDN was not sent.
msgsize	Document	The physical length of the message payload.
name	System	The assigned name of the master (listener).
protocol	System	The protocol on which the message was received.
requestType	Header	The type of HTTP request (GET, POST, or HEAD).
smime_compressed	Document	If set to *true* the S/MIME message is compressed. If set to *false*, the message is not compressed.
smime_encrypted	Document	If set to *true* the S/MIME message is encrypted. If set to *false*, the message is not encrypted.
smime_error	Document	The error message that can be used when sending an MDN.
smime_error_diag	Document	The error diagnostic message that can be used when sending an MDN.
smime_mic	Document	The message Identification Code extracted from the S/MIME message
smime_signed	Document	The unsigned, embedded or external depending on the S/MIME packaging that was used.
smime_signer	Document	The Distinguished Name from the Signer certificate.
smime_signer_cn	Document	Common Name (CN) extracted from the Signer certificate.

Special Register	Level	Description
smime_signing_time	Document	Signing time extracted from the S/MIME signed attributes.
source	Document	The host name of the sending system.
url	Header	The full URL of the HTTP request (GET, POST, or HEAD).
tid	Document	Unique transaction ID.

Configuring Emit Services

You can configure outbound processing of AS2 messages as a service that can be used within a process flow, which will become part of the route configuration or directly as a service assigned to a route. In this case, a business process can continue after an AS2 message has been sent out to the client. The following section describes how to configure an AS2 nonblocking emit service. For more information on configuring outlets and routes, see the *iWay Service Manager User's Guide*.

Procedure: How to Configure an AS2 Nonblocking Emit Service

To configure an AS2 nonblocking emit service:

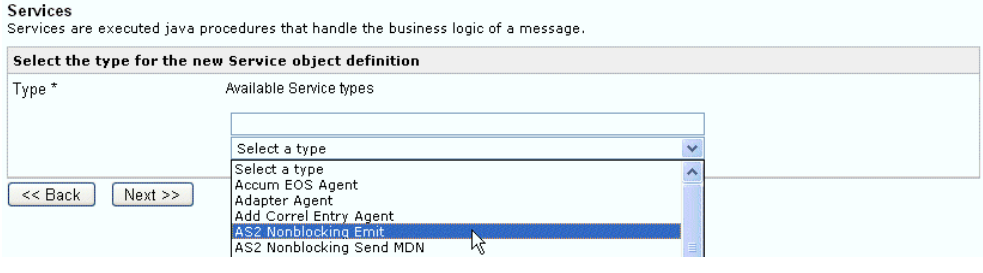
1. Click *Registry* in the top pane, and then click *Services* in the left pane.

   The Services pane opens.

   The table that is provided lists all the previously configured services and a brief description for each.

2. Click *Add*.

   The Select Service type pane opens.



3. Select *AS2 Nonblocking Emit* from the Type drop-down list.
4. Click *Next*.

   The configuration parameters pane for the AS2 nonblocking emit service opens.
5. Provide the appropriate values for the AS2 nonblocking emit service parameters.

   For more information, see *AS2 Nonblocking Emit Service Configuration Parameters* on page 58.
6. Click *Next*.

   The name and description pane opens.
7. Enter a name for the service and description (optional).
8. Click *Finish*.

Reference: AS2 Nonblocking Emit Service Configuration Parameters

The following table lists and describes parameters for the AS2 nonblocking emit service.

Parameter	Description
Configuration Parameters	
Destination	The URL that is used to post this information.
HTTP Client Provider	The HTTP client Provider that is used to manage connections for this emitter.
AS2-From	A textual value identifying the sender of data exchange.
AS2-To	A textual value identifying the receiver of data exchange.
Subject	Sets the Subject header.
Request Receipt	Tells the emitter to send a request for receipt in the form of a Message Disposition Notification (MDN).

Parameter	Description
Asynchronous Receipt URL	If an asynchronous receipt is requested, you must specify the URL to which the receipt should be sent. Supported values are in the form: ❏ mailto:user@host- for asynchronous receipt by email ❏ http://host[:port]/- for asynchronous receipt by HTTP ❏ https://host[:port]/- for asynchronous receipt by HTTPS
Receipt Destination	The directory to which synchronous MDNs are stored. Specific file names are optional. Use an asterisk (*) in file name to be replaced by timestamp, # by sequential counter.
Content-Type	Specifies the content-type of data that is sent. Select from the drop-down list or provide your own.
Message ID	Set this to control the emitted message ID. Usually this is left blank to let the system generate a unique ID meeting the requirements of RFC 822. Use this only to override the default. This is not recommended.
Content Disposition	The file name to put in the Content-Disposition header value.
User ID	The user ID for Basic Authentication challenges.
Password	The password for Basic Authentication challenges.
Domain	The domain for NTLM authentication challenges. Note that to use NTLM, you must enable connection persistence.
Request Header Namespace	The special register namespace from which HTTP headers for the outgoing request will be taken. ❏ Default Namespace to send HDR type registers ❏ Supply a namespace prefix here to indicate which headers to send ❏ None means that no special registers will be sent as HTTP headers.

Parameter	Description
Request Main Part Header Namespace	The special register namespace from which MIME headers for the outgoing request are taken. Provide a prefix to control the request Main BodyPart headers in the presence of attachments. Selecting *none* means that no special registers will be sent as MIME headers.
Response Header Namespace	The special register namespace to which HTTP headers for the incoming response are saved. ❑ Default Namespace to create special registers with no namespace ❑ Supply a namespace prefix here to indicate header namespace
MDN Header Namespace	The special register namespace into which MIME headers of the multipart/report entity will be saved. This namespace is ignored if the MDN is unsigned since all headers will be in the Response Header Namespace.
MDN Field Namespace	The special register namespace into which MDN fields are saved.
Excluded Headers	A comma delimited list (case-insensitive) of headers that should not be sent with the request, even if they are found in the request header namespace.
Ask for Compressed Response	If set to *true*, the requests will set the Accept-Encoding header to indicate that the client can accept a compressed response, as described in RFC-2616. If the response has a compressed content encoding, the client will automatically inflate.
Compress Request	If set to *true*, the request entities will be compressed using the selected encoding and the content-encoding header are set accordingly.
Replace Connection?	If set to *false*, the connection will not be returned to the connection pool immediately. The identifier connection will be stored in the httpclient-key special register and the connection can be handled by the HTTP Client Manager agent.

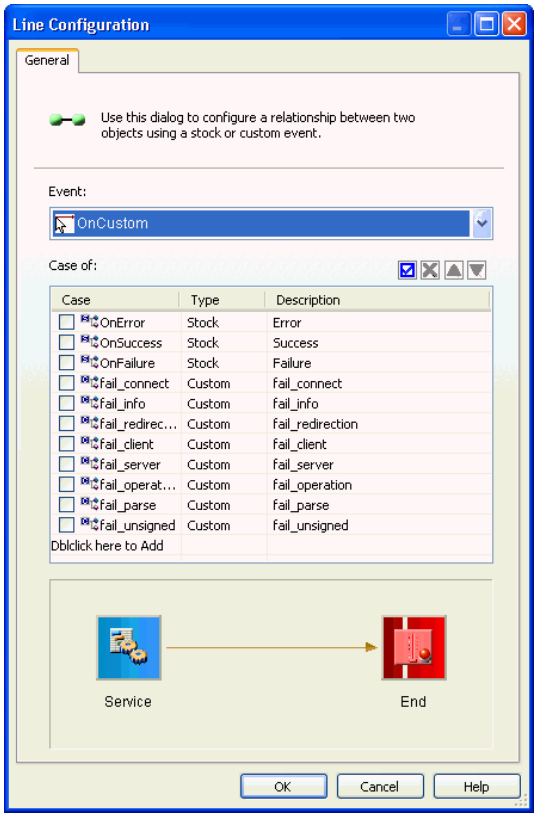
Parameter	Description
Maximum HTTP Client Manager Delay	The maximum time the HTTP Client Manager can take to deal with a particular connection before it is automatically aborted. The format is [xxh][xxm]xx[s]. The default is 60 seconds.
Maximum Request Size	The maximum size, after compression, of a request entity that is sent with this emitter. 0 means no maximum and blank will default to 256KB.
Maximum Response Size	The maximum size of a response entity that is received by this emitter. 0 means no maximum and blank will default to 256KB.
Try Expect/Continue Handshake?	If checked, the client will send the HTTP Expect: 100-continue header and await HTTP 100 response before sending request body.
S/MIME	
Packaging	Tells the emitter how the document should be packaged for transmission. Select from the drop-down list: ❑ Encrypted ❑ Signed ❑ Signed and Encrypted ❑ Un-encrypted
Compression	Determines when message compression should be applied. Select from the drop-down list: ❑ Compress After Signature ❑ Compress Before Signature ❑ No Compression
S/MIME Keystore Provider	The provider for the Keystore used to sign and encrypt messages.

Parameter	Description
S/MIME Truststore Provider	The provider for the Keystore containing the S/MIME Certificate Authorities.
S/MIME Certificate Store Providers	A Comma-separated list of Keystore, Directory CertStore, or LDAP providers for the certificate stores, used to complete signer certificate chains when the signed message contains fewer certificates than needed.
S/MIME JCE Cryptography Provider	The JCE Provider for S/MIME Cryptography services.
S/MIME Verification JCE Crypto Provider	The JCE Provider for S/MIME verification cryptography services. Normally left blank. Defaults to S/MIME JCE Provider.
S/MIME PKIX JCE Provider	The JCE Provider for S/MIME PKIX services. If left blank, the default JCE provider for PKIX will be used.
Recipient Public Key Alias	The alias for the recipient public key entry used for encryption.
Signature Key Alias	The alias for the private key entry used for signing.
Signature Key password	The password to access the signature private key. If left blank, the password used to access the Keystore will be used.
Digest Algorithm	The algorithm used for signing.
Encryption Algorithm	The algorithm used for encrypting.
Include Certificate Chain	Determines how much of the signer certificate chain is included in the message. Select from the drop down: ❑ Complete Certificate Chain ❑ No Certificate ❑ Signer Certificate only

Parameter	Description
Enforce KeyUsage Extension	If on, verify certificates used for signing allow the digital Signature KeyUsage extension, and certificates used for encryption allow the key Encipherment KeyUsage extension.
Enable Certificate Revocation	If set to *true*, this uses the CRLs from the CertStores to check whether the certificate signer has been revoked.
Unrecognized Certs Location	The directory to store unrecognized certificates found in S/MIME messages.
TCP	
Persistence	If checked, ask the server to maintain the connection.
Response Timeout value in seconds	The seconds to wait for response before signaling error.

Available Response Edges for NAS2Emit Agent

When you connect the NAS2EmitAgent object to an End object using the *OnCustom* build relation in a process flow, the available line edges are provided in the Line Configuration dialog box.



The following table lists and describes the available line edges for the NAS2EmitAgent object.

Line Edge	Description
OnError	Error
OnSuccess	Success
OnFailure	Failure
fail_connect	fail_connect

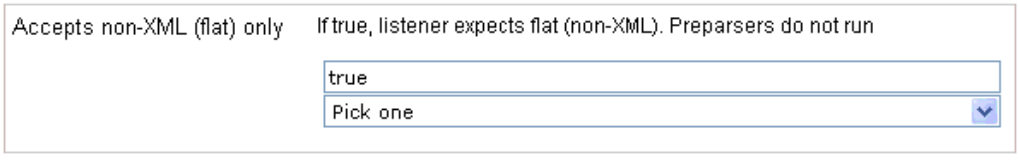
Line Edge	Description
fail_info	fail_info
fail_redirection	fail_redirection
fail_client	fail_client
fail_server	fail_server
fail_operation	fail_operation
fail_parse	fail_parse
fail_unsigned	fail_unsigned

Configuring S/MIME Packer and Un-Packer Services

The S/MIME packer service and corresponding S/MIME un-packer service are two new services that are available in the NAS2 adapter configuration. These services allow you to securely exchange information using the S/MIME format through any protocol. Using these services enable you to receive a payload from any source and package it into an S/MIME message that can be sent through any supported protocol. On the receiving side, you can use the S/MIME un-packer service to validate and verify the received message and unpack it.

Note: Since an unencrypted S/MIME packaged message is the same as a MIME message, the S/MIME packer and un-packer services can process MIME and S/MIME message formats. As a result, when you use the S/MIME packer service and select *Un-encrypted* from the Packaging drop-down list during configuration, a MIME message is generated. Similarly, the S/MIME un-packer service can process a MIME message since it is identical to the un-encrypted S/MIME message. In addition, the un-encrypted packaging for the message indicates that the message will not be signed and will always produce a document using MIME format as a result.

The S/MIME packer service allows you to send a packaged S/MIME message to any type of listener. The listener that receives the packaged S/MIME message must be able to unpack and process this package. Since an S/MIME packaged message can not be parsed by the listener and also represents a flat document, you must disable parsing for the listener. Using the iWay Service Manager Administration Console, the Accepts non-XML (flat) only parameter for the listener receiving the message must be set to *true* to disable parsing, as shown in the following image.



These services can be used as a stand alone service within a route configuration or as part of a process flow for more complex configuration. This process simulates the message exchange via NAS2 HTTP based adapter, but allows you to separate the protocol part of the adapter from the message processing part. As a result, you can exchange messages via any supporting protocol, such as file, email, and so on.

S/MIME Packer Service

The S/MIME packer service is configured with a special register message namespace that saves the message headers generated by the packaging process. This namespace will contain the headers for the message that will be required by the un-packer service on the receiving side to correctly unpack the S/MIME package. The S/MIME packer outputs a bytes-type XDDocument with any message headers stored in the specified message namespace. When a signed packaging is requested, for example, the output will consist of a bytes-type document that starts with the first message boundary. As with the AS2 emit service, another namespace can also be specified for payload headers.

Procedure: How to Configure a S/MIME Packer Service

To configure a S/MIME packer service:

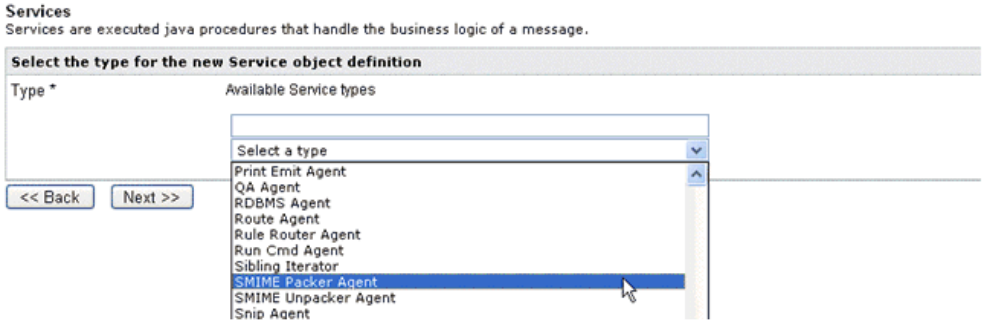
1. Click *Registry* in the top pane, and then click *Services* in the left pane.

   The Services pane opens.

   The table that is provided lists all the previously configured services and a brief description for each.

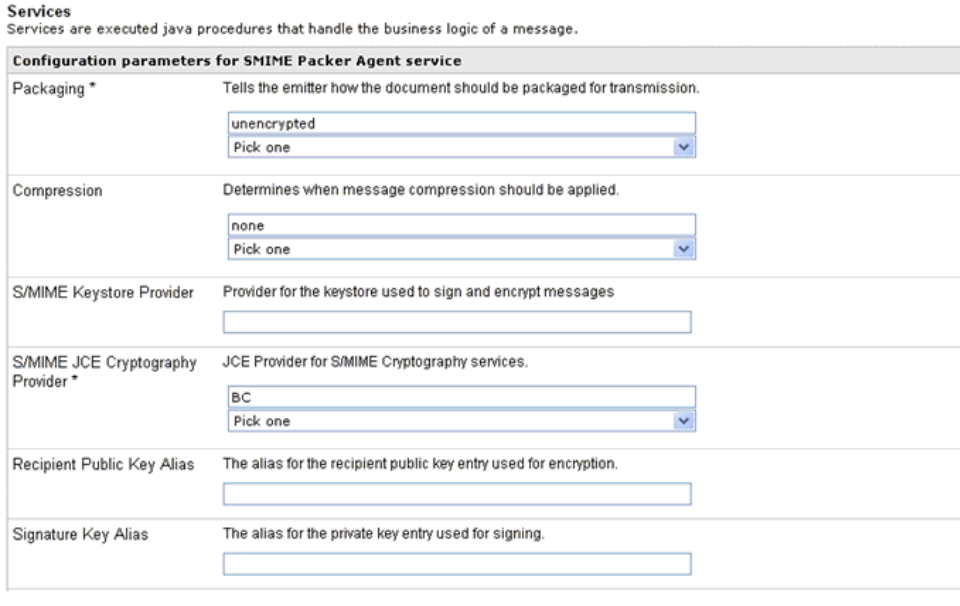
2. Click *Add*.

The Select Service type pane opens.



3. Select *SMIME Packer Agent* from the Type drop-down list.
4. Click *Next*.

   The configuration parameters pane for the S/MIME packer service opens.



5. Provide the appropriate values for the S/MIME packer service parameters.

   For more information, see *S/MIME Packer Service Configuration Parameters* on page 68.
6. Click *Next*.

   You are returned to the Select Service type pane.

7. Enter a name for the service and description (optional).
8. Click *Finish*.

Reference: S/MIME Packer Service Configuration Parameters

The following table lists and describes parameters for the S/MIME packer service.

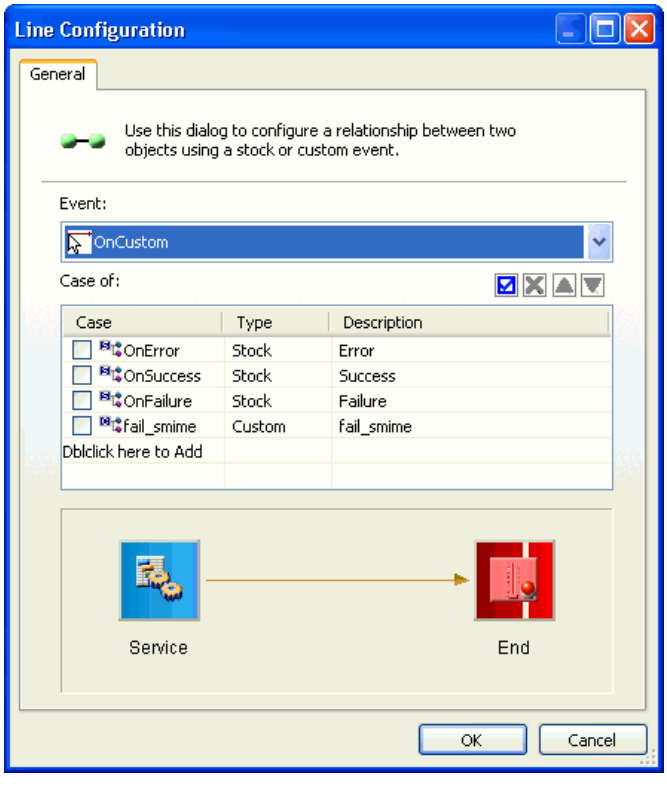
Parameter	Description
Configuration Parameters	
Packaging	Tells the emitter how the document should be packaged for transmission. Available options include: ❑ Encrypted ❑ Signed ❑ Signed and Encrypted ❑ Un-encrypted
Compression	Determines when message compression should be applied. Available options include: ❑ Compress After Signature ❑ Compress Before Signature ❑ No Compression
S/MIME Keystore Provider	The provider for the Keystore used to sign and encrypt messages.
S/MIME JCE Cryptography Provider	The JCE Provider for S/MIME Cryptography services.
Recipient Public Key Alias	The alias for the recipient public key entry used for encryption.
Signature Key Alias	The alias for the private key entry used for signing.
Signature Key password	The password to access the signature private key. If left blank, the password used to access the Keystore are used.

Parameter	Description
Digest Algorithm	The algorithm used for signing.
Encryption Algorithm	The algorithm used for encrypting.
Include Certificate Chain	Determines how much of the signer certificate chain is included in the message. Select from the drop-down list: ❏ Complete Certificate Chain ❏ No Certificate ❏ Signer Certificate only
Enforce KeyUsage Extension	If on, verify certificates used for signing allow the digital Signature KeyUsage extension, and certificates used for encryption allow the key Encipherment KeyUsage extension.
Main	
Message ID	Set this to control the emitted message ID. Usually this is left blank to let the system generate a unique ID meeting the requirements of RFC 822. Use this only to override the default. This is not recommended.
Content-Type	Specifies the content-type of data to be send. Select from drop down or provider your own.
Content Disposition	The file name to put in the Content-Disposition header value.
Header Management	
Payload Header Namespace	The special register namespace from which additional MIME headers for the payload are taken. If not supplied, no MIME headers are added beyond the content headers generated by the packaging process.

Parameter	Description
Message Header Namespace	The special register namespace to which message headers generated by the S/MIME packaging process are stored. If not supplied, message headers are saved in the default namespace.

Available Response Edges for SMIMEPackerAgent

When you connect the SMIMEPackerAgent object to an End object using the *OnCustom* build relation in a process flow, the available line edges are provided in the Line Configuration dialog box.



The following table lists and describes the available line edges for the SMIMEPackerAgent object.

Line Edge	Description
OnError	Error
OnSuccess	Success
OnFailure	Failure
fail_smime	fail_smime

S/MIME Un-Packer Service

The S/MIME un-packer service expects input in the same form, which is a MIME document without message headers. This service is configured with a register message namespace where it can find the message headers, which are added back to the document before unpacking. This message namespace must match the message namespace configured for the S/MIME packer service. Output of the S/MIME un-packer service depends on the content-type of the input. Also, considering that the S/MIME package is a flat document, the listener that will accept the S/MIME message must be configured to accept flat documents.

Procedure: How to Configure a S/MIME Un-Packer Service

To configure a S/MIME un-packer service:

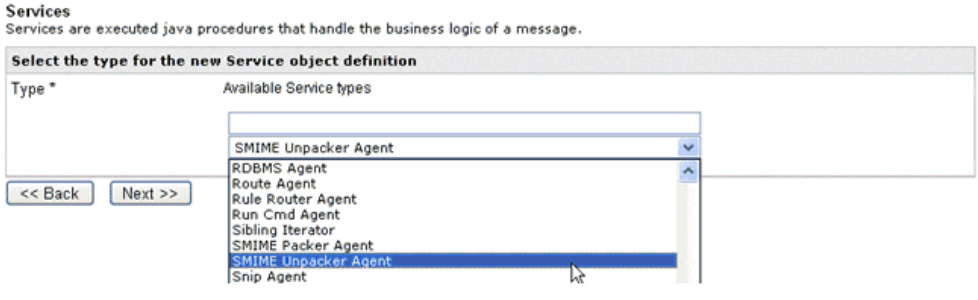
1. Click *Registry* in the top pane, and then click *Services* in the left pane.

   The Services pane opens.

   The table that is provided lists all the previously configured services and a brief description for each.

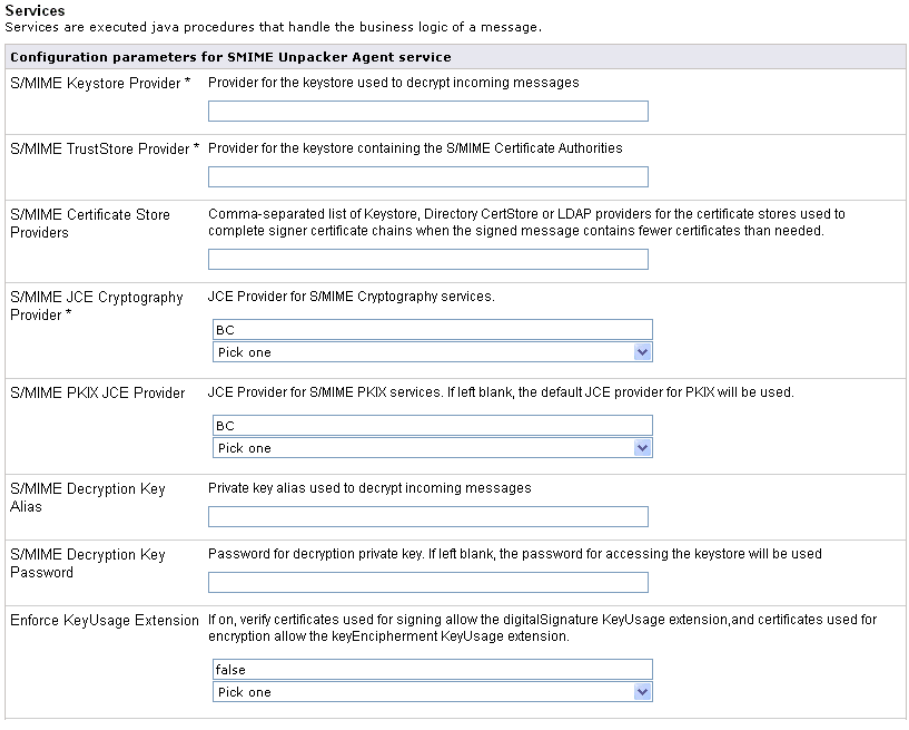
2. Click *Add*.

   The Select Service type pane opens.



3. Select *SMIME Unpacker Agent* from the Type drop-down list.
4. Click *Next*.

   The configuration parameters pane for the S/MIME un-packer service opens.

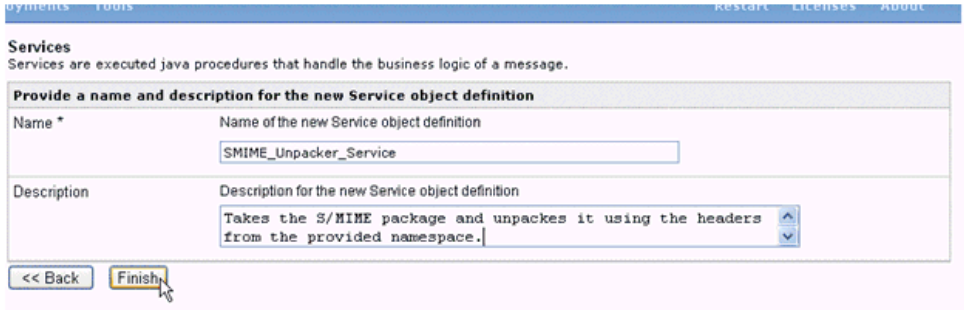


5. Provide the appropriate values for the S/MIME un-packer service parameters.

   For more information, see *S/MIME Un-Packer Service Configuration Parameters* on page 73.

6. Click *Next*.

The name and description pane opens.



7. Enter a name for the service and description (optional).
8. Click *Finish*.

Reference: S/MIME Un-Packer Service Configuration Parameters

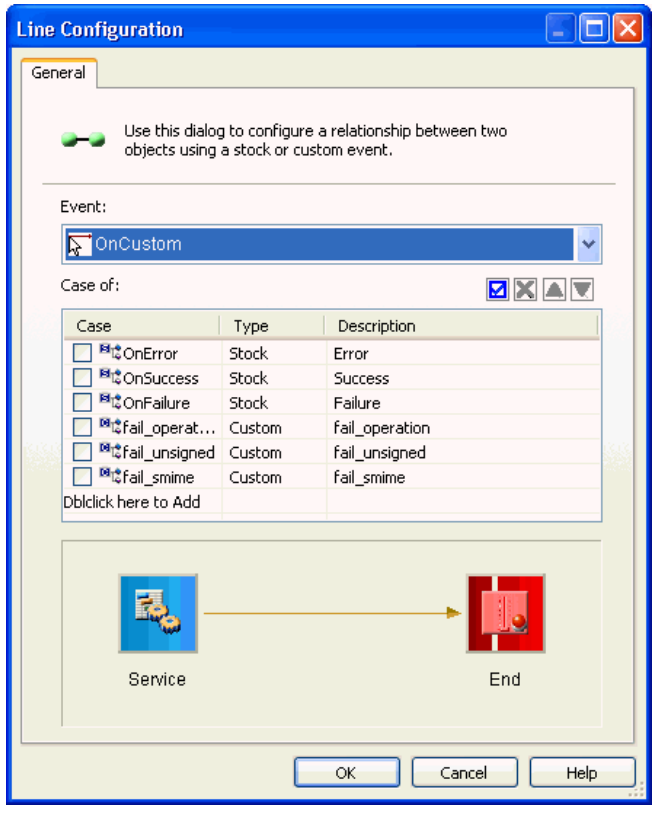
The following table lists and describes parameters for the S/MIME un-packer service.

Parameter	Description
Configuration Parameters	
S/MIME Keystore Provider	The named iWay Security provider used to decrypt incoming messages and sign receipts.
S/MIME Truststore Provider	The named iWay Security provider containing the S/MIME certificate authorities.
S/MIME Certificate Store Provider	The comma-separated list of Keystore, Directory Certstore or LDAP providers for the certificate stores used to complete signer certificate chains when the signed message contains fewer certificates than needed.
S/MIME JCE Cryptography Provider	The JCE provider for S/MIME cryptography services.
S/MIME PKIX JCE Provider	The JCE provider for S/MIME PKIX services.
S/MIME Decryption Key Alias	The private key alias used to decrypt incoming messages.

Parameter	Description
S/MIME Decryption Key Password	The password for decrypting a private key. If left blank, the password for accessing the keystore is used.
Enforce KeyUsage Extension	If set to *true*, then verify the certificates used for signing allow the digital signature KeyUsage extension, and certificates used for encryption allow the keyEncipherment and KeyUsage extension.
Enable Certificate Revocation	If set to *true*, use the CRLs from the CertStore to check whether the certificate of the signer has been revoked.
Unrecognized Cert Location	The directory to store unrecognized certificates found in S/MIME messages.
Signature Required	If set to *true*, incoming documents will require a valid signature.
Error Return	This determines which document is returned when an error occurs.
Keep Message Flat	If set to *true*, use the body of the message as an array of bytes.
Header Management	
Message Header Namespace	The special register namespace to which message headers generated by the S/MIME packaging process are stored. If it is not supplied, message headers are saved in the default namespace.
Payload Header Namespace	The special register namespace from which additional MIME headers for the payload are taken. If it is not supplied, no MIME headers are added beyond the content headers generated by the packaging process.

Available Response Edges for SMIMEUnpackerAgent

When you connect the SMIMEUnpackerAgent object to an End object using the *OnCustom* build relation in a process flow, the available line edges are provided in the Line Configuration dialog box.



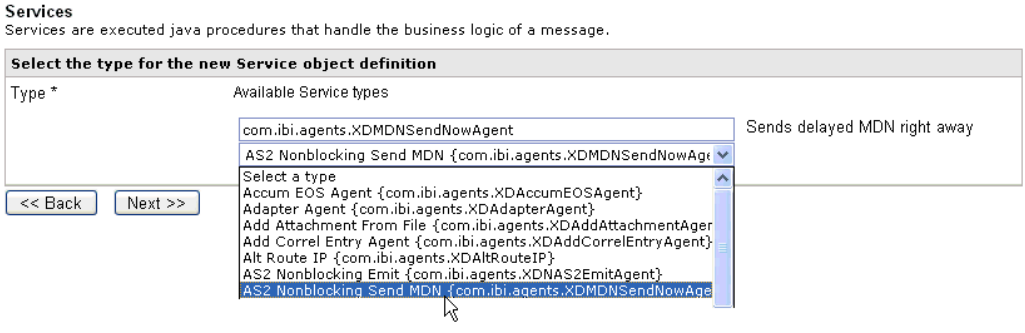
The following table lists and describes the available line edges for the SMIMEUnpackerAgent object.

Line Edge	Description
OnError	Error
OnSuccess	Success
OnFailure	Failure
fail_operation	fail_operation
fail_unsigned	fail_unsigned

Line Edge	Description
fail_smime	fail_smime

Configuring MDNSendNow Services

The MDNSendNow service is a new service that is available in the NAS2 adapter configuration.



Overview

The following MDN elements are associated and described in this section:

- **Reqns.** The Request Header namespace.
- **Respns.** The Response Header namespace.
- **Mdnns.** The MDN Field namespace.

You can override the Comment to augment the human part with additional text. Custom HTTP headers are special registers of type HDR in the Response Header namespace. Extension fields in the machine readable part are special registers of any type in the MDN field namespace, but the name must start with X- or x-.

HTTP Header Fields

AS2-From	reqns.AS2-To
AS2-To	reqns.AS-From
AS2-Version	'1.1'
Message-ID	uniquely generated

Human Part Fields

MessageID	reqns.AS2-To
From	reqns.From
To	reqns.To
Sent on	reqns.Date
Subject	reqns.Subject
Status	tail of Disposition
Error	mdnns.Error if present
Warning	mdnns.Warning if present
Failure	mdnns.Failture if present
Comment	mdnns.Comment or else default message

Machine Part Fields

Reporting-UA	Reporting User Agent parameter
Original-Recipient	reqns.To
Final-Recipient	reqns.To
Original-Message-ID	reqns.Message-ID
Received-Content-MIC	calculated MIC if available
Disposition	mdnns.Disposition or else based on Error, Warning, or Failure
Error	mdnns.Error if present
Warning	mdnns.Warning if present
Failure	mdnns.Failure if present

X-??	mdnns.X-?? if present

The MDN is formed as described in this section. Specific parameters have been modified to eliminate any limitations on field content. Fields that cannot be set based upon agent parameters can be set as described in the tables above. Currently, the use of specific special registers simplifies configuration and does not impose any functional limitations.

MDN Human Readable Part

1. Status now contains the tail of the Disposition. This is the Disposition-Modifier, if present, otherwise it is the Disposition-Type. As a result, Status now contains the value that used to be in the Error field.
2. Error field now contains the value that used to be in Detailed Error.
3. Detailed Error no longer exists.
4. There is no namespace to augment the Human Readable Part. The value of the Comment field can be overridden to add extra text. This is not a limitation, since the human readable part is unstructured.

Machine Readable Part

1. Reporting-UA is configurable as a listener parameter.
2. The Disposition can be overridden as one value by assigning it to the Disposition register. The format is:

   ```
   disp-mode; disp-type[/disp-modifier[:dispdesc]]
   ```

3. If the Disposition register is absent, but the Disposition-Modifier register is assigned, the Disposition will be computed as follows:

   a. The disposition mode is an automatic-action/MDN-sent-automatically.

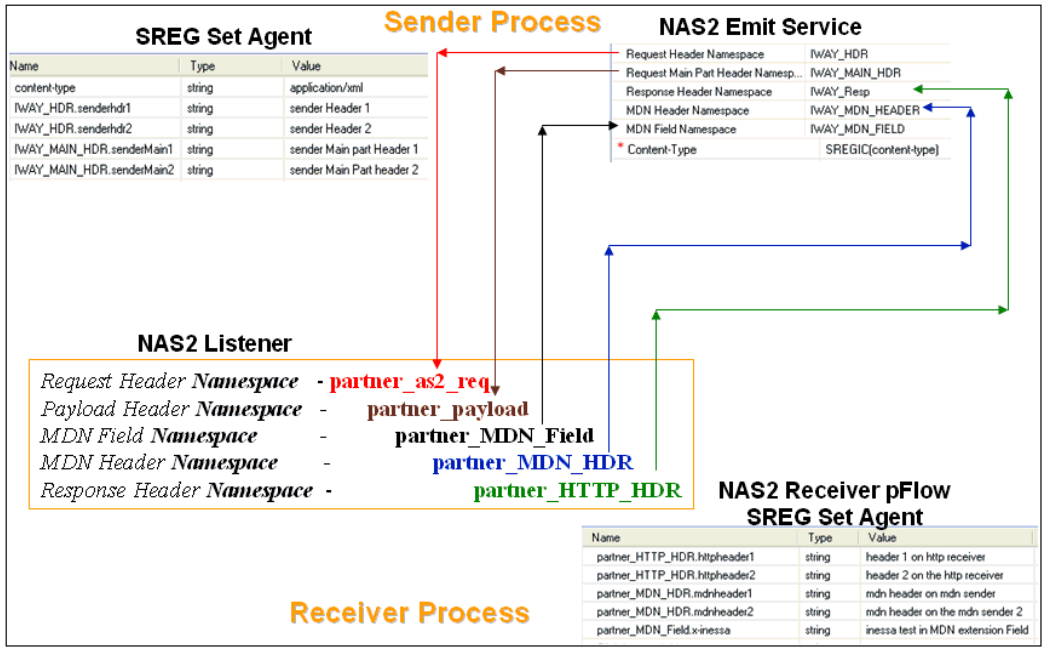
   b. The disposition modifier is specified by the register.

   c. The disposition type is computed based on the first few characters of the disposition modifier.

      If the disposition modifier starts with a failure, the disposition type is failed, otherwise it is processed. Notice that it is possible to specify the Disposition description as the tail of the Disposition-Modifier value.

4. If the Disposition and Disposition-Modifier registers are absent, the Disposition is computed as follows:
   a. The disposition mode is an automatic-action/ MDN-sent-automatically.
   b. If the Error is assigned, the disposition modifier is processed/error.
   c. If the Warning register is assigned, the modifier is processed/warning.
   d. If the Failure register is assigned, the modifier is failed.
   e. If Error, Warning, and Failure are absent, the modifier is processed.
5. The MDN Field namespace contains more than extension fields. To be recognized as an extension field, a special register in the MDN Field namespace must have a name that starts with X- or x-.

The following diagram illustrates the sender process and the resulting receiver process.



Procedure: How to Configure an MDNSendNow Service

To configure an MDNSendNow service:

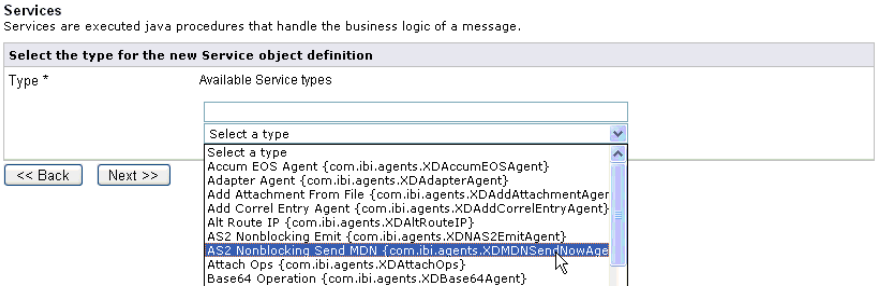
1. Click *Registry* in the top pane, and then click *Services* in the left pane.

   The Services pane opens.

   The table that is provided lists all the previously configured services and a brief description for each.

2. Click *Add*.

   The Select Service type pane opens.



3. Select *AS2 Nonblocking Send MDN* from the Type drop-down list.
4. Click *Next*.
5. Provide a name for the service and description (optional).
6. Click *Finish*.

Chapter 4

Application Protocol Adapters

This section describes how to configure and use the iWay Application Protocol adapters (for example, EDAAPI, RDBMS, MySAP, and Scheduler). It is intended for developers of integration projects involving these application packages.

In this chapter:

iWay Adapter for EDAAPI

The iWay Adapter for EDAAPI (CS3) enables you to modify existing client applications to access packaged adapters as well as terminal emulation adapters. Client applications, such as Microsoft Query, can use the iWay Connector for ODBC, which generates EDAAPI (CS3) network calls usually intended for a data server.

Note: The CS3 listeners provide access to both JDBC and non-JDBC data sources.

Using EDAAPI Calls

The iWay Adapter for EDAAPI (CS3) can accept EDAAPI calls in the same way as an iWay Server. The results are returned to the client in CS3 format or in XML format to alternate destinations.

When EDAAPI calls are received from the client, a standard iWay request document is generated automatically. The information required for the connection and query is extracted from each call and built into the request document. Requests are processed synchronously as soon as the SQL or RPC command is received from the client.

Reference: EDAAPI Listener Properties

The following table lists and describes the EDAAPI listener properties. For instructions on creating a listener, see *Configuring Listeners* on page 18.

Property Name	Property Description
Port (required)	The service port number that the listener uses to exchange messages. The iWay Service Manager configuration tool requires that the port be numeric. **Note:** The port must match the value that the EDAAPI client is using to send the CS3 calls.
Delimiter (%)	The delimiter for embedded commas. Data encapsulated by the delimiter (%data, more data, more data%) is sent to the server as a single data element that allows you to send special characters, such as commas to the server. If the delimiter is not used, the CS3 listener treats commas as separators.
Multithreading	Indicates the number of documents that can process in parallel. To enable the listener to handle a second request while an earlier request is still being processed, set this to a value greater than 1. The total throughput of a system is affected by the number of threads operating. Default: 1 Max value: 99
Agent Precedence	This changes the order in which the engine selects agents. Usually, the document overrides the listener. This is used to manage iWay documents.

Property Name	Property Description
Return Message Only	If set to *true*, this returns to an earlier behavior where an acknowledgment is returned to an application instead of an answer set.
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined replies.
Execution Time Limit	The maximum time to complete a request. Prevents runaway requests. iWay Service Manager terminates any request that takes longer than this value to complete.

LDAP Security

Using LDAP security allows the CS3 listener to confirm the incoming user name and password values against values stored in the LDAP repository. Otherwise, EDAAPI calls are processed without user validation.

To use LDAP security you must have configured an LDAP repository. For more information, see *Obtaining Configuration Properties Using LDAP* on page 373.

Secure	If selected, the listener confirms the password against the LDAP value.
iWay Client Login Field	The LDAP attribute that represents the client login.
iWay Client Password Field	The LDAP attribute that represents the client password.
Access Context	The context where the client attributes are stored.

Reference: EDAAPI Listener Special Registers

The following table lists and describes the special registers (SREGs) available on the EDAAPI listener.

Name	Source	Level	Type
iwayconfig	Listener	System	String
msgsize	Listener	Document	Integer

Name	Source	Level	Type
name	Listener	System	String
protocol	Listener	System	String
tid	Listener	Document	String

Configuring an EDAAPI Client (ODIN.CFG Node)

An EDAAPI client must be configured to connect to a server with a name that matches the name of a data server defined in the iWay Service Manager configuration.

For example, you can use the iWay Server test tool as a EDAAPI client to the iWay Service Manager EDAAPI listener. The test tool obtains a list of available servers from the odin.cfg file. The file must contain a node entry name matching the name of a data source configured to iWay Service Manager.

The following is a node in the odin.cfg file used by the EDAAPI Client:

```
NODE=IXTEJLNK
BEGIN
PROTOCOL=TCP
PORT=2222
HOST=IXTESERV
END
```

The client is configured to communicate with an EDAAPI listener, hosted by a machine called iXTESERV, listening on port 2222. IXTEJLNK must be a configured data source in iWay Service Manager.

iWay Adapter for RDBMS

The iWay Adapter for RDBMS supports integration with applications that can write to a relational table. If an application cannot send data or messages to iWay Service Manager using a queue or other standard protocol, such as TCP, iWay Service Manager can periodically retrieve the data directly from any JDBC data source. The listener reads one or more rows from the table and creates an XML document representing the column data in each row. The standard server business logic facilities are then applied to the constructed XML documents, including transformation, validation, security management, and application processing. Each row in the table is optionally deleted or updated if the business logic completed properly.

The RDBMS listener can:

- Monitor data changes by repeatedly performing an SQL query.

  The SQL listener also supports customized user exits with Java classes to define custom operations on the row sets.

- Be configured to operate one row at a time or to operate on sets of data, only sending events to a business process workflow when a specified minimum number of rows become available in the source DBMS.

- Enable the configuration of an optional SQL post-query statement, which performs certain DBMS operations after the adapter has sent the row set (formatted as XML) to a business process workflow.

  The default operation is to delete the rows that were transferred to the workflow. However, other options may include moving the rows to an archive table or marking the rows with an SQL UPDATE.

The iWay Adapter for RDBMS can function in either a destructive or non-destructive read mode. In the destructive read mode, it is presumed that all data in the table is new and is to be consumed by iWay Service Manager. This is analogous to a message in a queue that is deleted after it is read. The non-destructive read mode provides a mechanism to preserve the data and track which rows were not yet processed.

Software Requirements

The RDBMS listener requires the JDBC driver for the database being monitored. Contact your DBMS vendor to obtain the appropriate JDBC driver.

Choosing an RDBMS Listening Technique

You can detect an RDBMS event and propagate it into a workflow process using an RDBMS listener.

An elaborate polling technology enables the specification of SQL SELECT statements to be executed on a periodic basis. After data is polled, it passes through the adapter to the iWay Service Manager for further processing.

The following are techniques you can use to listen to an RDBMS event:

- **Standard technique with row tracking.** This technique listens to a table, sends each newly inserted row to the Reply-to destination, and uses a control table to track the most recently read row. The control table prevents the most recently read row from being reread during the next listening cycle.

  This is a flexible, yet simple technique that you can apply in most situations.

For more information, see *Standard Event Processing With Row Tracking* on page 86. For information about Reply-to, see the *iWay Service Manager User's Guide*.

- **Standard technique with row removal.** This technique listens to a table, sends each newly inserted row to the Reply-to destination, and then deletes the new row from the table to prevent it from being reread during the next listening cycle.

  You apply this technique when the source table is used as a conduit to pass data to the adapter, and the table rows are not required to persist. Rows are deleted as they are processed.

- **Trigger-based technique.** This technique assigns triggers to a joined group of tables, has the triggers write information about table changes to a common control table, listens to the control table, and sends the information about the table changes to the Reply-to destination. It deletes new rows from the control table to prevent them from being reread during the next listening cycle.

  You apply this technique when listening for events in a group of large joined tables, or when you need to know about updated or deleted rows.

Before you configure an RDBMS listener, register JDBC driver libraries. You must register JDBC driver libraries only once for each instance of the iWay Service Manager.

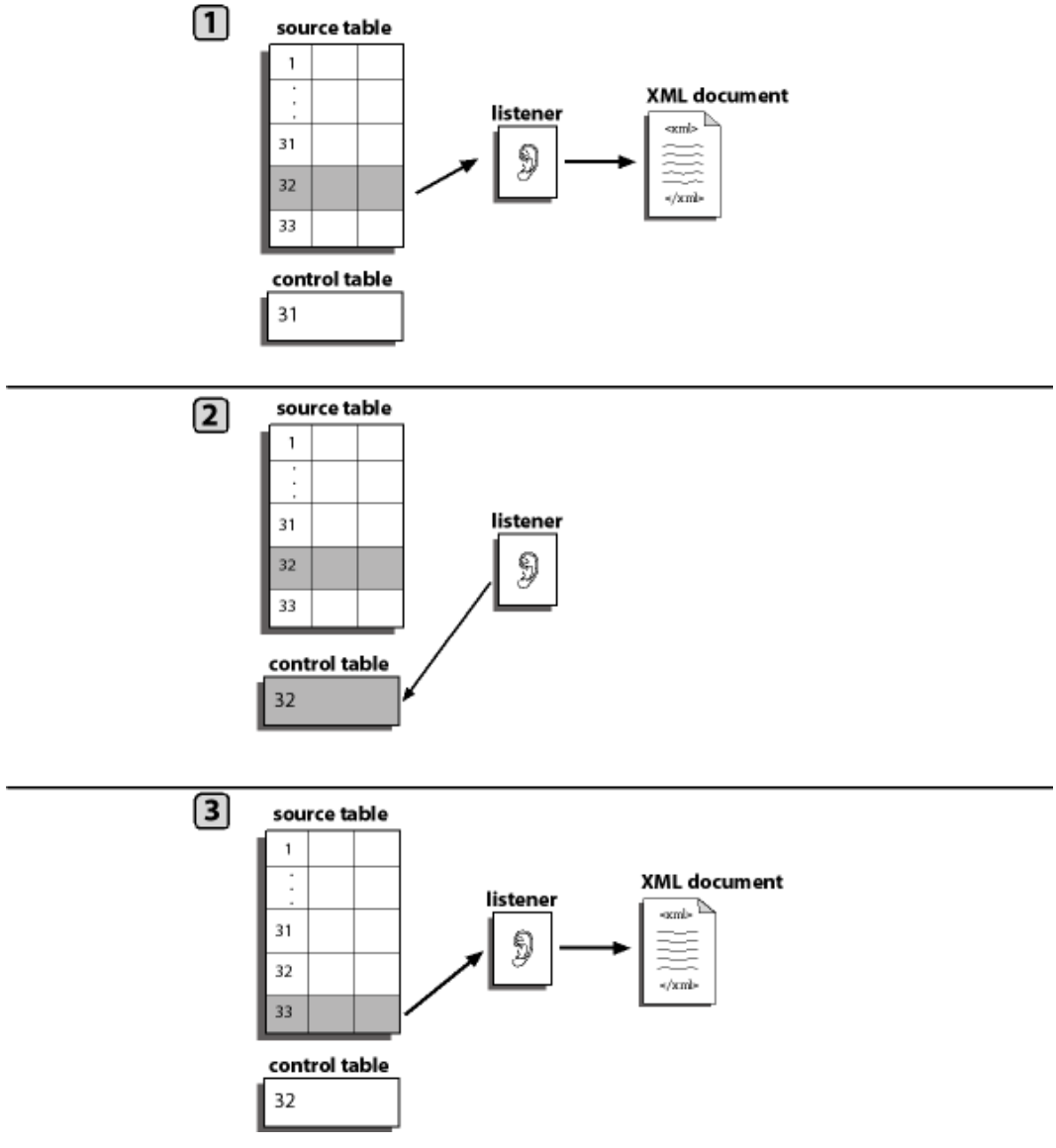
Standard Event Processing With Row Tracking

The standard event processing with row tracking technique enables you to listen to the source table without removing its rows. It requires that you create a single-row control table that tracks the last new row the listener reads from the source table.

The control table has a single column that corresponds to a column (or a group of columns) in the source table that is unique, sortable, and indicates how recently the row was added to the source table relative to the other rows. For example, the first row added to the source table has the lowest value, and the last row added has the highest value. You can consider this value the event key. When you create the control table, initialize this field to the value of the most recently added row of the source table. You configure the listener SQL Post-query property to update the event key in the control table.

Each time the listener queries the source table, it looks for rows added since the last query. That is, for rows whose event key is greater than the current value of the field in the control table. It reads each of these rows and returns them to the Reply-to destination through an XML document. To ensure that a row is not read again the next time the listener queries the table, the listener updates the field in the control table to match the value of the row just read from the source table.

The standard event processing with row tracking technique is shown in the following figure.



In this figure:

1. The listener queries the source table and copies each source table row whose event key is greater than the control table event key. The listener copies the row to an XML document that it sends to the Reply-to destination.

2. The listener then updates the event key in the control table to match the row it most recently read.
3. The listener copies the next source table row to an XML document.

The process repeats itself. To implement this event processing technique, see the following procedure.

Procedure: How to Implement Standard Event Processing With Row Tracking

To implement standard event processing with row tracking:

1. Create the control table. For an example, see *Creating the Control Table Required for an RDBMS (Oracle) Event* on page 88.
2. Configure the listener in the iWay Service Manager console.

   In addition to the required listener properties for standard event processing with row tracking, you also must provide values for the following optional properties:

   SQL Query. The SQL SELECT statement that identifies the source table to which the adapter listens and with which it queries the table.

   SQL Post-query. The SQL statements that maintain the field in the control table.

   For information on how to configure a listener, see *How to Configure the RDBMS Listener* on page 89.

Example: Creating the Control Table Required for an RDBMS (Oracle) Event

This example uses an Oracle E-Business Suite (also known as Oracle Applications) table. You can apply the same technique in a similar way to other types of relational data sources.

Follow the steps in this example to create an Oracle E-Business Suite table named TEMP_NEW_YORK_ORDER_ENTITY that has a single field named WIP_ENTITY_ID. You specify this table when you configure the RDBMS listener, as described in *How to Configure the RDBMS Listener* on page 89.

When discrete jobs are created through the Oracle E-Business Suite graphical interface, an entry is created in the WIP.WIP_DISCRETE_JOBS table. For this example, you configure an event to detect new entries to this table. Do not rely on a destructive read of this table, because Oracle E-Business Suite processing cannot delete rows from the table. To accomplish a non-destructive poll, first create a simple table to track the records processed.

1. From within Oracle SQL*PLUS, run the following SQL:

```
CREATE TABLE WIP.TEMP_NEW_WORK_ORDER_ENTITY_ID
(
   WIP_ENTITY_ID   NUMBER
)
```

This creates a single table with a single field.

Note: Oracle SQL*Plus is part of the Oracle client software. If it is not installed, contact your Oracle database administrator.

You must be logged in under the APPS schema or a similar ID that has access rights to the Oracle E-Business Suite WIP schema.

2. Create a single record in this table and provide it with the highest WIP_ENTITY_ID ID from your system. You can obtain this from the WIP.WIP_DISCRETE_JOBS table.

   This sets the value at which to start detecting events as records enter the WIP_DISCRETE_JOBS table.

After creating a simple table in Oracle, the iWay Service Manager portion of the RDBMS listener must be configured. For configuration steps, see *How to Configure the RDBMS Listener* on page 89.

Procedure: How to Configure the RDBMS Listener

This procedure describes how to configure an RDBMS listener that is used with the row tracking technique. In the scenario used to illustrate this procedure, the event propagates a message that contains record data when a new WIP discrete job is created.

Note: Ensure that you already created the table for tracking records that were processed. For more information on creating the table, see *Creating the Control Table Required for an RDBMS (Oracle) Event* on page 88.

Reference: RDBMS Listener Properties

The following table lists and describes the RDBMS listener properties. For instructions on creating a listener, see *Configuring Listeners* on page 18.

Property Value	Description

Driver	Fully-qualified name of the JDBC driver used to access the database, for example:
	`com.microsoft.jdbc.sqlserver.SQLServerDriver`
	`com.ibm.db2.jdbc.app.DB2Driver`
	`oracle.jdbc.driver.OracleDriver`
	`com.mysql.jdbc.Driver`
	`com.sybase.jdbc2.jdbc.SybDriver`
	`com.informix.jdbc.IfxDriver`

URL	The URL of the JDBC driver, for example:
	`jdbc:microsoft:sqlserver://PMSNJC:1433;DatabaseName=dbname;selectMethod=cursor`
	`jdbc:db2://host:port/DatabaseName;translatebinary=true;trace=true`
	`jdbc:oracle:thin:@myhost:1521:ORADB1`
	`jdbc:mysql://myhost:3306/inventory`
	`jdbc:sybase:Tds:host:port/DatabaseName?DYNAMIC_PREPARE=true`
	`jdbc:informix-sqli://HOST:PORT/DB:INFORMIXSERVER=SERVER_NAME`

User Name	The user ID employed to access the input table.
Password	The password for the user ID.
Table (required if SQL Query is not selected)	The name of the database table to query with the default SELECT statement if the SQL Query property is omitted. This is ignored if the SQL Query is specified.
Maximum Rows	The maximum number of rows to include in each document.

Property Value	Description

SQL Query (required if Table is not selected)	The SQL statement that the listener issues to poll the table.
	If this is omitted, it defaults to:
	`SELECT * FROM table`
	where:
	table
	Is the value of the Table property.
	Note: If the SQL statement includes a date column or long text column, you must provide a value for the SQL Post-query property or Delete Keys property, and the value you provide must not contain a date column or a long text column. This applies whether you provide an SQL statement here or rely upon the default.
	For example, the following SELECT statement retrieves all unprocessed records from the DISCRETE_JOBS table:
	`SELECT * FROM WIP_DISCRETE_JOBS DJ`
	`WHERE DJ.WIP_ENTITY_ID >`
	`(SELECT WIP_ENTITY_ID`
	`FROM WIP.TEMP_NEW_WORK_ORDER_ENTITY_ID)`

Property Value	Description
SQL Post-query	One or more SQL statements that are executed after each new record has been read from the table. If you specify more than one statement, then terminate each with a semicolon (;). Case-sensitive; the case used to specify the column names must match the case used in the SELECT statement that polled the table. If the SQL Query property was omitted so that a default SELECT statement polled the table, the case used to specify the column names must match the case used to define the columns in the DBMS native schema. If you do not specify a value for SQL Post-query, each record read from the table is deleted after it has been read. How this happens depends on whether you specify the Delete Keys property. If you: ❏ **Specify the Delete Keys property.** By default, the adapter issues a DELETE statement with a WHERE clause containing every key column specified for the Delete Keys property. At run time this is faster than if you do not specify the Delete Keys property if there is an index on the key, or if there are fewer key columns than there are columns in the SELECT statement that polled the table. ❏ **Do not specify the Delete Keys property.** By default, the adapter issues a DELETE statement with a WHERE clause that specifies every column from the SELECT statement that polled the table. You can choose to retain the table data after it is read by specifying a value for this property. Note that the SQL Post-query and Delete Keys properties are mutually exclusive, as Delete Keys applies to the default DELETE statement, and SQL Post-query overrides the default DELETE statement. You can provide a value for one or the other, but not for both. There are two field operators, ? and ^, that you can use in a Post-query SQL statement.

Property Value	Description
Delete Keys	A comma-separated list of key columns to be used in the default DELETE statement. DELETE operates on keys, so specify the key columns of the table. Case-sensitive. The case used to specify the column names must match the case used in the SELECT statement that polled the table. If the SQL Query property was omitted so that a default SELECT statement polled the table, the case used to specify the column names must match the case used to define the columns in the DBMS native schema. The Delete Keys and SQL Post-query properties are mutually exclusive, as Delete Keys applies to the default DELETE statement, and SQL Post-query overrides the default DELETE statement. You can provide a value for one or the other, but not for both. For more information, see the description of the SQL Post-query property in this table.
Generated XML format	The format can be field or column.
Include xLOBs in Tree	If set to *true*, the contents of all BLOB/CLOB are included in the tree and not as external files. BLOBs are base 64 encoded and enclosed in base64() markers. The inclusion of xLOBs in the tree can result in significant memory use.
Transaction Isolation Level	The transaction isolation level to set if possible
Root Name	The default is the listener name.
Column Limit	The Varchars exceeding this limit are carried as external files.
Whitespace Normalization	Specifies how the parser treats whitespace in Element content. Choose preserve to turn off all normalization as prescribed by the XML Specification. Choose condense to remove extra whitespace in pretty printed documents and for compatibility with earlier versions.
Accepts non-XML (flat) only	If set to *true*, the listener expects flat (non-XML). Preparsers do not run.

Property Value	Description
Optimize Favoring	The selection of memory is useful for large input document.
Multithreading	The number of documents that can be processed in parallel.
Execution Time Limit	The time limit for document execution (in seconds) before it is terminated. (Also see system property kill interval).
Polling Interval	The interval, in seconds, at which to check for new input.
Default Java File Encoding	The default encoding if the incoming message is not self-declaring (that is, XML).
Always reply to listener default	If set to *true*, then the default reply definition is used in addition to defined replies.
Error Documents treated normally	If set to *true*, the, the error documents are processed by any configured preemitters. **Note:** There is no default emitter location. You must add a protocol emitter to your channel.
Listener is Transaction Manager	If set to *true*, then the agents run within a local transaction.
Record in Activity Log(s)	If set to *true*, then the activity on this channel will be recorded in the activity logs, else the activity will not be recorded.

Reference: RDBMS Listener Special Registers

The following table lists and describes the special registers (SREGs) available on the RDBMS listener.

Name	Level	Type	Description
eos	Document	Integer	Set to 1 at the end of a set.
iwayconfig	System	String	The current active configuration name.

Name	Level	Type	Description
msgsize	Document	Integer	The physical length of the message payload.
name	System	String	The assigned name of the master (listener).
protocol	System	String	The protocol on which this message was received.
recordcount	Document	Integer	The number of records in this feed.
tid	Document	String	Unique transaction ID.

iWay Application Adapter for MySAP

MySAP ERP is the latest implementation of the SAP Enterprise Reporting and Planning product. Releases are numbered ECC (Enterprise Central Component) 5 and 6.

There are two MySAP listeners that are available for configuration in iWay Service Manager:

- MySAP-AS (Application Server)
- MySAP-MS (Message Server)

Reference: MySAP-AS Listener Properties

The following table lists and describes the MySAP-AS listener properties. For instructions on creating a listener, see *Configuring Listeners* on page 18.

Property Name	Property Description
Accepts non-XML (flat) only	If set to *true*, the listener expects flat (non-XML). Preparsers do not run.
Client (required)	The client number defined for the MySAP application for client communications.
User (required)	The valid user ID for the MySAP application.
Password (required)	The valid password defined to the MySAP server.

Property Name	Property Description
Authentication mode (required)	Specifies how the connection is validated. ❑ **Password** (default) Use the value specified in the Password field. ❑ **Logon ticket (SSO2)** Specify the User to be $MYSAPSSO2$ and pass the base64 encoded ticket as the Password property. ❑ **Logon ticket (X509)** Specify the User to be $X509CERT$ and pass the base64 encoded certificate as the password property.
System	
Gateway host (required)	The name of a MySAP Gateway server.
Gateway service (required)	The service name, which is usually a compound of the service name and system number.
Program ID of the server (required)	A unique program identifier that has been specified on the MySAP Gateway server (case-sensitive).
Application Server (required)	The host name or IP address for the computer that is hosting the MySAP application.
System number (required)	The system number defined to MySAP for client communications.
Security	
SNC mode	The flag for activating SNC.
SNC partner	The SNC name of the application server.
SNC level	The level of protection to use for the connection.
SNC name	The SNC name.

Property Name	Property Description
SNC library path	The path and file name of the external library.
Advanced	
IDOC Format	Specifies the IDOC format. ❏ XML (default) ❏ XML-CDATA-ENVELOPED ❏ NATIVE-IDOC
IDOC release	The version in which the IDOC definition was released.
IDOC release provider	Specifies where the adapter will get the release information. ❏ **IDOC DOCREL field** (default) Uses the information in the IDOC header. ❏ **SAP release** Retrieves the information from the user account logon. ❏ **user input** Uses the value in the IDOC release field to retrieve the information.
SAP trace	Traces SAP Java connector and RFC libraries. This option is disabled by default.
Processing Mode	Specifies the processing mode to use for your channel. ❏ REQUEST (default) ❏ REQUEST_RESPONSE
Execution Time Limit	The time limit for document execution (in seconds) before it is terminated.

Property Name	Property Description
Multithreading	Indicates the number of worker threads (documents or requests) that iWay Service Manager can handle in parallel. A value greater than 1 enables the listener to handle a second request while an earlier request is still being processed.
Maximum threads	Parallel threads can grow to this count automatically on demand.

Note: An asterisk in the console indicates a required property.

Reference: MySAP-MS Listener Properties

The following table lists and describes the MySAP-MS listener properties. For instructions on creating a listener, see *Configuring Listeners* on page 18.

Property Name	Property Description
Accepts non-XML (flat) only	If set to *true*, the listener expects flat (non-XML). Preparsers do not run.
Client (required)	The client number defined for the MySAP application for client communications.
User (required)	The valid user ID for the MySAP application.
Password (required)	The valid password defined to the MySAP server.
Authentication mode (required)	Specifies how the connection is validated. ❑ **Password** (default) Use the value specified in the Password field. ❑ **Logon ticket (SSO2)** Specify the User to be $MYSAPSSO2$ and pass the base64 encoded ticket as the Password property. ❑ **Logon ticket (X509)** Specify the User to be $X509CERT$ and pass the base64 encoded certificate as the password property.

Property Name	Property Description
System	
Gateway host (required)	The name of a MySAP Gateway server.
Gateway service (required)	The service name, which is usually a compound of the service name and system number.
Program ID of the server (required)	A unique program identifier that has been specified on the MySAP Gateway server (case-sensitive).
Message Server (required)	The host name or IP address for the computer that is hosting the MySAP application.
R/3 name (required)	The system ID of the MySAP system.
Server group	The logon group of the message server. This value is case-sensitive.
Security	
SNC mode	The flag for activating SNC.
SNC partner	The SNC name of the application server.
SNC level	The level of protection to use for the connection.
SNC name	The SNC name.
SNC library path	The path and file name of the external library.
Advanced	
IDOC Format	Specifies the IDOC format. ❏ XML (default) ❏ XML-CDATA-ENVELOPED ❏ NATIVE-IDOC
IDOC release	The version in which the IDOC definition was released.

Property Name	Property Description
IDOC release provider	Specifies where the adapter will get the release information. ❑ **IDOC DOCREL field** (default) Uses the information in the IDOC header. ❑ **SAP release** Retrieves the information from the user account logon. ❑ **user input** Uses the value in the IDOC release field to retrieve the information.
SAP trace	The Traces SAP Java connector and RFC libraries. This option is disabled by default.
Processing Mode	Specifies the processing mode to use for your channel. ❑ REQUEST (default) ❑ REQUEST_RESPONSE
Execution Time Limit	Time limit for document execution (in seconds) before it is terminated.
Multithreading	Indicates the number of worker threads (documents or requests) that iWay Service Manager can handle in parallel. A value greater than 1 enables the listener to handle a second request while an earlier request is still being processed.
Maximum threads	Parallel threads can grow to this count automatically on demand.

Note: An asterisk in the console indicates a required property.

Operating Modes

The adapter supports two operating modes:

- **Design** (Metadata browse and schema creation valid for service or event configuration).
- **Run Time** (Service or Event).

Communication Modes

The adapter supports two communication modes:

- **Service**
  - Inbound to SAP - Outbound from App Server
  - RFC Client
  - SAP JCO Connection Pooling supported
- **Event**
  - Outbound from SAP - Inbound to App Server
  - RFC Server
  - SAP JCO multithreading supported

The adapter offers metadata browsing and schema creation for services and events. The service mode is used for all metadata functions.

Run-time Interface and Document Style

The adapter supports the following application interfaces and document styles as both Service and Event:

- Business Application Interfaces (BAPI) XML format.
- Remote Function Calls (RFC) XML format.
- ALE Intermediate Document (ALE IDOC) XML and native ALE/IDOC (flat).

The RFC API wrapped by the SAP JCO is the supported interface type.

Service Operation

The adapter makes a service connection to the ERP instance through the JCO. The standard method of operation for service operation is to browse the metadata of an interface object (BAPI/RFC/ALE) and to create an XML schema from the selected object. The XML schema is then used to create an XML instance populated with application specific data to produce a correct result set. If the results are incorrect the adapter produces either a java exception or an error document, depending on user selection.

Event Operation

Using the JCO, the adapter accepts a registered server program connection from the SAP gateway. The adapter then listens for events sent to the adapter destination, and processes the document. The adapter can operate in either synchronous or asynchronous event mode. In synchronous mode, the adapter passes off the results to a user defined function or adapter on the local application server, waits for the result, and passes the result back to MySAP ERP. In asynchronous event mode, the adapter produces a document with the data sent to it by the MySAP ERP application module.

Transaction Modes

The adapter supports the SAP transactional Remote Function call protocol (TRFC). The transaction is automatic for each service or event operation. The adapter can also support multiple ALE transactions in one call (Collected IDOC) either Service or Event. Multiple BAPI/RFC calls can be embedded in a single document to achieve a sequence of events in one transaction.

Schedule Listener

Scheduling provides a way to automatically start a process at a predefined interval or time.

The Schedule listener automates routine tasks and database maintenance so they occur without manual intervention. It automatically calls an iWay-supplied agent or a user coded custom agent at a predefined interval. The custom agent then executes the required business logic.

Schedule Listener Properties

The following table lists and describes the Schedule listener properties. For instructions on creating a listener, see *Configuring Listeners* on page 18.

Property Name	Property Description
Execution Interval (required)	Frequency in seconds that the agent is triggered. (This is a standard duration expression.)
Pass Document	Name of the XML document to pass. If omitted, <clock/> is passed.
Time Zone	Time zone to use.

Property Name	Property Description
Start Time	Time when the document can first be sent (hh:mm:ss, using a 24-hour clock).
End Time	Time when the document can no longer be sent.
Monday through Sunday	Indicate the days on which the schedule is active.
Send Document on Start	Select whether to send the document when the listener starts or wait until the next interval.
Accepts non-XML (flat only)	Select this option if you expect non-XML input. XML input can still be passed to the listener if the setting is enabled.
Multithreading	Indicates the number of worker threads. (Equivalent to the number of requests that iWay Service Manager can handle in parallel.) Setting this to a value greater than 1 enables the listener to handle a second request while an earlier request is still being processed. The total throughput of a system can depend on the number of threads operating. Default: 1 Max value: 99
Listener is Transaction Manager	If true, agents run within a local transaction managed by the listener. This property is set to false by default.
Execution Time Limit	Maximum time it takes to complete a request. Prevents runaway requests. iWay Service Manager attempts to terminate a request that takes longer to complete.
Record in Activity Log(s)	If set to *true*, activity on this channel will be recorded in the activity logs. If set to *false*, the activity will not be recorded.

Reference: Schedule Listener Special Registers

The following table lists and describes the special registers (SREGs) available on the Schedule listener.

Name	Source	Level	Type	Description
iwayconfig	Listener	System	String	Current active configuration name
name	Listener	System	String	Assigned name of the master (listener)
protocol	Listener	System	String	Protocol on which this message was received
tid	Listener	Document	String	Unique transaction ID.

Daily Scheduler Activation

The Schedule listener can activate every 24 hours.

The Schedule listener works by checking to see if the current day is not active (set to true). If it is not active then it will go back to sleep for a set amount of seconds known as the Execution Interval.

If Start Time is less than current time or current time is greater than End Time, it returns to sleep for the amount of time (seconds) specified for the Execution Interval parameter.

Note: Because the listener has an Execution Interval of 50 minutes and the time that the listener starts has to be considered, there is a window of 9 minutes 59 seconds (between 01:50:00 and 01:59:59) where the Schedule listener could execute twice.

The current Schedule listener can activate every 24 hours. However, that 24 hours will start right after the Schedule listener is started. If the user is looking to have the listener activate at the same time each day, then the Schedule listener, along with the process flow, needs to be developed. The Schedule listener is configured to activate along a set period of time, depending on how it was configured. The process flow would then check the time of day against the time of day configured in a special register. If the current time of day is greater or equal to the special register value, then a channel start can be executed.

The time that the listener activates is dependent on the time it was started. So if the listener was started at 12:01 am on a Monday, the Monday flag is checked to see if the listener can start. If Monday is set, the listener starts processing. When the processing completes, the listener then goes to sleep for however many seconds it is set to (84400 seconds if a day delay is wanted). The Schedule listener then activates on Tuesday at 12:01 am. It checks the Tuesday flag and does not start if Tuesday is not set. The whole process repeats for the rest of the days of the week, but only at 12:01 am on those days.

Note: If iWay Service Manager is being stopped and restarted between 01:50:00 and 02:50:00 and the Send Document on Start parameter is set to false, then the listener will wait 50 minutes and miss the window that the user has set.

iWay Application Protocol Adapter for LDAP

The iWay Application Protocol Adapter for LDAP provides connectivity to LDAP compliant servers. Connections are made using an LDAP URL and port number. The adapter also provides a means to exchange data between LDAP servers and third-party application, database, or external business partner systems.

The LDAP listener provides the ability to extract events from the LDAP directory using:

- A directory sync protocol.
- An asynchronous event notification.

For more information on configuring and using the LDAP listener, see the *iWay Application Protocol Adapter for LDAP User's Guide*.

iWay Hot Backup Listener

The iWay Hot Backup listener is a distributed backup facility. In distributed backup iWay Service Manager (iSM) can be configured to operate normally with its own compliment of listeners, and also to carry a configuration of listeners and services defining work to be done in the event that a monitored iSM fails. In this configuration, the iSMs share the workload in normal operation and in the event of unanticipated failure of one iSM the other can invoke the required services to accomplish the tasks of the failing iSM.

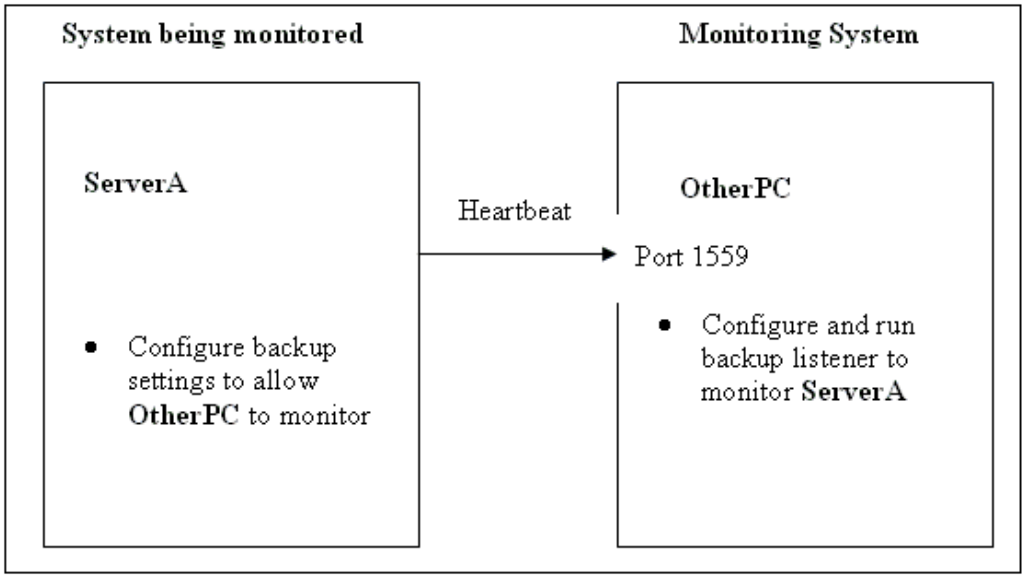
For example assume that there are two iSM instances in the enterprise. The first is configured with four active listeners, A, B, C and D. The second has its own listeners E, F, G and H. In this case, the first iSM instance would also be configured with E, F, G and H, but these would be marked as inactive. Similarly, the second iSM would be configured with A through D, marked inactive. As these two iSMs run, each handles its own four listeners. In the event of failure of the either iSM, the other marks the inactive listeners as active and begins to handle the failing workload.

To accomplish this, install the Hot Backup listener, available from iWay Software. This adds a new listener type, Backup, to the list of available listeners. The Backup listener must be configured on the iSM that is to handle the monitoring and recovery of the four listeners. No additional software is required on the iSM being monitored. The iSM being monitored needs to point to the server on which the Backup listener is running through the backup settings screen on the console.

The Hot Backup listener can monitor multiple configurations. Each configuration being monitored specifies the host name and port number of the Hot Backup listener. The generated state signal delineates the host and configuration that was being monitored when its state has changed.

Application designers should be aware that when multiple servers are being monitored, it is possible for one or more servers to fail simultaneously (for example, a section of the network failing), which will trigger more than one copy of the backup process flow. Also, application designers are reminded that a process flow can be selected based on routing, which examines the host or configuration attributes.

You can use the following diagram for reference purposes during the configuration stages of the iWay Hot Backup listener.



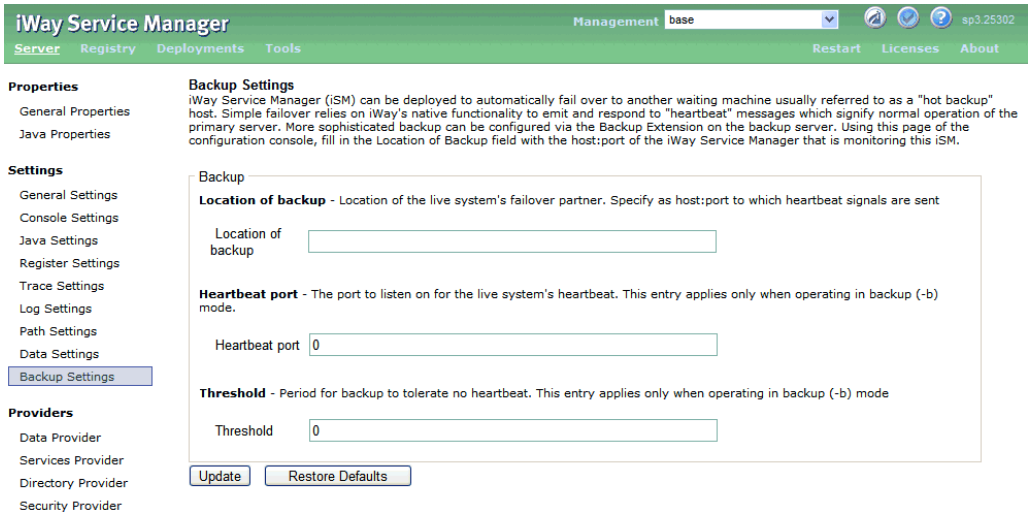
Configuring Backup Settings

To access the backup settings pane, click Backup Settings, under the Server section in the iSM

In the Backup Settings page, provide a value for the Location of Backup field with the host name and port number of the iSM that is monitoring this instance of iSM.

iSM can be deployed to automatically failover to another waiting machine usually referred to as a hot backup host. Simple failover relies on the native functionality of iWay to emit and respond to heartbeat messages which signify normal operation of the primary server. More sophisticated backup can be configured through the Backup listener on the backup server.

Using the Backup Settings page, provide a value for the Location of Backup field with the host name and port number of the iSM that is monitoring this instance of iSM, as shown in the following image.



The following table lists and describes the parameters that are found in the Backup Settings page:

Parameter	Description
Location of backup	Location of the live system failover partner. This is the server that would be monitoring the iway server. Specify the host name and port number (host:port) of the werver to which heartbeat signals are sent. The port should match the port on the backup listener configured on the system which is monitoring this server.
Heartbeat port	The port to listen on for the live system heartbeat. This entry applies only when operating in backup (-b) mode.

Parameter	Description
Threshold	Period for backup to tolerate no heartbeat. This entry applies only when operating in backup (-b) mode.

In this example, this iSM is being monitored by a computer named otherbox on port 1559. Do not fill in the other fields since these fields are not considered for the backup listener, which is being explained here. These fields instruct the iSM that it is a full backup, and prevents any processing until a monitored signal on the heartbeat port detects a failure. This means that distributed backup cannot operate on a computer that is monitoring for full backup.

While the iSM is operating, a heartbeat is sent to the backup iSM. In this case it is sent to otherbox.

Configuring the Backup Listener

The backup listener on the otherbox machine performs the task of monitoring the iWay server, serverA. To participate in distributed backup, a backup listener needs to be added and configured. The listener properties are explained below:

Property Name	Property Type	Property Description
Port*	integer	Port(socket) number on which messages are exchanged. This port number must match the port number of the server that is defined in the Location of backup field in the Backup Settings pane.
Local Bind Address	string	Local bind address for multi-homed hosts: usually leave empty. **Note:** The Local bind Address is used for multiple homed networks, and is not further considered here.

Property Name	Property Type	Property Description
Threshold	integer	Period (seconds) for backup to tolerate no heartbeat. For instance, if the heartbeat were set to 6 (seconds), then the listener would wait for 6 seconds to generate a signal say, DEAD if the server did not return any response (heartbeat). The default is 60 seconds. The time needs to be set as per the network speed requirements. Refer Notes below.
Whitespace Normalization	choice	Specifies how the parser treats white spaces in Element content. Choose preserve to turn off all normalization as prescribed by the XML Specification. Choose condense to remove extra white spaces in pretty printed documents and for compatibility with earlier versions.
Record in Activity Log(s)	boolean	If set, activity on this channel will be recorded in the activity logs, else the activity will not be recorded.

Note: A heartbeat is generated every tenth of a second by the monitored iSM and sent to the monitoring server. When the server being monitored does not send a heartbeat, even after the period specified by the Threshold value, the monitoring server sends a final message to reconfirm that the server being monitored is DEAD. In the event of not receiving a response, the monitoring server determines that the server is DEAD and generates a message accordingly.

The listener generates messages that can be handled by a normal iSM workflow or workflows of varying complexity. The dispatched messages are in XML, and take the form:

```
<signal host='hostname' config='configname'>state</signal>
```

where:

hostname

Is the name of the system where iSM is running.

configname

Is the name of the iSM configuration being used (for example, base).

state

Can be LIVE, DEAD, or CLOSE.

The following table describes each state in more detail.

State	Description
LIVE	The monitored iSM has gone live. This message appears when the monitored iSM resumes or begins operation. If the monitoring iSM starts first, this message will be dispatched when the monitored iSM starts.
DEAD	The monitored iSM has not sent a heartbeat for the threshold duration, and the monitored iSM was previously in a LIVE state.
CLOSE	The monitored iSM has signaled that it is terminating normally. If it restarts you will receive a LIVE message. Once a monitored iSM signals CLOSE, missed heartbeats are ignored: heartbeats are monitored only when the monitored iSM is in a LIVE state.

The dispatched messages are used to process the signals coming from the monitored iWay server. This can be done by:

- Using the control agent (XDControlAgent) your workflow can post internal messages to start or stop other named listeners on the monitoring server. In this manner the monitoring iSM controls the startup and shut down of listeners on its own server as the state of the iSM being monitored changes. The agent accepts a list of configured listener names and posts a start or stop message, as selected in the Action parameter.

- Using an email emit agent to send email notifications when the server comes down or starts up.

You are not limited to using the control agent. Any form of messaging system that can alert a manager about a change in status or request to perform specific actions will suffice.

Selecting the appropriate action to take based on the dispatched message(s) is accomplished by standard iSM routing and switching control, based on an analysis of the incoming message.

Configuring the Backup Listener

The backup listener on the otherbox machine performs the task of monitoring the iWay server, serverA. To participate in distributed backup, a backup listener needs to be added and configured. The listener properties are explained below:

Property Name	Property Type	Property Description
Port*	integer	Port(socket) number on which messages are exchanged. This needs to be the same as the one set under host:port under backup settings.
Local Bind Address	string	Local bind address for multi-homed hosts: usually leave empty. **Note:** The Local bind Address is used for multiple homed networks, and is not further considered here.

Property Name	Property Type	Property Description
Threshold	integer	Period (seconds) for backup to tolerate no heartbeat. For instance, if the heartbeat were set to 6 (seconds), then the listener would wait for 6 seconds to generate a signal say, DEAD if the server did not return any response (heartbeat). The default is 60 seconds. The time needs to be set as per the network speed requirements. Refer Notes below.
Whitespace Normalization	choice	Specifies how the parser treats white spaces in Element content. Choose preserve to turn off all normalization as prescribed by the XML Specification. Choose condense to remove extra white spaces in pretty printed documents and for compatibility with earlier versions.
Record in Activity Log(s)	boolean	If set, activity on this channel will be recorded in the activity logs, else the activity will not be recorded.

Note: A heartbeat is generated every tenth of a second by the monitored iSM and sent to the monitoring server. When the server being monitored does not send a heartbeat, even after the period specified by the Threshold value, the monitoring server sends a final message to reconfirm that the server being monitored is DEAD. In the event of not receiving a response, the monitoring server determines that the server is DEAD and generates a message accordingly.

The listener generates messages that can be handled by a normal iSM workflow or workflows of varying complexity. The dispatched messages are in XML, and take the form:

```
<signal host='hostname' config='configname'>state</signal>
```

where:

hostname

Is the name of the system where iSM is running.

configname

Is the name of the iSM configuration being used (for example, base).

state

Can be LIVE, DEAD, or CLOSE.

The following table describes each state in more detail.

State	Description
LIVE	The monitored iSM has gone live. This message appears when the monitored iSM resumes or begins operation. If the monitoring iSM starts first, this message will be dispatched when the monitored iSM starts.
DEAD	The monitored iSM has not sent a heartbeat for the threshold duration, and the monitored iSM was previously in LIVE state.
CLOSE	The monitored iSM has signaled that it is terminating normally. If it restarts you will receive a LIVE message. Once a monitored iSM signals CLOSE, missed heartbeats are ignored: heartbeats are monitored only when the monitored iSM is in LIVE state.

The dispatched messages are used to process the signals coming from the monitored iWay server. This can be done by:

- Using the control agent (XDControlAgent) your workflow can post internal messages to start or stop other named listeners on the monitoring server. In this manner the monitoring iSM controls the startup and shut down of listeners on its own server as the state of the iSM being monitored changes. The agent accepts a list of configured listener names and posts a start or stop message, as selected in the Action parameter.

- Using an email emit agent to send email notifications when the server comes down or starts up.

You are not limited to using the control agent. Any form of messaging system that can alert a manager about a change in status or request to perform specific actions will suffice.

Selecting the appropriate action to take based on the dispatched message(s) is accomplished by standard iSM routing and switching control, based on an analysis of the incoming message.

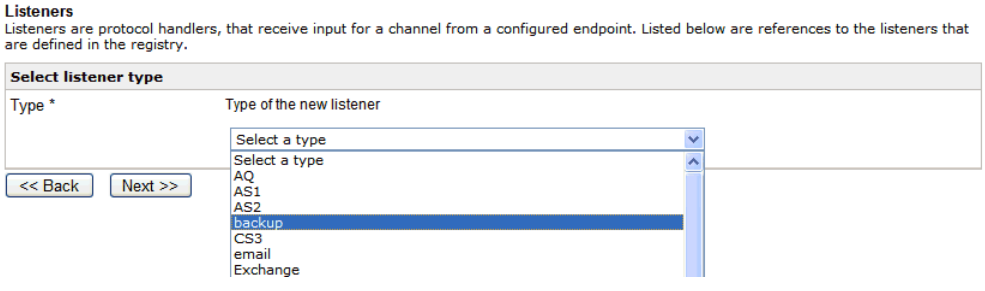
Procedure: How to Create a Backup Listener

To create a Backup listener:

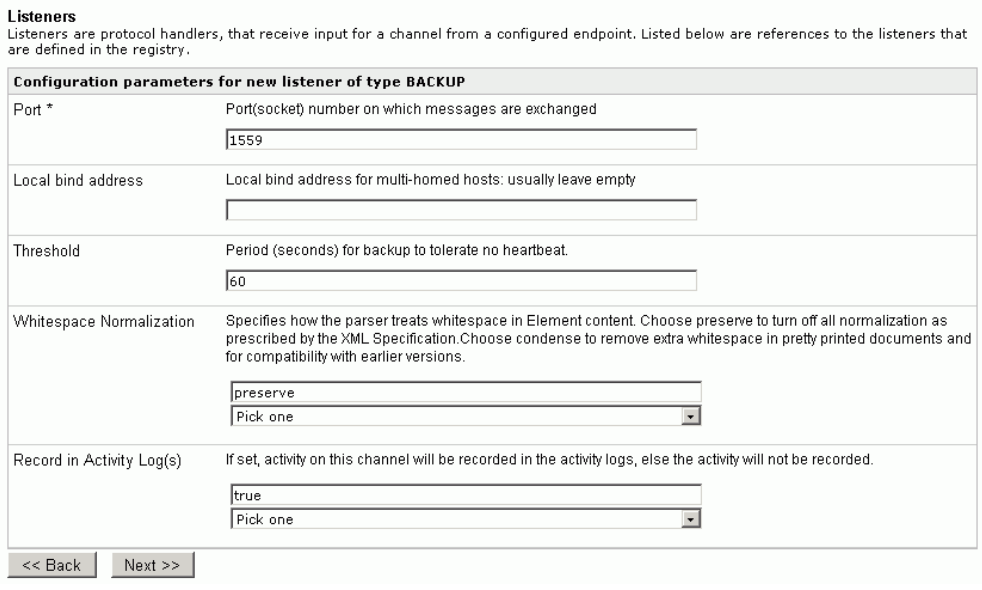
1. Start the iSM.
2. Once the iSM is open, click *Registry* in the top frame.

   The Registry section is displayed.
3. Under the Components section on the left side, select *Listeners*.

   The Listeners pane is displayed.
4. Click *Add*, to add a new listener.
5. Select *backup* from the Type drop-down list, as shown in the following image.



Configuration parameters for the listener are displayed.



6. Provide values for the configuration parameters according to the following table below and click *Next*.

Parameter	Value
Port *	1559
Local bind address	Do not specify a value for this parameter
Threshold	60
Whitespace Normalization	preserve
Record in Activity Log(s)	true

7. Enter the name of the listener as *Backup* and leave the description empty.
8. Click *Finish* to create the listener.

   The backup listener is created.

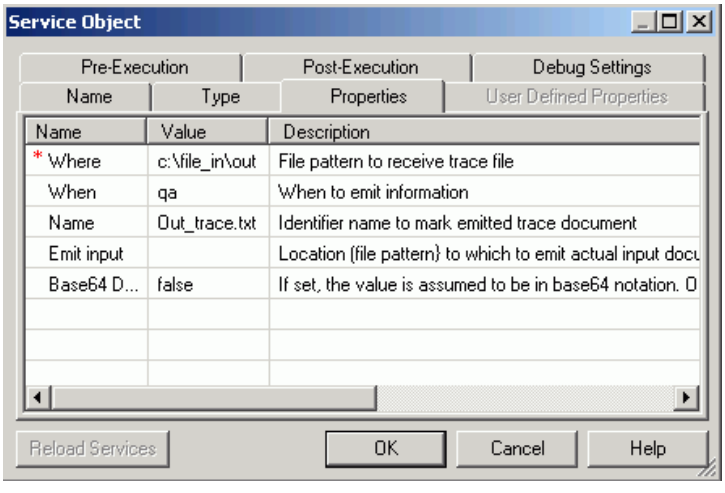
Procedure: How to Configure the Backup Listener Channel and the System Being Backed Up

This section describes how a backup listener can be used to monitor a system.

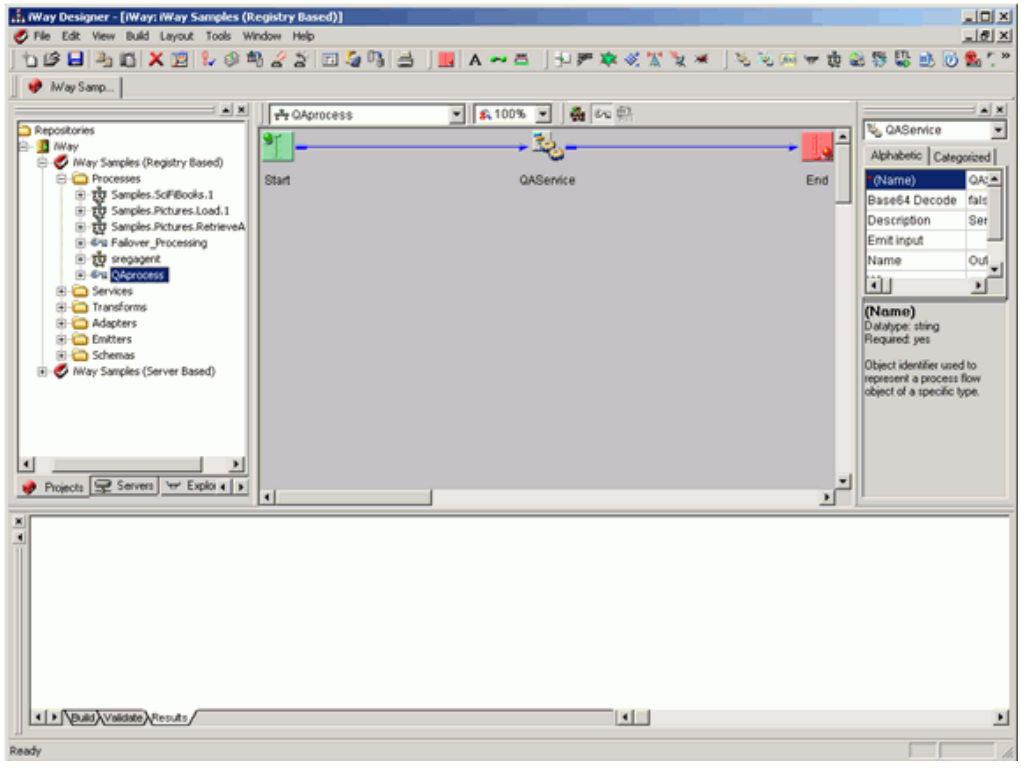
1. Construct an inlet consisting of a backup listener on the machine, which is performing the backup.

   L=t the backup listener be configured on the machine, which is performing the backup.

2. Construct a process flow which consists of a QAAgent configured with the properties in the following image.



3. Add the process flow, QAprocess, to a route called QAroute, as shown in the following image.



4. Construct a channel called testbackup, which consists of the inlet QAroute and a default outlet.
5. Build and deploy the channel.
6. Start the channel.
7. Let ServerA be the machine which needs to be backed up.
8. Stop ServerA then Start it from the command line.
9. Restart the server using the restart link in the iSM console.

   The channel, testbackup, on the server that is taking the backup, otherpc, processes a message upon server stop and processes another message when the server is started. The parameters of the messages exchanged can be viewed using the trace files created by the QAAgent. For example, the following entries in the trace files created by the QA Agent show the signal state.

```
<?xml version="1.0" encoding="UTF-8" ?><debug>Out_trace.txt
<document> <signal errors="0">CLOSE</signal> </document>
<?xml version="1.0" encoding="UTF-8" ?><debug>Out_trace.txt
<document> <signal errors="0">LIVE</signal> </document>
```

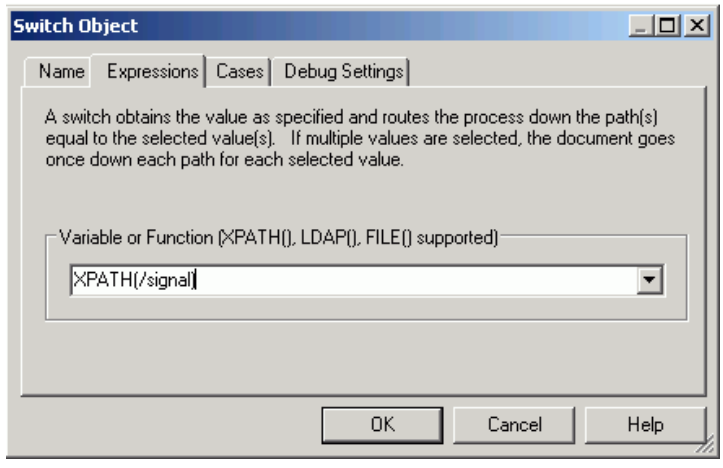
The message can also be processed using xpath(/signal) as described in the next section. Similarly, the channel processes messages when the restart option is clicked from the console.

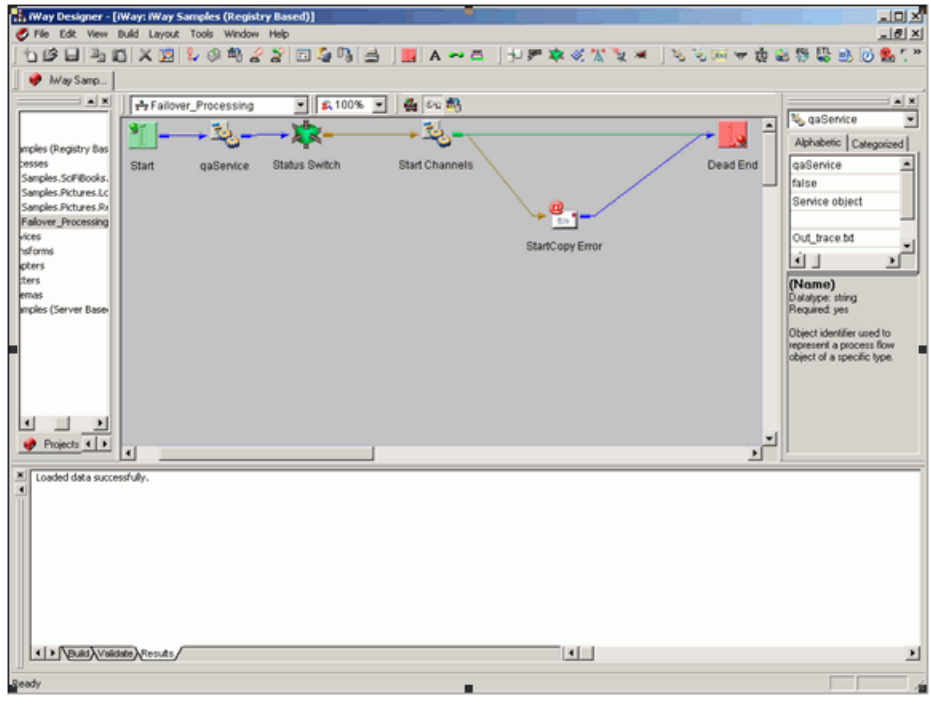
Monitoring Server State Through the Backup Listener

As the state of the monitored iSM changes, the monitoring iSM dispatches messages.

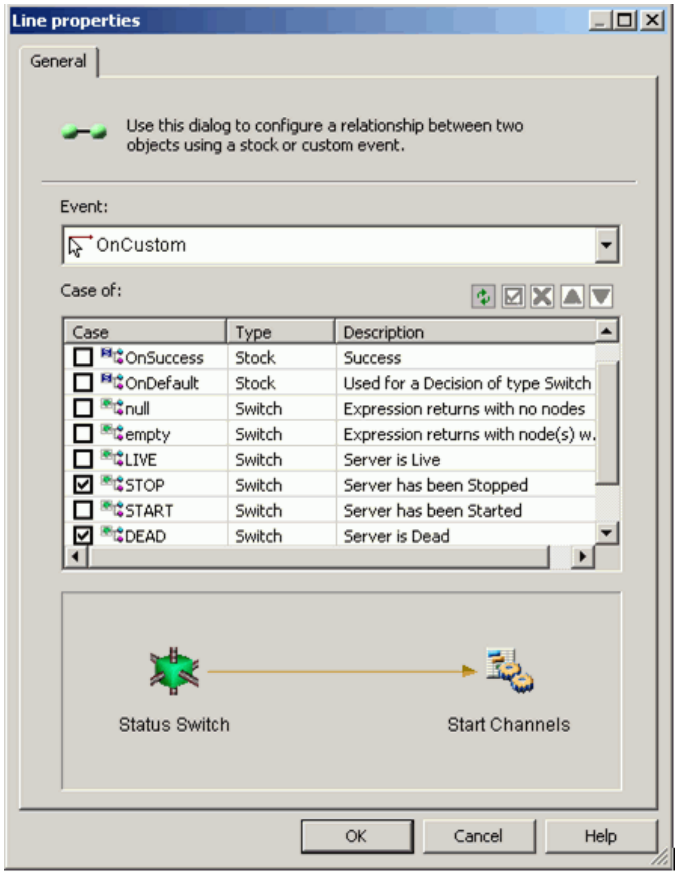
For instance, consider the below process flow, where the control agent is invoked based on the state of the monitored iSM using a decision switch. The switch is based on a signal element in the XML message generated by the backup listener.

The Status switch is a decision based object, which is configured as follows:

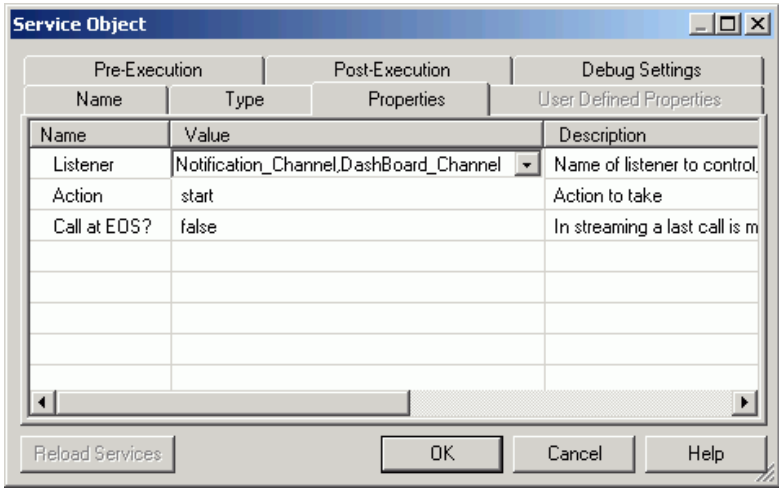




The control agent Start Channels is invoked when the listeners need to be started, that is, when they are no longer running on the configured iSM. The edges are configured as follows:



In the following example, the control agent processes the signal DEAD and STOP and accordingly stops the channels (Notification_Channel and DashBoard_Channel) as required by the monitoring backup application (backup listener). The control agent Start Channels is configured, as shown in the figure below.



Procedure: How to Start the Notification and Dashboard Channels

This example will show you how to start the channels (Notification_Channel and DashBoard_Channel).

1. Start iWay Service Manager.
2. Publish the process flow (Failover_Processing).
3. Add the process flow (Failover_Processing) to a route, and call it Failover_Route.
4. Configure the Backup listener to monitor the iWay server desired.
5. Add the listener to an inlet called backupinlet.
6. Click *Registry* in the top pane.
7. Click *Channels* under the Conduits section.
8. Build, Deploy, and start the channel backup.
9. Stop the monitored iWay server.

   The channels (Notification_Channel and DashBoard_Channel) are started when the monitored application is stopped and the signal is generated by the backup listener. The backup listener is monitoring ServerA.

Hot Backup Use Cases

The use case in this section describes how to add distributed Hot Backup capability to provide automatic failover for the primary server.

An iSM server instance must be able to perform automatic failover to a backup iSM server instance in the event of failure. It is also desirable for the backup server to be able to be actively running to perform other server tasks to maximize server resources while it is also waiting to pick up the workload of a failed server.

In this scenario, there are two deployed channels that contain two process flows:

- One process flow using a File Listener to pick up a batch of files containing financial records and then transform the records to SWIFT XML files.
- Second process using a File Listener to pick up the SWIFT XML files and then transform them to standard SWIFT FIN formats.

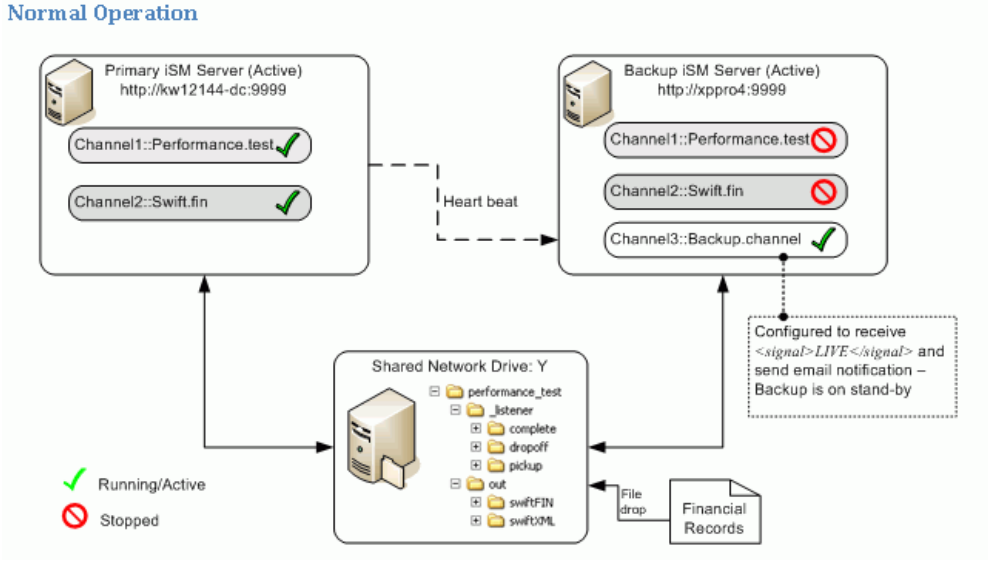
Note: These channels are deployed to the primary iSM server instance and are in their normal Started and Active states.

In the event of a failure:

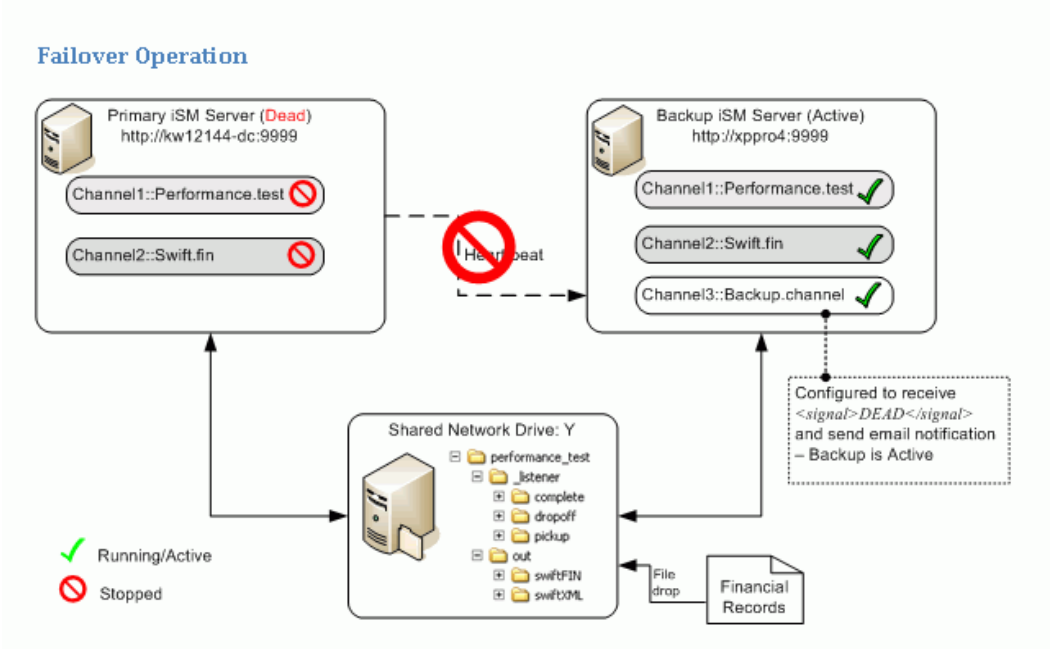
- An iSM server instance is installed on a different machine with the same configuration as the primary. This server is designated as the backup iSM server instance.
- Deploy the same channels to the backup server but put them in the Stopped and Inactive states.
- Deploy a Backup listener.
- Finally, update the primary server to include a Backup setting that points to the backup server.

Hot Backup Use Case Architecture

The following diagram illustrates the normal operation of a primary iSM server.



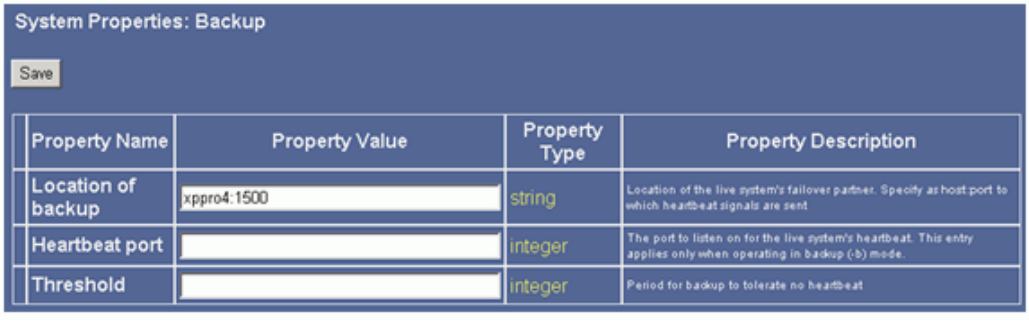
The following diagram illustrates the failover operation when the primary iSM server has failed.



Configuring the Primary iSM Server Instance

To navigate to the Backup system properties, go to the iAM console (blue console). Click *Configuration*, *System Properties*, then *Backup*.

At this screen you can enter the name of the backup server and its port number. Port number is the port specified in the Backup Listener configured on the backup server.

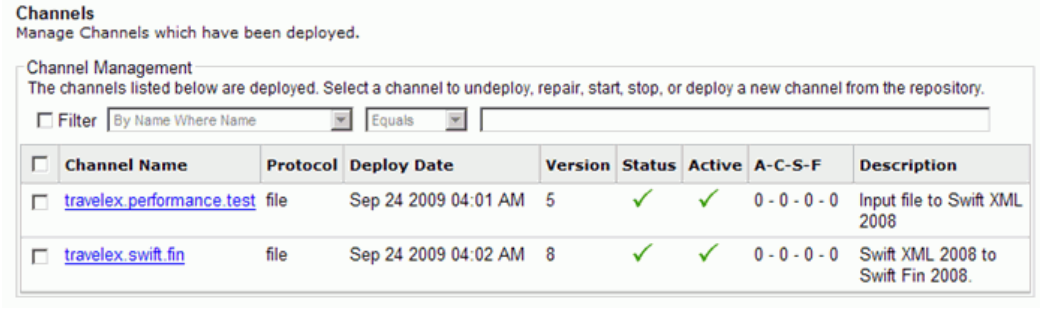


- A heartbeat is generated approximately every tenth of a second by the monitored iSM and sent to the monitoring server.

- Heartbeat rate could be affected if the system clock were to slow down. But the work load of the server should not have any real impact on the heartbeat rate.

Deploying the Primary iSM Server Instance

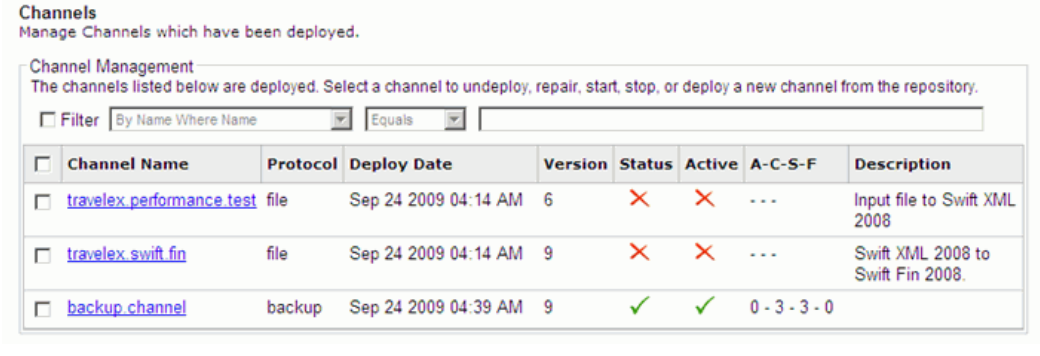
Channels are deployed as usual, as shown in the image below.



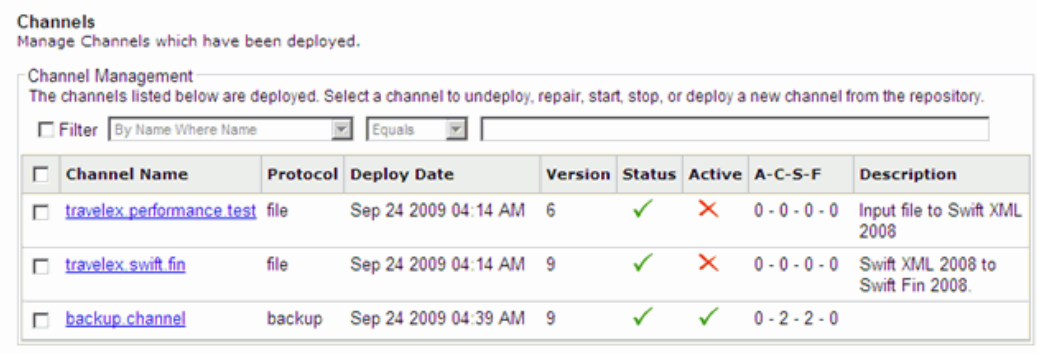
Backup iSM Server Instance

The same Channels in the primary server are deployed in the backup but in the Stopped and Inactive states. They are only to be activated when there is a failover detected from the primary server. Failover is detected by the backup.channel by listening for heartbeats from the primary server.

The image below is a normal operation that is receiving heartbeats.

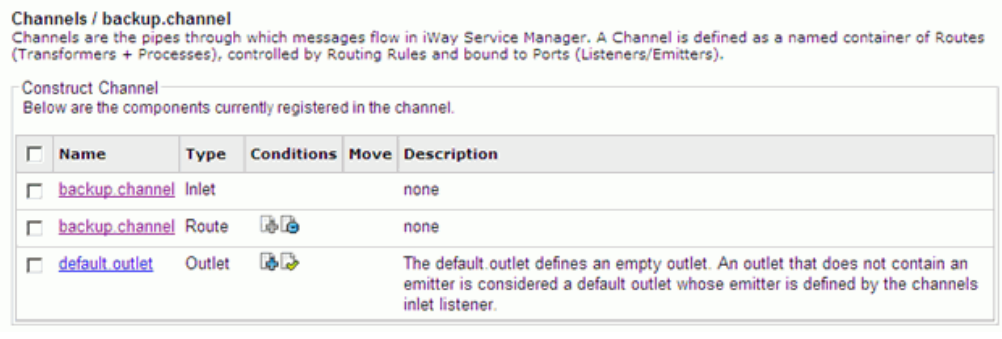


The following image illustrates a backup operation that did not receive heartbeats.



Configuring the Backup Channel

The backup.channel consists of a backup listener and process flow that handles messages resulted from the heartbeats.



Backup Listener

The Backup Listener is created from the iSM Console by selecting the Backup type from the Listener type drop-down list. The listener listens for the heartbeats emitted from the primary server. Depending on the condition, it may create an XML message that contains:

```
<signal host='hostname' config='configname'>state</signal>
```

where:

hostname

Is the name of the system where iSM is running.

configname

Is the name of the iSM configuration being used (for example, base).

state

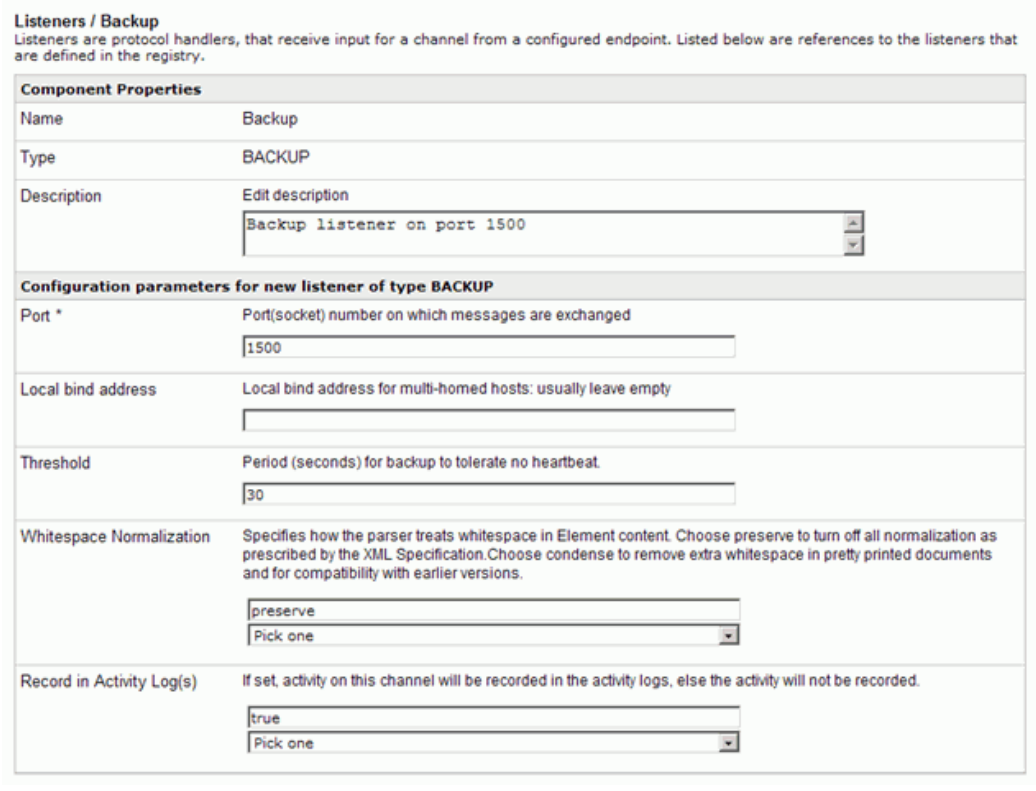
Can be LIVE, DEAD, or CLOSE.

The following table describes each state in more detail.

State	Description
LIVE	The monitored iSM has gone live. This message appears when the monitored iSM resumes or begins operation. If the monitoring iSM starts first, this message will be dispatched when the monitored iSM starts.
DEAD	The monitored iSM has not sent a heartbeat for the threshold duration, and the monitored iSM was previously in LIVE state.
CLOSE	The monitored iSM has signaled that it is terminating normally. If it restarts, you will receive a LIVE message. Once a monitored iSM signals CLOSE, missed heartbeats are ignored. Heartbeats are monitored only when the monitored iSM is in LIVE state.

- Enter a port number that this server will use to listen for heartbeats emitted from the primary server.

- Threshold is the length of time that the backup server waits for a heartbeat before starting itself up.



Listeners / Backup

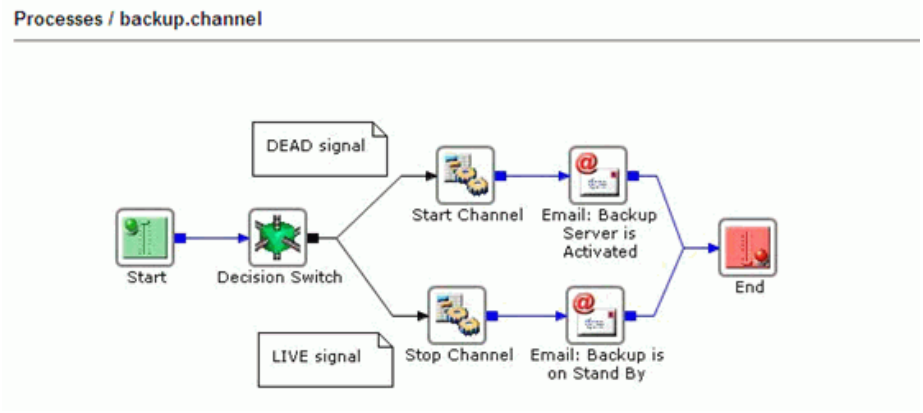
Listeners are protocol handlers, that receive input for a channel from a configured endpoint. Listed below are references to the listeners that are defined in the registry.

Component Properties	
Name	Backup
Type	BACKUP
Description	Edit description Backup listener on port 1500
Configuration parameters for new listener of type BACKUP	
Port *	Port(socket) number on which messages are exchanged 1500
Local bind address	Local bind address for multi-homed hosts: usually leave empty
Threshold	Period (seconds) for backup to tolerate no heartbeat. 30
Whitespace Normalization	Specifies how the parser treats whitespace in Element content. Choose preserve to turn off all normalization as prescribed by the XML Specification.Choose condense to remove extra whitespace in pretty printed documents and for compatibility with earlier versions. preserve Pick one
Record in Activity Log(s)	If set, activity on this channel will be recorded in the activity logs, else the activity will not be recorded. true Pick one

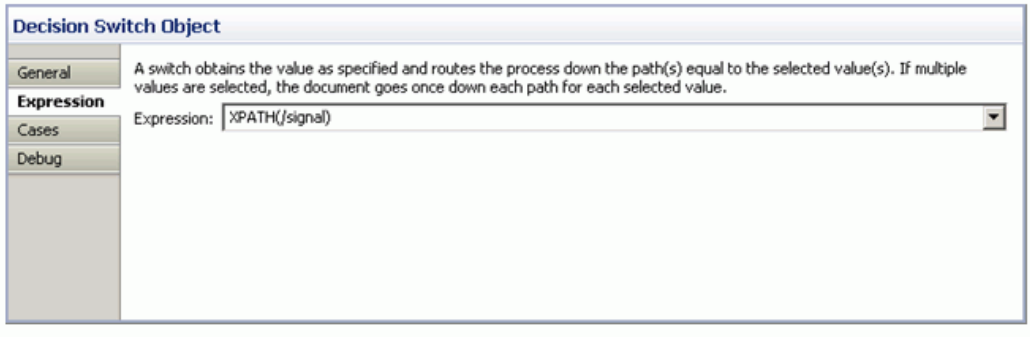
Note: The threshold value will depend on the deployed environment.

Process Flow

The process flow interrogates the inbound XML that is generated by the backup listener. Using XPATH, the state of the heartbeat or the primary server is retrieved and using a Decision Switch to determine the next action to take.

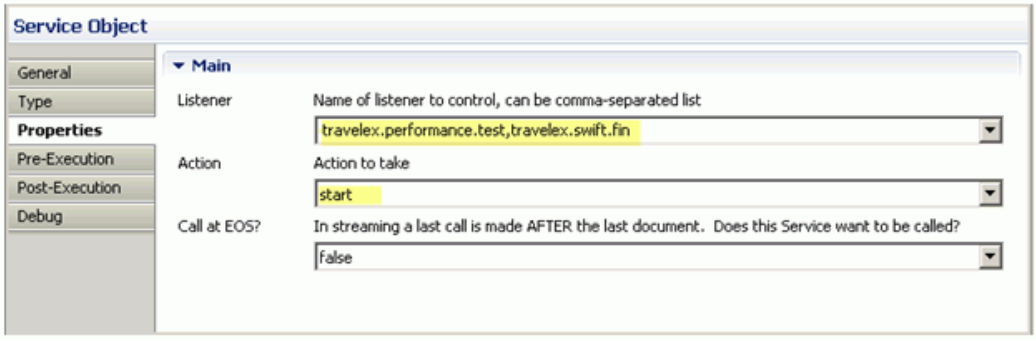


The Decision Switch Object is shown below.

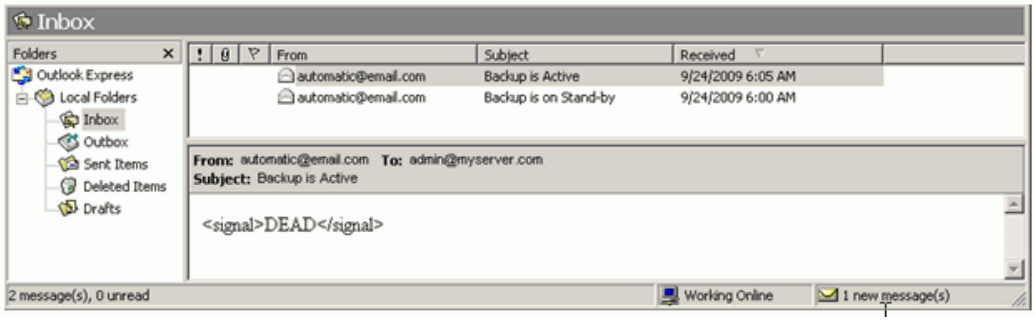


Case	Description
null	Expression returns with no nodes
empty	Expression returns with node(s) with no value
LIVE	Receiving heart beats from the monitored server instance
DEAD	No heart beat from the monitored server instance

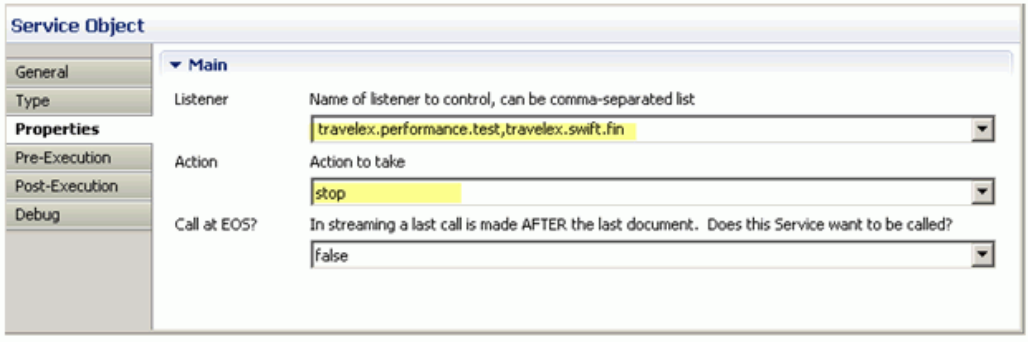
The Start Channel Service contains a list of channels to be started when a failover from the primary server occurs, as shown in the image below.



The Email Object notifies someone when the primary server failure is detected and backup has become active, as shown in the image below.



The Stop Channel Service contains a list of channels to be stopped when the primary server is LIVE, as shown in the image below.



The Email Object notifies someone when the primary server is back up and the backup server is on stand-by, as shown in the image below.

iWay LDAP High Water Mark Listener

The iWay LDAP High Water Mark (HWM) listener follows the HWM design pattern by acquiring new or changed records (LDAP entries) from a data store (the LDAP server) that meet a specific, evolving criteria. The data that is returned is determined by the scope and the specified filter. To avoid retrieving duplicate entries, the iWay LDAP HWM listener keeps track of the highest value of the filter attribute which has been read. Usually the filter value is the creation or modification time of the entry, but any unique and steadily increasing value can be used. Each entry satisfying the filter is returned as an iWay XML document for further processing in an iWay Service Manager channel.

The iWay LDAP HWM listener is triggered (begins processing messages) when an entry is inserted or modified in the LDAP directory that is based on an increasing value of an attribute. The iWay LDAP HWM listener works similar to the iWay Relational Database High Water Mark listener.

Configuring the iWay LDAP High Water Mark Listener

The iWay LDAP HWM Listener acquires LDAP entries, formats them into XML format, and inserts them into an iWay channel for further processing. The configuration process for the LDAP HWM listener defines the properties that the listener requires to accomplish these tasks. For example, the address of the LDAP server, account authentication credentials, filter criteria, and which attributes to return.

For best results in building an application using the LDAP HWM listener, having knowledge about the schema that is employed by your LDAP implementation is recommended.

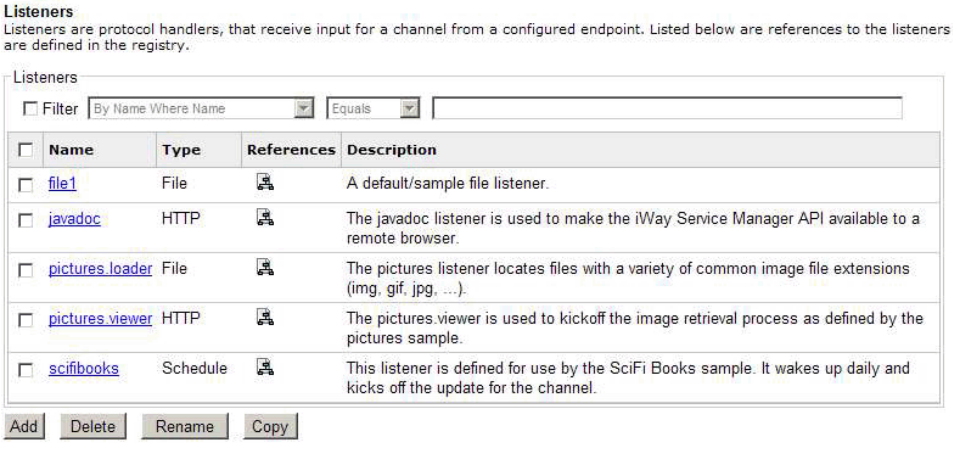
Note: This section describes how to configure an iWay LDAP HWM listener. To construct a fully populated iWay Service Manager channel, incorporate the listener into an inlet and then use the inlet in a channel. For more information on how to design and build a channel, see the *iWay Service Manager User's Guide*.

Procedure: How to Configure the iWay LDAP HWM Listener

To configure the iWay LDAP HWM listener:

1. In the left console pane of the Registry menu, select *Listeners*.

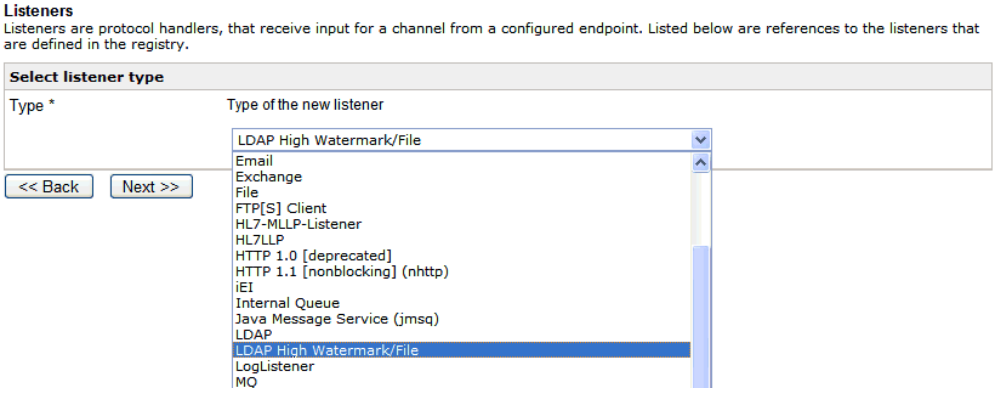
   The Listeners pane opens, as shown in the following image.



   The table that is provided lists any existing listeners and a short description for each.

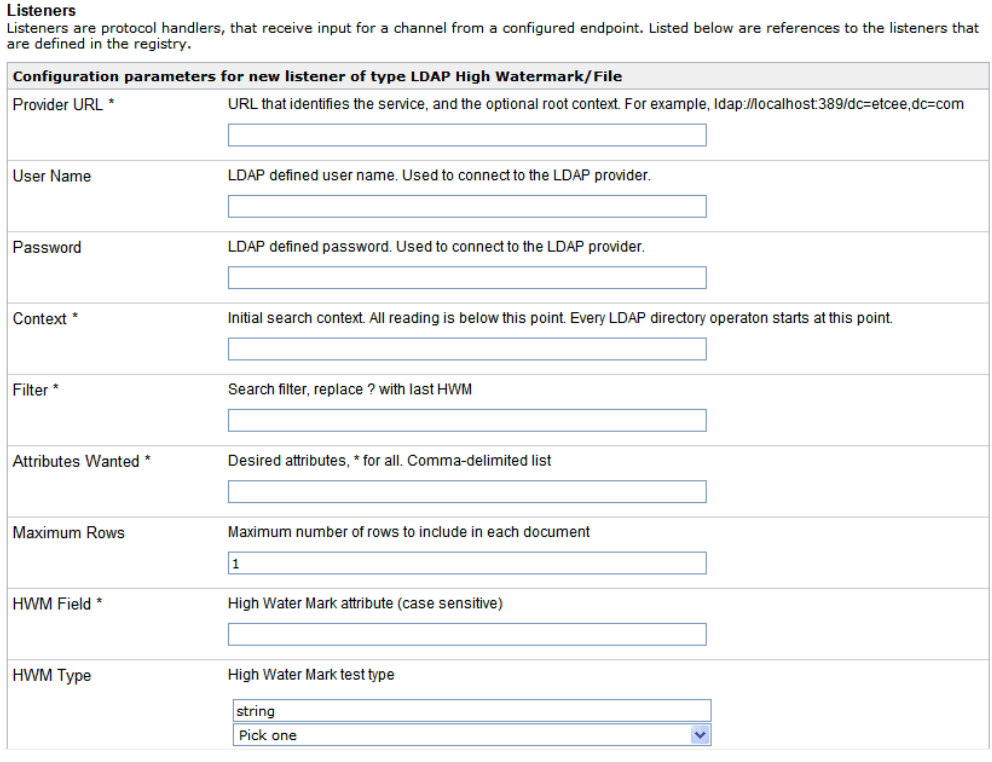
2. Click *Add*.

   The Select listener type pane opens.



3. Select the *LDAP High Watermark/File* listener from the drop-down list, then click *Next*.

   The Configuration parameters for new listener of type LDAP High Watermark/File pane opens.



4. Provide the appropriate property values for the LDAP HWM listener, as defined in the following table.

   Note: An asterisk indicates a required property.

Property	Definition
Provider URL *	URL that identifies the service, and the optional root context. For example: `ldap://localhost:389/dc=etcee,dc=com`
User Name	LDAP defined user name.
Password	LDAP defined password.

Property	Definition
Context *	Context under which the search is to take place. For example: `ou=bookkeeping`
Filter *	The criteria for selecting entries from the LDAP server. The question mark character (?) is substituted by the current HWM value. The filter must be set to a non-operational attribute. Attributes that have special meaning to the Directory server (for example, createTimestamp, modifiersName, modifyTimestamp) are known as operational attributes.
Attributes Wanted *	A list of attributes to be returned. The attributes are presumed to be strings. Any attribute can be single or multi-valued. If an asterisk character (*) is entered, then all attributes are read.
Maximum Rows	Maximum number of rows that will be read into the current XML document. Any unread rows meeting the HWM criteria will be read in the next iteration of the listener.
HWM Field *	Name of the attribute which is the source of the next HWM value. This must be a single-valued attribute. For example, if the last modified timestamp was the high water mark, then the following filter would not work: `modifyTimeStamp>=?` As a result, the value in the HWM field must be a non-operational attribute.

Property	Definition
HWM Type	The type of HWM field. This property is used to test HWM attributes. Select one of the following values from the drop-down list: ❑ Date ❑ Number ❑ String **Note:** Dates are recorded in most LDAP directories as strings using the following format: `yyyymmddhhmmss` As a result, a string compare is sufficient. Date attributes should be described as date types to ensure compatibility.
HWM Path *	Name of the file in which the HWM value is stored.
HWM Default *	Value to be used as the HWM on the first run of the listener, or if the HWM value file cannot be found.
Base 64	Each value is examined and if the value cannot be represented in the normal character set, it is replaced with its base64 representation. This is not necessary for binary data if processing the result will not be processed as an iWay XML document.

Property	Definition

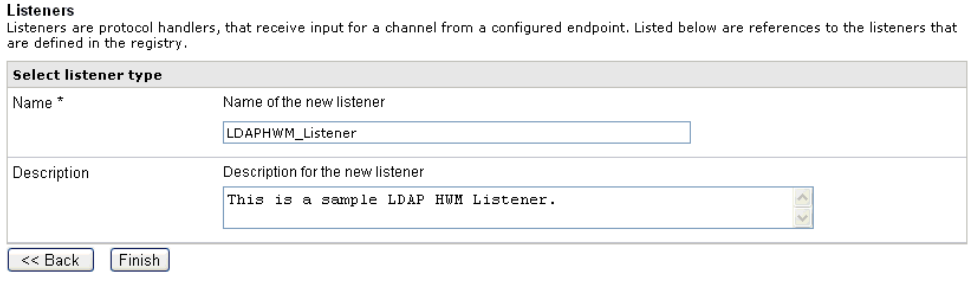
Parse Attributes	Determines whether values can be parsed, which means that the attributes are in the form of token/value pairs separated by commas. If so, the value node contains an element for each token. For example:
	`<attribute values="1" name="distinguishedName">`
	`<value>`
	`<CN>Herz, Judy</CN>`
	`<OU>USERS</OU>`
	`</value>`
	`</attribute>`
	If the contents of the value do not meet the criteria for parsing, the value is not parsed. For a more detailed sample, see *Sample LDAP HWM iWay Documents* on page 138.

Whitespace Normalization	Specifies how the parser treats whitespaces in element content. Choose preserve (default) to disable all normalization as prescribed by the XML specification. Choose condense to remove extra whitespaces in pretty printed documents and for compatibility with earlier versions.
Accepts non-XML (flat) only	If set to true, the listener expects only flat (non-XML) files and the preparsers do not run.
Optimize Favoring	Select one of the following values from the drop-down list: ❑ performance ❑ memory **Note:** Selecting memory is recommended if you are expecting large input documents.
Multithreading	Number of documents that can be processed in parallel.
Maximum threads	Number of parallel threads that can grow automatically on demand.
Execution Time Limit	Time limit (in seconds) for a document to execute before it is terminated.

Property	Definition
Polling Interval	Interval at which to check for new input.
Default Java File Encoding	Default encoding if the incoming message is not self-declaring, for example, XML.
Always reply to listener default	If set to true, the default reply definition is used in addition to the defined replies.
Error Documents treated normally	If set to true, error documents are processed by any configured preemitters.
Listener is Transaction Manager	If set to true, agents run within a local transaction managed by the listener.
Record in Activity Log(s)	If set, activity on this channel will be recorded in the activity logs, otherwise the activity will not be recorded.
AES Key	If the channel will receive encrypted AFTI messages, set the AES key (maximum 16 characters) to be used for decrypting.

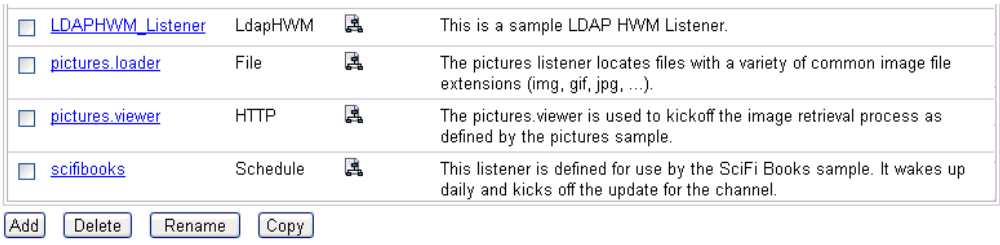
5. Click *Next*.

   The Select listener type pane opens.



6. Provide a name and, optionally, a description, for the LDAP HWM listener, and click *Finish*.

The newly created LDAP HWM listener is added to the list in the Listeners pane.



Sample LDAP HWM iWay Documents

The XML document that is produced by one iteration of the LDAP HWM listener is formed as follows:

```
<listenerName>
   <row>
      <attribute values="count of values" name="attribute name>
         <value>data<value>*
```

With no parsing of the attributes, the resulting document appears as:

```
<?xml version="1.0" encoding="UTF-8" ?>
<ibidca>
  <row>
    <attribute values="1" name="l">
      <value>New York</value>
    </attribute>
    <attribute values="1" name="uSNCreated">
      <value>159019</value>
    </attribute>
    <attribute values="1" name="distinguishedName">
      <value>CN=Herz\,Judy,OU=USERS,OU=EI,OU=IWAY,
             OU=COR,DC=ibi,DC=com</value>
    </attribute>
    <attribute values="1" name="postalCode">
      <value>10121</value>
    </attribute>
    <attribute values="1" name="co">
      <value>United States</value>
    </attribute>
    <attribute values="1" name="cn">
      <value>Herz, Judy</value>
    </attribute>
    <attribute values="1" name="st">
      <value>NY</value>
    </attribute>
    <attribute values="1" name="streetAddress">
      <value>2 Penn Plaza</value>
    </attribute>
  </row>
</ibidca>
```

If the base64 attribute is selected, each attribute value is checked as to whether it can be represented in the ISO-8859-1 character set. Those that cannot, are converted to base64:

```
<attribute values="1" name="objectGUID">
  <value>base64(77+9Sinvv71nUk9P77+96LhG77+978+9JO+/vm==value>
</attribute>
```

The parsing parameter causes the listener to attempt to break down the value of each attribute into token/value pairs:

```
<?xml version="1.0" encoding="UTF-8" ?>
<ibidca>
  <row>
    <attribute values="1" name="l">
      <value>New York</value>
    </attribute>
    <attribute values="1" name="uSNCreated">
      <value>159019</value>
    </attribute>
    <attribute values="1" name="distinguishedName">
      <value>
        <CN>Herz, Judy</CN>
        <OU>USERS</OU>
        <OU>EI</OU>
        <OU>IWAY</OU>
        <OU>COR</OU>
        <DC>ibi</DC>
        <DC>com</DC>
      </value>
    </attribute>
    <attribute values="1" name="st">
      <value>NY</value>
    </attribute>
    <attribute values="1" name="streetAddress">
      <value>2 Penn Plaza</value>
    </attribute>
  </row>
</ibidca>
```

Later, an option will allow selection of DSML or another standard.

Using the Diagnostic Tool

The following command invokes the command line diagnostic tool from the iWay Service Manager command console:

```
tool testldap <providerurl> <userid> <password>
```

Three prompts are subsequently presented for the context, filter, and attribute list. Queries made to the LDAP server from within the diagnostic tool are standalone, for example, the LDAP HWM listener filter character and HWM value storage are not active. The diagnostic tool displays the query result as an iWay XML document, exactly as it would be constructed by the LDAP HWM listener during run time.

To exit the diagnostic tool, enter a simple null line or the word *end* at the context request prompt.

iWay Relational Database High Water Mark Listener

The iWay Relational Database High Water Mark listener (RDBHWM) operates in real-time to acquire newly inserted or updated rows from a relational database. In order to identify the desired rows, a column (or concatenation of columns) must exist, which contains an increasing high water mark value.

The listener saves the highest HWM value read with each SELECT, and constructs the subsequent query using the saved value. Unlike the iWay RDBMS listener, no deletes or updates are necessary to prevent the rereading of processed rows. Because the iWay RDBHWM listener can access the database as read-only, the performance of the iWay RDBHWM listener is faster than the iWay RDBMS listener.

Configuring the iWay Relational Database High Water Mark Listener

This section describes how to configure an RDB High Water Mark (RDBHWM) listener. Be sure you have identified a good candidate for the high water mark value column. A column chosen for the high water mark should be:

- Always increasing, never decreasing.

  If the column value can become lower with time, any new rows with decreasing values can never be read because of the uniform upward movement of the high water mark.

- Unique (for example, quickly changing, relative to the listener query interval)

  If the column value stays constant for a number of records (for example, a date versus a timestamp), then some rows may be omitted because of the strictly greater-than predicate of the SELECT.

- Restricted to the result set retrieved by a single SELECT. Although it is permitted to store multiple rows with the same key (for example, using a change timestamp that may have several rows in that time) the key value must not span multiple selects. For example, using a clause, such as FIRST 5, can cause the duplicate High Water Mark (HWM) value to appear in more than one SELECT. However, this can conflict with the strictly greater-than predicate of the SELECT, causing rows to be lost.

  Note: This section describes how to configure an RDB High Water Mark (RDBHWM) listener. To construct a fully populated iWay Service Manager channel, incorporate the listener into an Inlet and then use the Inlet in a Channel. For more information on how to design and build a channel, see the *iWay Service Manager User's Guide*.

Procedure: How to Configure the RDB High Water Mark Listener

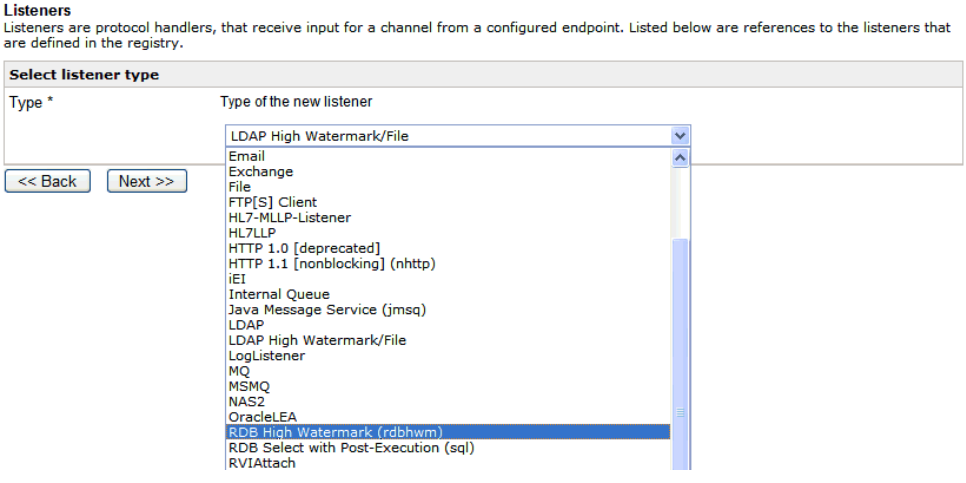
To configure the RDB High Water Mark (RDBHWM) listener:

1. From the iWay Service Manager Administration Console, click *Registry* and then *Listeners*.

   The Listeners pane opens.

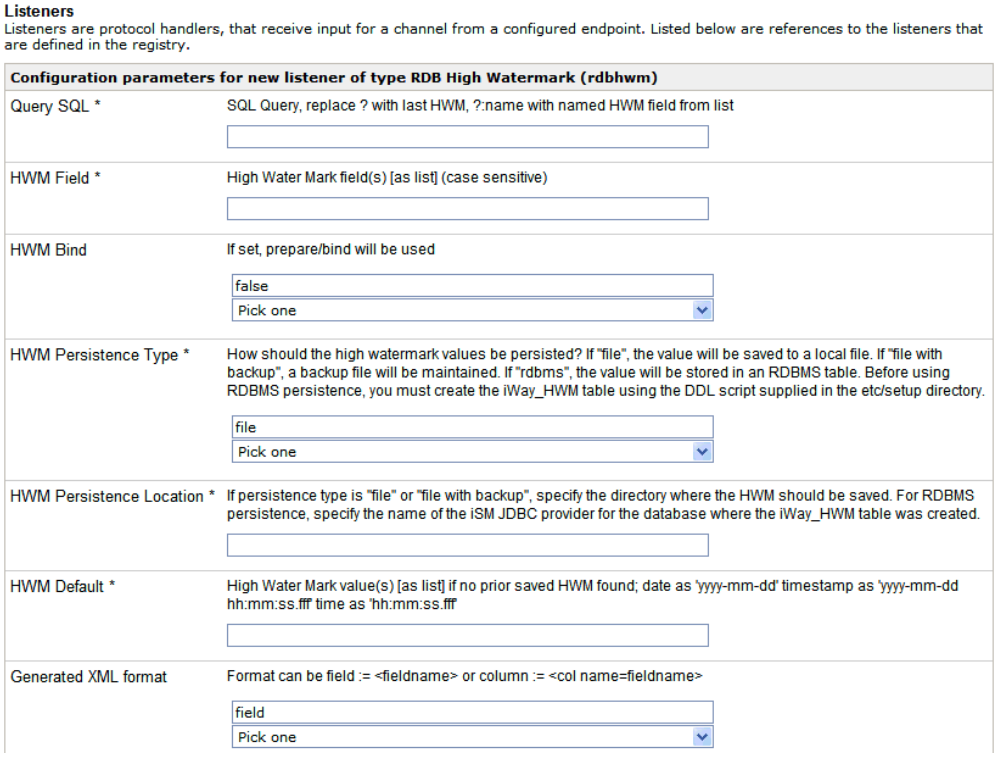
2. Click *Add*.

   The Select listener type pane opens.



3. Select the *RDB High Watermark (rdbhwm)* listener from the drop-down list, then click *Next*.

The Configuration parameters for new listener of type RDB High Watermark (rdbhwm) pane opens.



Note: This image is used for demonstration purposes and only shows the first set of configuration parameters for the RDBHWM listener in the iWay Service Manager Administration Console. The complete list of configuration parameters are listed and described in the table in step 4.

4. Provide the appropriate property values for the RDBHWM listener, as defined in the following table.

   Note: An asterisk indicates a required property.

Parameter	Definition
Query SQL*	Query statement to be issued. It must contain at least one question mark (?), which will be replaced with the high water mark value from the prior SQL select. The following example assumes that your HWM column is called hwm. `select col1, col2, hwm from tab1 where hwm > '?' order by hwm` This may be a date-time field or an increasing integer. Numeric fields should not be enclosed in single quotes. Character fields and date fields must have the single quotes coded in the SELECT as illustrated. If multiple HWM columns are used, the ? notation is replaced with ?:name, where name is an HWM column name. **Note:** If you are using an EDA server, column names are case-sensitive.
HWM Field*	Name of the column(s) containing the high water mark in the retrieved row(s). It is case-sensitive. In the above example, the value would be hwm. If the high water mark consists of multiple columns, they must be comma delimited in order from the most significant to least significant value (for example, most frequently changing field entered last). For better performance and to minimize complexity, it is preferable to identify single column which can be used as the HWM value. The HWM column(s) must also be present in the SELECT list of the query SQL.

Parameter	Definition
HWM Bind	If set to *true*, a bind strategy is used to construct the SELECT statement. If set to *false*, the SQL is constructed for each iteration of the listener. **Note:** HWM Bind allows the system to prepare, and then execute the SELECT statement with different values. Some JDBC drivers optimize for prepared statements while others do not. This option is made available so you can choose the strategy that works best for your application. The prepared statement must follow the SQL rules for such statements in the database being used. The question mark (?) must not be enclosed in quotes.
HWM Persistence Type *	Determines whether the HWM values should be persisted. Select one of the following options from the drop-down list: ❏ **File.** The value will be saved to a local file. ❏ **File With Backup.** A backup file will be maintained. ❏ **RDBMS.** The value will be stored in an RDBMS table. Before using RDBMS persistence, you must create the iWay_HWM table using the DDL script that is provided in the etc/setup directory. For more information, see *Configuring the HWM Persistence Type Parameter* on page 152.
HWM Persistence Location *	If the HWM Persistence Type parameter is set to *File* or *File With Backup*, then specify the directory where the HWM should be saved. For RDBMS persistence, specify the name of the iSM JDBC provider for the database where the iWay_HWM table was created.
HWM Default*	Value to be used as the high water mark on the first run of the listener, or if the high water mark value file is not found.

Parameter	Definition
Generated XML Format	Select one of the following values from the drop-down list: ❑ column ❑ field ❑ row
Column Limit	Varchars exceeding the specified limit will be carried as external files.
Maximum Rows	The maximum number of rows to include in each document. The default value is 1.
Ignore Case	Some databases are case-sensitive and cannot recognize a field if the case is different than the one in which the field is stored in the database. As a result, setting this field to *true* facilitates column recognition.
Base64 Check	Each value is examined and if the value cannot be represented in the normal character set, it is replaced with its base64 representation. This is not necessary for binary data if processing the result will not be processed as an iWay XML document.
Include xLOBs in Tree	If set to *true*, the contents of a Binary Large Object (BLOB) or Character Large Object (CLOB) are included in the tree and not as external files. BLOBs will be encoded using Base64. The inclusion of xLOBs in the tree can result in significant memory use.

Parameter	Definition
Transaction Isolation Level	Transaction isolation level to be set if possible. Select one of the following options from the drop-down list: ❑ **Asis.** Leaves the isolation level as defined by the database. ❑ **Compute.** Selects the highest isolation level supported by the database. ❑ **Read Committed.** Will never read data that another application has changed and not yet committed. This option does not ensure that data will not be changed before the end of the transaction. ❑ **Read Uncommitted.** Allows reading of a record that may be rolled back later. ❑ **Repeatable Read.** Dirty reads and non-repeatable reads cannot occur. All data used in the query is locked and another transaction cannot update the data. Each option functions as defined, but selecting the appropriate option can be confusing. For most applications, selecting the Asis option is sufficient, which effectively sets the Transaction Isolation Level to none. These options are provided so that application developers can control transactionality for their solution. Isolation levels differ among database engines and developers are encouraged to test how these options impact processing.

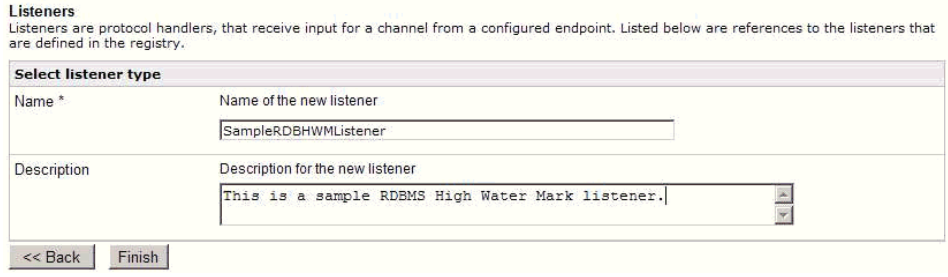
Parameter	Definition
When To Save	Determines at what point during the life cycle of a message that the High Water Mark (HWM) value should be saved. Select one of the following values from the drop-down list: ❑ **dispatch.** Saves the HWM value as each row of the database is dispatched to the channel. This is the default. ❑ **result set.** Saves the HWM value when all of the records retrieved by an SQL execution are completely processed. ❑ **row.** Saves the HWM value as each row is completed. One thread is allowed. For more information, see *Configuring the When To Save Parameter* on page 151.
Connection	
Use JNDI *	If set, the SQL connection is taken from a data source provider. If not set, the connection is established from locally entered driver and URL fields
JNDI Factory	Determines how to reach the data source provider. Use this parameter only if the Use JNDI parameter is set. The default is appropriate when using iSM data providers. If you are running in an application server, you may want to change this value to reach data sources on that server.
JNDI Name	The JNDI name of the requested data source. To use an iWay JDBC provider, specify as jdbc/provider. This parameter is required if the Use JNDI parameter is set to *true*.
Driver	The full name of the JDBC driver to reach the database.
URL	The URL used by the JDBC driver to access the database.
User Name	The database user name to access input table.
Password	The database password for the user.
Tuning	

Parameter	Definition
Multithreading	Number of documents that can be processed in parallel for this listener. The default value is set to 1.
Maximum Threads	Parallel threads can grow to this count automatically on demand. The default value is set to 1.
Optimize Favoring	Select one of the following values from the drop-down list: ❏ performance ❏ memory **Note:** Selecting *memory* is recommended if you are expecting large input documents.
Polling Interval	The specified Interval at which to check for new input or check for server stop (in seconds). The default value is set to 2 seconds.
Events	
Failed ReplyTo Flow	Name of published process flow to run if a message cannot be emitted on an address in its reply address list.
Dead Letter Flow	Name of published process flow to run if an error cannot be emitted on an address in its error address list.
Channel Failure Flow	Name of published process flow to run if this channel cannot start or fails during message use. The server will attempt to call this process flow during channel close down due to the error.
Parse Failure Flow	Name of published process flow to run if XML parsing fails for incoming message.
Channel Startup Flow	Name of published process flow to run prior to starting the channel.
Channel Shutdown Flow	Name of published process flow to run when the channel is shut down.
Other	

Parameter	Definition
Whitespace Normalization	Specifies how the parser treats whitespaces in element content. Choose *preserve* (default) to disable all normalization as prescribed by the XML specification. Choose *condense* to remove extra whitespaces in pretty printed documents and for compatibility with earlier versions.
Accepts non-XML (flat) only	If set to *true*, the listener expects only flat (non-XML) files and the preparsers do not run.
Execution Time Limit	Time limit (in seconds) for a document to execute before it is terminated.
Default Java File Encoding	Default encoding if the incoming message is not self-declaring, for example, XML.
Always Reply to listener default	If set to *true*, the default reply definition is used in addition to the defined replies.
Error Documents treated normally	If set to *true*, error documents are processed by any configured preemitters.
Listener is Transaction Manager	If set to *true*, agents run within a local transaction managed by the listener.
Record in Activity Log(s)	If set to *true*, the activity on this channel will be recorded in the activity logs, else the activity will not be recorded.
Startup Dependencies	Comma-separated list of channel names that must be started before this one.

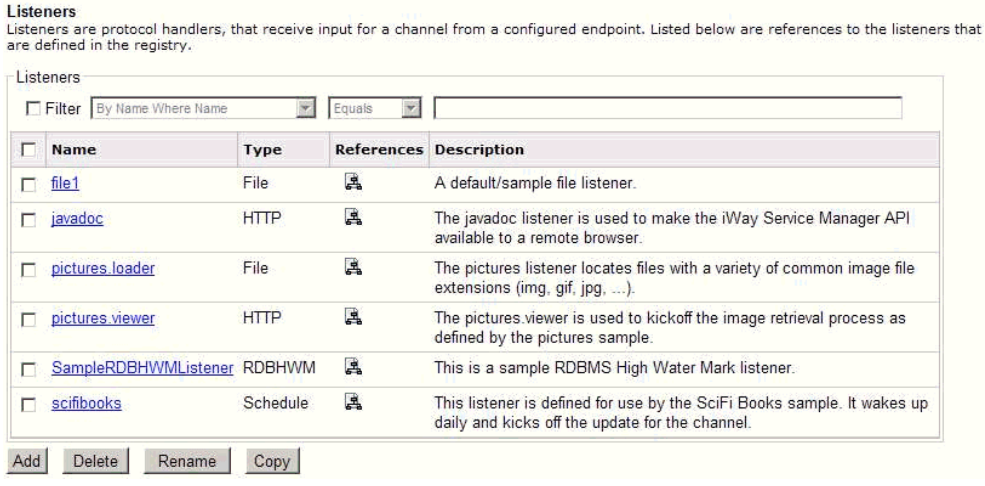
5. Click *Next*.

The Select listener type pane opens.



6. Provide a name and optionally, a description for the RDBHWM listener, and click *Finish*.

   The newly created RDBHWM listener is added to the list in the Listeners pane, as shown in the following image.



Configuring the When To Save Parameter

The When To Save parameter determines at what point during the life cycle of a message the High Water Mark (HWM) key(s) are saved to the external file. The external file is used to determine from which point to pick up the read when the listener starts next or resumes. The values that can be selected for this parameter depends on the requirements and design of the application:

- **dispatch.** The HWM value is saved before the message is dispatched for execution. This is the original saving point. This is the default.

- **resultset.** If your SQL DML selects more than one row, and you expect more than one record to be returned with the same HWM value (for example, multiple records for a given timestamp or order number), then this value may be appropriate. The HWM value would be stored when all of the records in the result set are completely processed. Failure to use this setting may result in the loss of some records if the server fails while records are being processed. In this scenario, the HWM key might have been written and the DML specifies a value greater than the HWM value during startup.
- **row.** The HWM value is saved as the key changes to the next key (or the end of the current result set). This allows multiple keys of the same value to be processed within the result set. This is applicable only for cases that involve a single processing thread. If multiple threads were used, then the order of completion for the rows could not be guaranteed. This would result in possible gaps in the HWM tracking value, and as a result, the potential loss of records during a restart. For example, assume the completion order of a three parallel thread channel completed in the order of 4, 6, and 5. The HWM update would be 4 and then 6. Record 5 would not be written, since it is not an increase of the previous record (6). If the server fails, then record 5 would not be reread.

Configuring the HWM Persistence Type Parameter

The High Water Mark (HWM) listeners read through a relational table in series, keeping the access keys of the last row processed. The assumption is that the access keys are ordered by the increasing value of key field(s). These state fields are written to external media as the states change. The state is tracked internally, and the external media is read when the listener starts. It is not read at any other time. The persistence options can be selected from the HWM Persistence Type drop-down list.

Three variations on this design pattern are provided, as listed in the following table.

Variation	Use	Managed By
RDBHWM/File	HWM kept in a local file. Fastest.	Listener
RDBHWM/Source DBMS	HWM kept in a source database.	Listener
RDBHWM/Target DBMS	HWM kept in a target database.	Application

File System Variation

As the listener progresses through the source database, the High Water Mark (HWM) is maintained in the local file system. On restart, the listener accesses the file system to determine the current state of the HWM, and resumes reading from that point. Optionally, the configuration can call for a duplicate file to be written as a backup if the quality of the external file system media is suspect.

Source DBMS Variation

As the listener progresses through the source data base, the High Water Mark (HWM) values are updated by the listener in the source database. The listener uses the source provider for access to the database. This requires that the provider offer write permissions to the database.

The advantage of a source database management system (DBMS) is the storage in a (possibly shared) database. This can simplify restart in the event that the system where iSM is hosted is down. Standard clustering facilities of the database can protect against loss of data for any piece of hardware or network segment.

Target DBMS Variation

The listener progresses through the source database, but the application is responsible for maintaining the High Water Mark (HWM) state in a target database, which must be identified to the listener by its provider. The application must update the target database with the values to be read by the listener on a restart.

Although there is no restriction on when the HWM is stored (for example, it can be in a channel in a multi-channel architecture if called for by the application design), the update statement must be compatible with the target access statement configured for the listener. The SQL object (XDSQLAgent) can be used to store the values in the target table.

A possible use case scenario for this design pattern is in a load, modify, and update application. By configuring the SQL object to use the same connection as the updating SQL object, a single commit can harden the application and the HWM table, or both can be rolled back. This keeps the target and the HWM values in synch. Operating the updating channel as a local transaction manager is recommended for this approach.

Although you must configure the target database provider for the listener, it is only used to access the saved HWM when the listener starts. For the purposes of constructing the update SQL, the HWM key value should be stored as a varchar. The value is in the <mastername>.HWM.key special register. The contributing field(s) are in <mastername>.HWM.<fieldname> special registers.

iWay Application Adapter for Salesforce

The iWay Application Adapter for Salesforce provides a means to exchange real-time business data between Salesforce systems and third-party application, database, or external business partner systems. The adapter enables external applications for inbound and outbound processing with Salesforce. In addition, the iWay Application Adapter for Salesforce provides interfaces and integration touchpoints for the Salesforce SaaS service. The adapter uses XML messages to enable non-Salesforce applications to communicate and exchange transactions with Salesforce.

The iWay Salesforce listener is used by applications to access Salesforce data when a Salesforce business event occurs.

For more information on configuring and using the Salesforce listener, see the *iWay Application Adapter for Salesforce User's Guide*.

iWay Enterprise Index

The iWay Enterprise Index (iEI) provides a secure, user-friendly way to search iWay Service Manager (iSM) messages. You can search for specific content, regardless of the order or context which the content criteria appears in the message. iEI enables you to make Key Word Out of Context (KWOC) inquiries to all messages that pass though iSM, and provides security to prevent unauthorized parties from accessing information.

The iEI listener fields query requests and produces the appropriate result. The process flow associated with the listener must interpret the request using the stored URL parameters (that appear in special registers at the start of the process flow) to determine the appropriate course of action. Audit Message Indexing may also use the iEI Renderer service to interact with the drivers to recover the audited data. Direct Indexing must recover the required information itself. In either case, the output document emitted by the process flow is generated.

For more information on how to configure the iEi listener, see the *iWay Enterprise Index User's Guide*.

iWay Log Event Adapter for Oracle

The iWay Log Event Adapter for Oracle is designed to replicate data changes from Oracle 10g or 11g to Oracle 10g or 11g and SQL 2000, 2005, or 2008. The iWay Log Event Adapter for Oracle uses the OracleLEA listener to capture the changed data from the Oracle redo logs using Oracle LogMiner. The OracleLEA listener starts an Oracle LogMiner session, which uses the redo logs to create a view. The listener then reads the transactions from the view and converts them to an XML document. That XML document is then passed into a process flow. In the event that a log switch occurs before the listener polls, the listener reads from the archive logs to ensure that no transactions are lost.

For more information on how to configure the OracleLEA listener, see the *iWay Log Event Adapter for Oracle* documentation.

Chapter 5

Transport Utility Protocol Adapters

The protocols for the iWay transport utility protocol adapters support email exchanges and file transfers between Internet users. The protocols also support exchanges and file transfers between those connected through TCP/IP and other networks.

iWay transport utility protocol adapters provide tools that simplify the implementation of service-oriented architectures. The simplification is accomplished by reducing the requirement for custom programming when implementing a range of distributed messaging services and file transport services and by providing access to packaged third-party adapter products. By minimizing the amount of code required for their implementation, these adapters provide a solid foundation for flexible service-oriented architecture.

In this chapter:

Email

Email messages typically consist of a set of header fields, an optional body, and an optional set of attachments. The headers contain information about the message, such as to whom it is sent, from whom it was sent, when it was sent, the subject, and so on.

There are several existing email standards. These protocols define how email messages are transferred, the format of the email, and other definitions that enable email to be handled uniformly by various mail servers (which receive and store the mail) and clients (which communicate to the servers to retrieve and read mail messages). The most popular protocols are Simple Mail Transfer Protocol (SMTP), Post Office Protocol (POP3), and Internet Message Access Protocol (IMAP4).

SMTP is a TCP/IP protocol used to send and receive email. However, because it has limited capability of queuing messages at the receiving end, it is usually used with one of the other two protocols, POP3 or IMAP4. These protocols enable users to save messages in a server mailbox and download them periodically from the server. Users usually use software that employs SMTP for sending email and POP3 or IMAP4 for receiving messages retrieved for them at their local server.

POP3 uses a store and forward service. Using POP3, mail is delivered to a centralized server, and users periodically connect to the server to copy or move pending emails to a mobile device. IMAP4 is a client-server mail protocol that enables handling of remote mailboxes as if they were local. With IMAP4, mail is stored on a centralized server and users can copy or move mail to their mobile device. Depending on how IMAP4 is used by a mail server, users can request the server for the headers or bodies of specified messages. Messages remain in the mail repository until marked for permanent deletion. The iWay Adapter for E-mail supports both POP3 and IMAP4 protocols.

The iWay Adapter for E-mail can use the POP3 or IMAP protocols to monitor an email account, select inbound messages, and filter or customize the processing of email based on content using contextual pattern matching of the subject. You can configure the email listener to poll an account at a configured level. The email listener provides support for both preprocessing and postprocessing of both message bodies and attachments.

You can deploy the iWay Adapter for E-mail on all J2EE supported platforms including Windows, Solaris, AIX, and OS/390.

The iWay Adapter for E-mail enables transfer of data to and from an email server. The adapter supports inbound and outbound integration operations. The adapter connects to the email server, extracts data, and converts it to the data types supported by the operation using the conversion instructions in its adapter business object type. Standard iWay processing is then performed. The adapter interacts with the server engine to provide responses from business logic back to the email server. The iWay Adapter for E-mail enables seamless integration of your email servers into your enterprise business processes and enables you to realize a rapid return on investment.

The adapter supports asynchronous bidirectional operations between the adapter and the email server. After receipt of the email, the text in the body of the email is processed. You can then apply standard document analysis, validation, transformation, and business logic capabilities to the document.

You can save attachments to a file or pass them through the system using the provided email emit agent. To process the content in the attachment, you must write a custom agent that implements the applicable business logic.

Note: Email listener behavior was modified from that of previous releases to make it more consistent with the rest of the iWay product line. Previously, it created a document for the body of the email regardless of whether the document was XML or not.

The email listener treats the body of a document like a regular listener. If the body of the email is non-XML, then you must use a preparser to handle the document. If you do not use a preparser, an exception occurs. Alternatively, if the input is non-XML, you can select the Accepts non-XML (flat) only property.

Email Listener Properties

The following table lists and describes the email listener properties. For instructions on creating a listener, see *Configuring Listeners* on page 18.

Property Name	Property Description
Incoming Mail Host (required)	The central repository where electronic mail is stored before the recipient downloads it. This value is obtained from your email administrator.
E-mail User (required)	A valid email account on the server. Requires the user ID, not the complete email address. For example, if the email address is abc@company.com, the email user types abc in the configuration tool.

Property Name	Property Description
E-mail Password (required)	A valid password for the email account.
Outgoing Mail Host (required)	The SMTP (Simple Mail Transfer Protocol) host. Location where outgoing electronic mail is placed. Obtain this value from your email administrator.
SMTP User	A valid user name to access the SMTP server.
SMTP Password	A valid password to access the SMTP server.
Subject Filter	The email listener can filter based on the subject header. This listener accepts any email that contains email with the subject in the header. A subject filter of error listens on any pattern that includes the word error. For example, error, RE: error, FWD: error, this is an error, errors.
From Filter	Use this setting if you only want to pick up emails from a specific email address.
Whitespace Normalization	Specifies how the parser treats whitespace in element objects. Select *preserve* (default) to turn off all normalization as prescribed by the XML Specification. Select *condense* to remove extra whitespaces in pretty printed documents and for compatibility with earlier versions.
Mail Protocol	The POP3 (default) or IMAP (Internet Mail Application Protocol).
Default iWay User	The user ID included in a request document if not supplied in the body of the email. Use if a simple SQL is supplied in the email.
Default iWay Password	The password included in a request document if not supplied in the body of the email. Use if a simple SQL is supplied in the email.
Default iWay Server	The data source included in a request document if not supplied in the body of the email. Use this if a simple SQL is supplied in the email.
Mailcap File	The location of the email management file that specifies custom classes for processing various email data types (text or images). Do not change unless instructed by server support.

Property Name	Property Description
Private Key Entry Alias	The alias for the private key entry for decrypting incoming messages and signing receipts (Message Disposition Notification or MDN). Ensure that you exchange keys with your trading partner and load his or her public key into your key store prior to initiating secure document exchanges.
Private Key Password	The password (if required) for private key. If left blank, then the password for the keystore is used.
From	The email address to put into the From field of the MDN.
Accepts non-XML (flat) only	If set to *true*, the input data is sent directly to the business logic step. The data is not preparsed, parsed, or validated. This flag is used primarily to send non-XML to the business logic or replyTo without processing it.
Optimize Favoring	Use this option to customize how the listener performs. For smaller transactions, select *performance*. For large input documents that could monopolize the amount of memory used by iWay Service Manager, select *memory*.
Multithreading	This indicates the number of worker threads (documents or requests) that iWay Service Manager can handle in parallel. Setting this to a value of greater than 1 enables the listener to handle a second request while an earlier request is still being processed. The total throughput of a system can be affected by the number of threads operating. Increasing the number of parallel operations may not necessarily improve throughput. Default: 1 Max Value: 99
Maximum Threads	Parallel threads can grow to this count automatically on demand. Over time, the worker count will decrease back to the multithreading level. Use this parameter to respond to bursts of activity.

Property Name	Property Description
Execution Time Limit	The maximum time that a request takes to complete. Used to prevent runaway requests. Any request that takes longer to complete than this value is attempted to be terminated. (See the kill interval for related information in Global Options.)
Polling Interval	Indicates the frequency (in seconds) with which the listener polls for a new message or a stop request. The listener is constantly connected to the queue to retrieve incoming messages. Default: 2.0 seconds.
Default Java File Encoding	The default encoding if an incoming message is not self-declaring (that is, XML).
Agent Precedence	Sets the order by which iWay Service Manager selects agents. iWay Service Manager selects the agent or agents to process the document by searching through the configuration dictionary. Usually, it looks for a document entry in the configuration and when a match is found, the agent specified in that document entry is selected. If a matching document entry is not found, or no agent is specified, the engine looks in the input protocol configuration (listener). To have the processing agent taken directly from the listener (thus ignoring the document entry), use <listener> overrides <document>. Possible values are <document> overrides <listener> and <listener> overrides <document>. The default value is: <document> overrides <listener>
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined reply-to and error-to destinations.
Error Documents treated normally	If set to *true*, the error documents are processed by any configured preemitters.
Listener is Transaction Manager	If set to *true*, agents run within a local transaction. Agents can roll back uncompleted transactions.

Property Name	Property Description
Record in Activity Log(s)	If set to *true*, the activity on this channel is recorded in the activity logs, else the activity will not be recorded.

Reference: **Email Listener Special Registers**

The following table lists and describes the special registers (SREGs) available on the email listener.

Name	Level	Type	Description
fnfrom	Document	String	The originator of the email with the characters replaced suitable for a file path.
from	Document	String	The originator of the email.
fromaddr	Document	String	The addr portion of from address.
fromalias	Document	String	The alias portion of from address.
to	Document	String	The receiver of the email.
iwayconfig	System	String	The current active configuration name.
msgsize	Document	Integer	The physical length of the message payload.
name	System	String	The assigned name of the master (listener).
protocol	System	String	The protocol on which message was received.
source	Document	String	The originator of the email.
subject	Document	String	The subject of the message.
bcc	Document	String	A Blind Carbon Copy (Bcc) list of recipient addresses of the message (delimited using a semicolon).
cc	Document	String	A Carbon Copy (Cc) list of recipient addresses of the message (delimited using a semicolon).

Name	Level	Type	Description
rcvddate	Document	String	A timestamp indicating when the message was received.
sentdate	Document	String	A timestamp indicating when the message was sent.
tid	Document	String	Unique transaction ID.

Support for Email Attachments

Attachments are files added to an email. Attachments come in many forms, including documents, addresses, calendar entries, programs, software patches, or other pieces of information. Handling attachments on multiple devices is a difficult task. Not all files can run across different device operating systems, and downloading large files is not cost effective in terms of airtime.

Flexibility when handling these message parts is important for any email management system. The iWay email emit agent provides users with the flexibility to efficiently handle messages and message parts.

The email emit agent manages attachments and must be the first agent in an agent chain. The agent optionally adds the attachments to the incoming message or can write the attachments to files on your local system.

For more information on configuring the email emit agent, see the *iWay Service Manager User's Guide*.

Configuring an Email Emitter

Messages are sent to particular destinations at the completion of a workflow. The state of the document determines which destination is used. The order in which a destination is used cannot be predicted.

Note: Configuring an email emitter is not required if the outlet (emitter) protocol is the same as the inlet (listener) protocol.

To route an output document or error message to a protocol other than that of the outlet (listener), you must configure an emitter. For example, an application can send input over TCP, but you want to route the output to an email queue.

Reference: **Email Emitter Properties**

The following table lists and describes the email emitter properties. For instructions on creating an emitter, see *Configuring Emitters* on page 29.

Property	Description
Destination (required)	The email address of the intended recipient.
Outgoing Mail Host (required)	The Simple Mail Transfer Protocol (SMTP) host. This is the location where outgoing electronic mail is placed. Obtain this value from your email administrator.
E-mail User	The user name at the host that can send emails, if the system is a secured email system.
E-mail Password	The password associated with the email user, if the system is a secured email system.
Subject of Msg	The value to insert in the email subject header.
Sender address	The email address of the sender of the message.
Copy To	The optional copy to email address. You can supply multiple email addresses (separated by commas).
Blind Copy To	The optional blind copy to email address. You can supply multiple email addresses (separated by commas).
Content Type	The content type headers for the payload based on MIME standards. For payload, the header is prefixed by application. Use the drop-down list for the preconfigured list of payload types, or type your preferred content type in the entry box.

Property	Description
Security Protocol	Indicate whether to use a security protocol. You can choose from the following: ❏ **None.** Uses a clear SMTP connection. ❏ **SMTPS.** Connects to a secure SMTP server. ❏ **STARTTLS.** Connects to an unsecured SMTP server, then negotiates an SSL/TLS connection.
Packaging (required for S/MIME)	The type of data packaging. This value defines how the payload of the MIME message is packaged. Transmission options are: ❏ **Regular:** No special packaging. ❏ **Signed:** The MIME headers are applied and the document is signed digitally using the server private key (the recipient of the message must have the public key of the signer to validate the signature). ❏ **Encrypted:** MIME headers are applied and the business document is encrypted using the public key of the trading partner. The trading partner's public key must be stored in the keystore, and an alias supplied in Public Key Entry Alias entry. ❏ **Signed and encrypted:** The message is signed (using the private key) and encrypted (using the public key of the trading partner).
Compression	If set to *true*, applies document compression.
Request Receipt (required for S/MIME)	The type of receipt (MDN) required. Options are as follows: ❏ **None:** No MDN required. ❏ **Unsigned:** An unsigned MDN is required. ❏ **Signed:** An MDN signed by the recipient is required (the public key of the trading partner is required to be stored in the keystore file). Signed receipts ensure non-repudiation of the message (for example, that you received the receipt from whom you believed sent it, based on the public/private key pair based validation).

Property	Description
Deliver Receipt To	This is required if asynchronous MDN is also required. For example, MDN is not delivered to the sending email address (specified in the sender address box).
Public Key Entry Alias	The public key entry alias for the trading partner receiving encrypted messages (as defined on the import process into the keystore).
Private Key Entry Alias	The private key alias for the server certificate.
Private Key Password	The password for the private key (if required). If left blank, the password for the keystore file is used.
Digest Algorithm	The message digest algorithm used for creating the digital signature. Options are as follows: ❏ MD5 (Message Digest Algorithm) ❏ SHA1 (Secure Hash Algorithm Version 1)
Encryption Algorithm	The algorithm that encrypts the payload (business document). The options are as follows: ❏ Des-ede3 (known as triple des) ❏ Rc2 (the default setting) ❏ Idea ❏ Cast5

Email Troubleshooting

The following table lists and describes errors that you may encounter when using an email listener.

Error	Reason	Solution

`unable to connect: javax.mail.AuthenticationFailedException:`
 	
`Password supplied for iwaytest7 is incorrect.: [line 6]`
	User ID or password specified is incorrect.	Log on to the email account using an email tool to verify that the user ID and password specified in the configuration console are correct.

`Unable to connect: javax.mail.MessagingException: Connect failed; nested exception is: java.net.UnknownHostException: ibirisc21: [line 3]`	Server is unable to connect to the incoming mail server (POP3 Server).	Use the configuration console to verify that you have correctly entered the POP3 Server. To verify the POP3 Server name, contact your email administrator.
`Unable to connect: javax.mail.MessagingException: Connect failed; nested exception is: java.net.UnknownHostException: ibi.com@ibirisc2: [line 3] unable to login: [line 3]`	Unable to log on to the POP3 Server.	Verify that the user ID is correct. Verify that it is in the format of user name, and not username@host.com.

File

Many technology adapters rely heavily upon polling to originate events. The iWay Adapter for File can monitor a well-known directory for the presence of a file. After it is found, the file is parsed according to metadata from the repository. Events can then be created from this parsed data.

The iWay Adapter for File is designed for the local file system. The adapter polls the local file system in user-defined directories for new files. The contents of the file can be either XML data or non-XML data structures. The adapter can monitor a given directory for the presence of a specific file or series of files.

iWay provides a direct file support to and from the iWay system. Transactionality is maintained at all times. The engine can accept files arriving in any named directory and route these messages to either another directory or to a non-file destination. Similarly, messages received through any of the other engine listeners can be directed to a directory or file. During the flow, the standard document analysis, validation, transformation, and business logic capabilities offered by the iWay server can be applied to the document.

The iWay Adapter for File accepts and processes files arriving on any local file system. As messages arrive, they are accepted by Service Manager and processed. An optional pending facility ensures that messages are retried if appropriate resources are not available at the time of execution.

Service Manager also has the capability to emit a message (either XML or non-XML, for example, HIPAA, SWIFT, HL7) to any file, regardless of the inbound protocol of the message. The adapter deletes the file after it is processed, ensuring that each request is executed only once.

The iWay Adapter for File is extremely useful for systems that can communicate only through batch files. This enables complete integration with old legacy systems.

iWay Adapter for File Listener Properties

The following table lists and describes the File listener properties. For instructions on creating a listener, see, *Configuring Listeners* on page 18.

Property	Description
Input Path (required)	The directory in which input messages are received. A specific file name or DOS-style pattern can be used. Do *not* use file suffix.
Destination (required)	The directory to which output files are received. Specific file name is optional. Use an asterisk (*) in file name to be replaced by time stamp; # by sequential counter.
Removal Destination	Once a file has been processed by the server, it is removed from the input location. The removal destination is configured as a directory to which the handled file is to be moved. If this is not configured, the file is deleted.
Suffix In	This limits input files to those with this extension, for example, xml. Do not insert a period (.) before the suffix. A dash (-) indicates no extension, that is, the field is not used.

Property	Description
Scan subdirectories	If set to *true*, all subdirectories are scanned for files to process.
Do not unzip ZIP files	This property dictates whether ZIP files are passed as a single file for processing. Requires that Accepts non-XML (flat) only be turned on.
Suffix Out	The extension for output files (name is the same as input file unless specified in destination property).
Sort Order	If set to *true*, sort incoming documents by Name or Arrival Time. Maintains sequence, but slows performance.
Pending Queue	The queue to hold documents pending retry.
Duration	The maximum time that a document can remain in the retry pending queue.
Retry	The interval between retrying pending requests.
Maximum File Size	Only files smaller than this value will be processed by the listener. If left blank or set to zero, there will be no limit. The value can be entered with KB or MB, such as 23MB for 23 megabytes, or enter with no suffix to specify size in bytes.
Large File Directory	Full path file pattern asserting where files exceeding the maximum file size will be moved. Use an asterisk (*) in the file name to be replaced by timestamp, # by sequential counter.
Batch Mode	If set to *true*, do not poll until all files from the previous poll have finished processing.
Accept Zero Length Files?	If set to *true*, the listener will accept empty files. The listener must be configured for flat documents if this option is selected.
Inbound Header Namespace	The special register namespace to which metadata values for the input file will be saved.
Maximum Inputs	The maximum number of files that can be accepted for processing in a single poll. Note that if the number of files in the input directory exceeds this, sorting options may not work as expected. Enter 0 or leave blank for no limit.

Property	Description
Whitespace Normalization	Specifies how the parser treats whitespace in element objects. Select *preserve* (default) to turn off all normalization as prescribed by the XML Specification. Select *condense* to remove extra whitespaces in pretty printed documents and for compatibility with earlier versions.
Accepts non-XML (flat) only	If set to *true*, the input data is sent directly to the business logic step. The data is not preparsed, parsed, or validated. This flag is used primarily to send non-XML to the business logic or replyTo without processing it.
Optimize Favoring	Use this option to customize how the listener performs. For smaller transactions, select *performance*. For large input documents that could monopolize the amount of memory used by iWay Service Manager, select *memory.*
Multithreading	This property indicates the number of worker threads (documents or requests) that iWay Service Manager can handle in parallel. Setting this to a value of greater than 1 enables the listener to handle a second request while an earlier request is still being processed. The total throughput of a system can be affected by the number of threads operating. Increasing the number of parallel operations may not necessarily improve throughput. Default: 1 Max Value: 99
Maximum threads	Parallel threads can grow to this count automatically on demand. Over time, the worker count will decrease back to the multithreading level. Use this parameter to respond to bursts of activity.
Execution Time Limit	The maximum time that a request may take to complete. Used to prevent runaway requests. Any request that takes longer to complete than this value will be attempted to be terminated. (See the kill interval for related information in Global Options.)

Property	Description
Polling Interval	The maximum time (in seconds) between checks for the presence of files to be processed or for commands. The higher this value, the longer the interval, and the fewer system resources that are used. The side effect of a high value is that the worker thread will not be able to respond to a stop command or accept a file awaiting processing. If timeout is set to 0, the listener will run once and terminate. Default: 2.0 seconds.
Default Java File Encoding	The default encoding if an incoming message is not self-declaring (that is, XML).
Agent Precedence	Sets the order by which iWay Service Manager selects agents. iWay Service Manager selects the agent or agents to process the document by searching through the configuration dictionary. Usually, it looks for a document entry in the configuration and when a match is found, the agent specified in that document entry is selected. If a matching document entry is not found, or no agent is specified, the engine looks in the input protocol configuration (listener). To have the processing agent taken directly from the listener (thus ignoring the document entry), use <listener> overrides <document>. Possible values are <document> overrides <listener> and <listener> overrides <document>. The default value is: <document> overrides <listener>
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined reply-to and error-to destinations.
Error Documents treated normally	If set to *true*, error documents are processed by any configured preemitters.
Listener is Transaction Manager	If set to *true*, agents run in a local transaction. Agents can roll back uncompleted transactions.
Record in Activity Log(s)	If set to *true*, activity on this channel will be recorded in the activity logs, or else the activity will not be recorded.

Note: The File listener supports streaming. Streaming is used for large documents or documents for which the application needs to split the input into sections under the same transaction. For more information on streaming and configuring streaming preparsers, see the *iWay Service Manager Component and Functional Language Reference Guide*.

Reference: iWay Adapter for File Listener Special Registers

The following table lists and describes the special registers (SREGs) available on the iWay Adapter for File listener.

Name	Level	Type	Description
basename	Document	String	The file name without an extension.
extension	Document	String	The extension to the file name (mime type).
filename	Document	String	The file basename.extension.
iwayconfig	System	String	The current active configuration name.
iwayhome	System	String	The base at which the server is loaded.
iwayworkdir	System	String	The path to base of the current configuration.
msgsize	Document	Integer	The physical length of the message payload.
name	System	String	The assigned name of the master (listener).
parent	Document	String	The path to the file name.
protocol	System	String	The protocol on which the message was received.
source	Document	String	The full name of the input file.
tid	Document	String	Unique transaction ID.

Configuring a File Emitter

Messages are sent to particular destinations at the completion of a workflow. The state of the document determines which destination is used. The order in which the destinations are used cannot be predicted.

This configuration designates where to send response documents when the adapter receives an incoming record that was triggered by a listener.

File Emitter Properties

The following table lists and describes the File emitter properties. For instructions on creating an emitter, see *Configuring Emitters* on page 29.

Property	Description
Destination (required)	Type a destination path. For example, c:\tempout*.xml. This value indicates where the Reply-to document resides. This entry yields document names that include a date and time stamp, such as 2006-11-10T18_57_59_823Z.xml, saved to the c:\tempout directory. Other destination formats include: **Directory.** MyDirectory (relative to the iWay home), or c:\MyDirectory (fully qualified). **Directory and file name.** MyDirectory\MyFile.xml **Directory and file name with a time stamp.** MyDirectory\MyFile*.xml. The asterisk is replaced by a time stamp when the file is written, accurate to a millisecond. This feature can be used to prevent overwriting of previous files since a unique name is guaranteed.
Create Directory	If set to *true*, the emitter creates the destination directory if it does not already exist. If set to *false*, then the emitter does not create the destination directory or the ref file emit Agent (this is the default).

FTP

A well-established standard for exchanging files over TCP/IP networks, File Transfer Protocol (FTP) is also designed for use over networks other than TCP/IP and for exchanging files with a broad variety of machines. FTP works in a client-server fashion, where the local client program connects to the remote FTP system. The client initiates commands to receive or send messages, and the server responds.

For more information on how to configure supported FTP and FTPS components (for example, listeners, emitters, and services) using iSM, see the *iWay FTP Solutions Development Guide*.

SFTP

When you connect to a server using Secure File Transfer Protocol (SFTP), SSH encryption is used to protect the connection between your client machine and the server. This protects your password and your data, preventing an eavesdropper from capturing or stealing them as they travel over the network.

Despite the similarity in name and operation, SFTP is a completely different protocol from FTP and does not support all the same features and commands as FTP. Also, while they are both secure file transfer protocols and have similar names, FTPS (FTP with TLS/SSL) should not be confused with SFTP.

To use SFTP for secure connections, the server you are connecting to must also support SFTP. If you try to connect with SFTP to a server that does not support it, you will receive an error. Your network administrator or service provider can tell you if your server supports SFTP, and what other information you might need to use SFTP if it does.

For more information on how to configure supported Secure File Transfer Protocol (SFTP) components (for example, listeners, emitters, and services) using iSM, see the *iWay FTP Solutions Development Guide*.

FTP Server

iWay offers an FTP server that is designed to extend the processing capabilities of iWay Service Manager (iSM). Although this service can be configured to operate as a standard FTP server, the purpose of this service is transaction receipt and mailboxing. The FTP Server listener listens on the port that is specified in the FTP server settings. The listener begins processing messages when a file is sent through FTP to the iWay server. Therefore, it differs from general FTP servers in several ways:

- User login security is managed through the iSM security facilities. Full function evaluation of all parameters enables the storage of attributes, such as passwords, in a security directory, such as LDAP.
- User attributes can be stored in the Partner Management system of iWay Trading Manager, if installed.
- File actions for Get and Put can be treated as messages. In this case, they are immediately processed through the appropriate configuration.
- Messages received for execution can optionally be safe-stored with their context to prevent loss in the event of failure. Safe-stored messages are reloaded when the listener is started, and executed at that time. Only when all safe-stored messages have been processed does the listener open to receive new messages.

- Input-streaming is supported when the input is to be treated as a message. In input-stream mode, a large document can be broken into parts as it is received. Each part is extracted, based upon the message type configured using a standard streaming preparser, and processed as a separate message.
- Tracing and diagnostics are available using the server tracing system.
- Messages can be stored in the iWay Audit Manager logs for later analysis.

For more information on how to configure supported FTP Server components using iSM, see the *iWay FTP Solutions Development Guide*.

HTTP

The Hypertext Transfer Protocol (HTTP) is an application-level protocol with the lightness and speed required for distributed, collaborative, hyper-media information systems. HTTP enables easy management of agent resources through management applications, such as Firefox, Internet Explorer, and so on, which are readily available in all systems.

The iWay Adapter for HTTP/S provides the iWay Service Manager Server Integration platform with the ability to send and receive messages over secure HTTP. It enables you to take advantage of the Internet by providing secure, reliable, and efficient communication with external business systems, whether within or outside of your enterprise.

The iWay Adapter for HTTP/S:

- Provides a bidirectional adapter.
- Enables HTTP clients to invoke business processes within iWay Service Manager through a URL.
- Enables a business process to request data from an HTTP server through a URL, thus acting as an HTTP client.
- Includes a built-in multi-threaded HTTP listener, as part of the iWay Service Manager base configuration, which is used to serve the iWay Configuration pages to the user.
- Supports HTTP and HTTPS protocols, both synchronous and asynchronous bidirectional invocation, and includes a limited-use license for the XML Data Handler.

For inbound messages, the adapter receives incoming messages through the registered URL, with the message payload being either an XML message or an arbitrary proprietary data structure.

HTTP Listener Properties

The following table lists and describes the HTTP listener properties. For instructions on creating a listener, see *Configuring Listeners* on page 18.

Property	Description
Port (required)	The TCP port for receipt of HTTP requests.
Local bind address	The local bind address for multi-homed hosts: usually leave empty.
Document Root (required)	The base directory from which all HTTP pages are served.
Timeout	The timeout interval for the TCP socket.
Default Page	The default page if no page is identified in the incoming HTTP(S) request.
Response content type	This overrides the content type of a response.
Default Text	The default text sent with 200 OK. Takes the configured Content Type.
Keystore	The full path to the keystore file, which provides certificate material to be used for secure connection.
Keystore Password	The password to access the keystore file.
Keystore Type	The type of keystore file (JKS is the default value).
Truststore	The file that provides trust certificates used to authenticate clients. Leave blank to use the default JVM truststore. For more information on security, see the *iWay Adapter for EDIINT User's Guide*.
Truststore Password	The password to access the truststore file if it is required.
Truststore Type	The type of truststore. For more information on security, see the *iWay Adapter for EDIINT User's Guide*.

Property	Description
Security Provider Class	Overrides the default Sun provider, which is: `com.sun.net.ssl.internal.ssl.Provider`
Security Protocol	The protocol to enable security. Security protocol values include: ❏ **SSL.** Supports some version of SSL. May support other versions. ❏ **SSLv2.** Supports SSL version 2 or higher. ❏ **SSLv3.** Supports SSL version 3. May support other versions. ❏ **TLS.** Supports some version of TLS. May support other versions. ❏ **TLSv1.** Supports TLS version 1. May support other versions.
Security Algorithm	This overrides the default security algorithm (SunX509).
Client Authentication	If set to *true*, then authentication is required from the client.
Require Authorization	If set to *true*, the listener implements HTTP basic authentication using one or more authorization drivers.
Whitespace Normalization	This specifies how the parser treats whitespace in element objects. Select *preserve* (default) to turn off all normalization as prescribed by the XML Specification. Select *condense* to remove extra whitespaces in pretty printed documents and for compatibility with earlier versions.
Accepts non-XML (flat) only	If set to *true*, then the input data is sent directly to the business logic step. The data is not preparsed, parsed, or validated. This flag is used primarily to send non-XML to the business logic or replyTo without processing it.
Optimize Favoring	Use this option to customize how the listener performs. For smaller transactions, select *performance*. For large input documents that could monopolize the amount of memory used by iWay Service Manager, select *memory.*

Property	Description
Multithreading	This indicates the number of worker threads (documents or requests) that iWay Service Manager can handle in parallel. Setting this to a value of greater than 1 enables the listener to handle a second request while an earlier request is still being processed. The total throughput of a system can be affected by the number of threads operating. Increasing the number of parallel operations may not necessarily improve throughput. Default: 1 Max Value: 99
Maximum threads	Parallel threads can grow to this count automatically on demand. Over time, the worker count will decrease back to the multithreading level. Use this parameter to respond to bursts of activity.
Execution Time Limit	The maximum time that a request may take to complete. Used to prevent runaway requests. Any request that takes longer to complete than this value will be attempted to be terminated. (See the kill interval for related information in Global Options.)
Default Java File Encoding	The default encoding if incoming message is not self-declaring (that is, XML).
Agent Precedence	This sets the order by which iWay Service Manager selects agents. iWay Service Manager selects the agent or agents to process the document by searching through the configuration dictionary. Usually, it looks for a document entry in the configuration and when a match is found, the agent specified in that document entry is selected. If a matching document entry is not found, or no agent is specified, the engine looks in the input protocol configuration (listener). To have the processing agent taken directly from the listener (thus ignoring the document entry), use <listener> overrides <document>. Possible values are <document> overrides <listener> and <listener> overrides <document>. The default value is: <document> overrides <listener>
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined reply-to and error-to destinations.

Property	Description
Error Documents treated normally	If set to *true*, the error documents are processed by any configured preemitters.
Listener is Transaction Manager	If set to *true*, the agents run within a local transaction. Agents can roll back uncompleted transactions.
Record in Activity Log(s)	If set to *true*, then activity on this channel is recorded in the activity logs. If set to *false*, the activity will not be recorded.

Note: The HTTP listener supports streaming. Streaming is used for large documents or documents for which the application needs to split the input into sections under the same transaction. For more information on streaming and configuring streaming preparsers, see the *iWay Service Manager Component and Functional Language Reference Guide*.

Reference: HTTP Listener Special Registers

The following table lists and describes the special registers on the HTTP listener.

Name	Level	Type	Description
...	Header	String	Each header value from the message.
action	Document	String	The action field of the post.
docroot	Document	String	The defined docroot from configuration.
ip	Document	String	The IP address of the sender.
iwayconfig	System	String	The current active configuration name.
iwayhome	System	String	The base at which the server is loaded.
iwayworkdir	System	String	The path to the base of the current configuration.
msgsize	Document	Integer	The physical length of the message payload.
name	System	String	The assigned name of the master (listener).
protocol	System	String	The protocol on which message was received.

Name	Level	Type	Description
requestType	Header	String	The type of HTTP request (GET, POST, or HEAD).
source	Document	String	The host name of the sender.
url	Header	String	The type of full URL of the HTTP (GET, POST, or HEAD).
tid	Document	String	Unique transaction ID.

HTTP Emitter Properties

The following table lists and describes the HTTP emit properties. For instructions on creating an HTTP emitter, see *Configuring Emitters* on page 29.

Property	Description
Destination (required)	The URL to which to post this information.
Action Method	Select *GET* (with data on the URL and URL encoded) or *POST* (with a content length header).
Response content type	This overrides response content type value. Choose from among the following values: ❑ application/EDI-X12 ❑ application/EDIFACT ❑ application/XML ❑ text/html ❑ text/plain
User ID	A valid user ID for basic authentication challenges.
Password	A password associated with the user ID.

Property	Description
Response Timeout value in seconds	The time (in seconds) to wait for response before signaling an error as integer.
IP Interface Host	The local IP interface from which the outgoing IP socket originates.
IP Interface Port	The local IP port from which the outgoing IP socket originates.
Relay Inbound Content Type	If set to *true*, then the relay headers are received as content type.
Set TCP No Delay	If set to *true*, then this parameter disables the Nagle Algorithm on the client socket. This results in a faster line turnaround at the expense of an increased number of packets.
Proxy	If set to *true*, emit through proxy server.
Proxy URL	The URL of the proxy server.
Proxy User ID	The user ID for proxy authentication challenges.
Proxy Password	The password to access proxy server.
Secure Connection	If set to *true*, then the emitter uses a secure connection. You may be required to configure the keystore under the HTTPS section of the system properties if client authentication is required, or if a certificate is used that does not have a matching entry in the default truststore of the JVM.
Use 128-bit Encryption	Select *true* to use 128-bit encryption.

Property	Description
Security Protocol	Select from the drop-down list. **SSL.** Supports some version of SSL. May support other versions. **SSLv2.** Supports SSL version 2 or higher. **SSLv3.** Supports SSL version 3. May support other versions. **TLS.** Supports some version of TLS. May support other versions. **TLSv1.** Supports TLS version 1. May support other versions.

nHTTP

The nHTTP adapter is a nonblocking HTTP with improvement in performance, connection management, and various other security features.

The nHTTP adapter provides extensive flexibility by exposing an array of configurable parameters for security, connectivity, and header manipulation. Below are descriptions of some features that have been added as part of the improvement to the nHTTP component.

nHTTP REST Support

Representational State Transfer (REST) is a simpler alternative to SOAP and Web Services Description Language (WSDL)-based web services. There are many advantages of using this simpler HEEP-based design approach to calling server-based services, and it has been widely adopted by many advanced web service providers including Amazon, Google, and Facebook. These entities have either avoided SOAP or offered REST as a simpler alternative.

A RESTful service:

- Uses explicit HTTP methods.
- Is stateless (no session management is allowed or required).
- Exposes the calling URI as a directory-like structure.
- Transfers the payload as XML, JavaScript Object Notation (JSON), or both.

iWay Service Manager (iSM) offers a group of facilities to enable the use of RESTful services when executed by process flows. The nHTTP listener for iSM complies with the HTTP 1.1 specification. This listener implements all of the available verbs used in REST-style communication, including the main verbs (GET, POST, PUT, and DELETE).

iWay Providers

Named SSL Context Provider

Since this provider uses configured keystore/truststore providers, it allows you to configure multiple SSL context providers and use them as named providers in the nHTTP configuration.

Features

- **Persistent Connection Support.** The nHTTP adapter supports persistent connections, which will allow for better connection handling and management.
- **Session Resumption.** Session resumption is one of the new features available for the SSL configuration.
- **Large File Limit.** The nHTTP adapter contains various internal improvements to handle large file sizes. As an addition, a new option has been exposed on the nHTTP inbound processing that allows the user to limit the message size accepted by the adapter.

Configuring nHTTP Listeners

A listener is a component that is responsible for receiving inbound messages through an assigned listener protocol. After a listener is created, it must be added to an inlet configuration. An inlet will become part of the final channel configuration that will consist of an inlet, route, and an outlet. For more information on configuring channels, see the *iWay Service Manager User's Guide*.

Procedure: How to Configure a nHTTP Listener

To configure a nHTTP listener:

1. Ensure that iWay Service Manager is running.

   On Windows, you can start iWay Service Manager by clicking *Start*, selecting *Programs*, *iWay 7.0 Service Manager*, and then *Start Service Manager* for the configuration you are currently using.

   For more information on starting and stopping iWay Service Manager, see the *iWay Service Manager User's Guide*.

2. Open a browser window and point to the following URL:

`http://host:port/ism`

where:

`host`

Is the host machine on which iWay Service Manager is installed.

`port`

Is the port on which iWay Service Manager is listening. The default port is 9999.

On Windows, alternatively, you can click *Start*, select *Programs*, *iWay 7.0 Service Manager*, and then click *Console*.

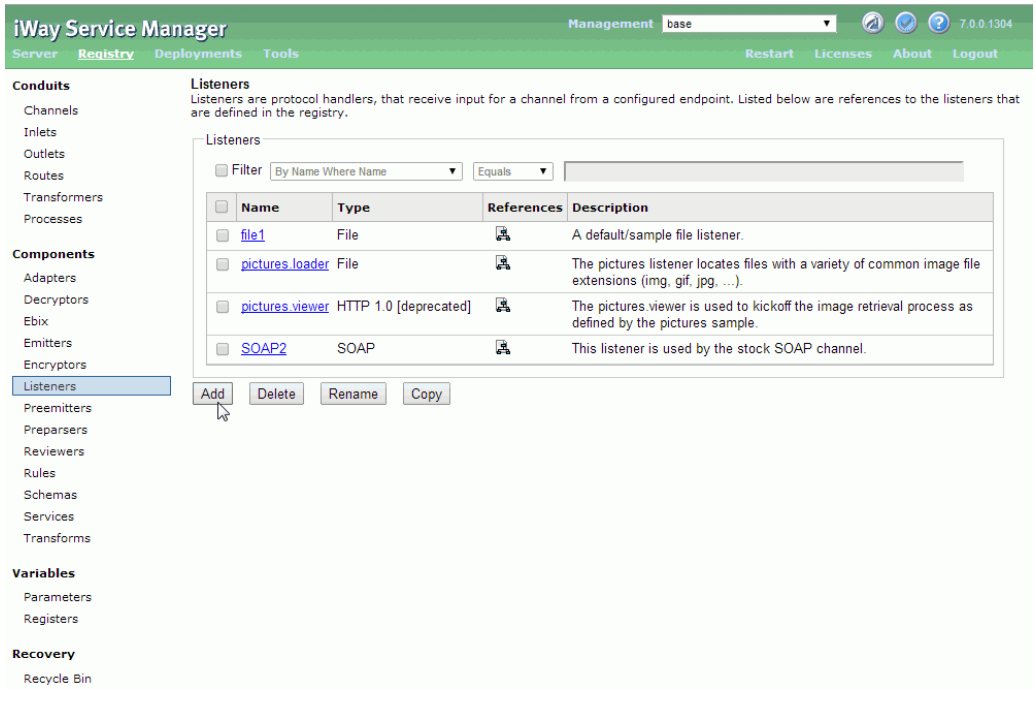
A login dialog box opens.

3. Type a user name and password for the configuration you are using, and click *OK*.

   The iWay Service Manager Administration Console opens.

4. Click *Registry* in the top pane, and then click *Listeners* in the left pane.

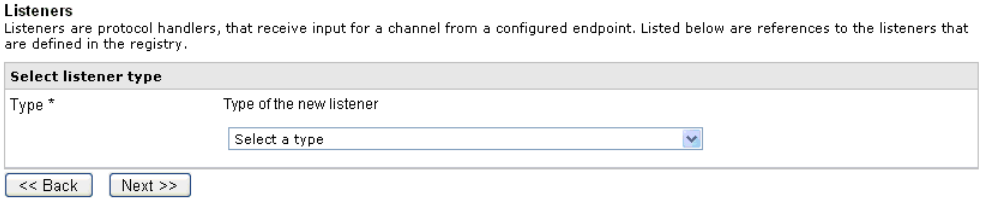
   The Listeners pane opens.



The table that is provided lists all the previously configured listeners and a brief description for each.

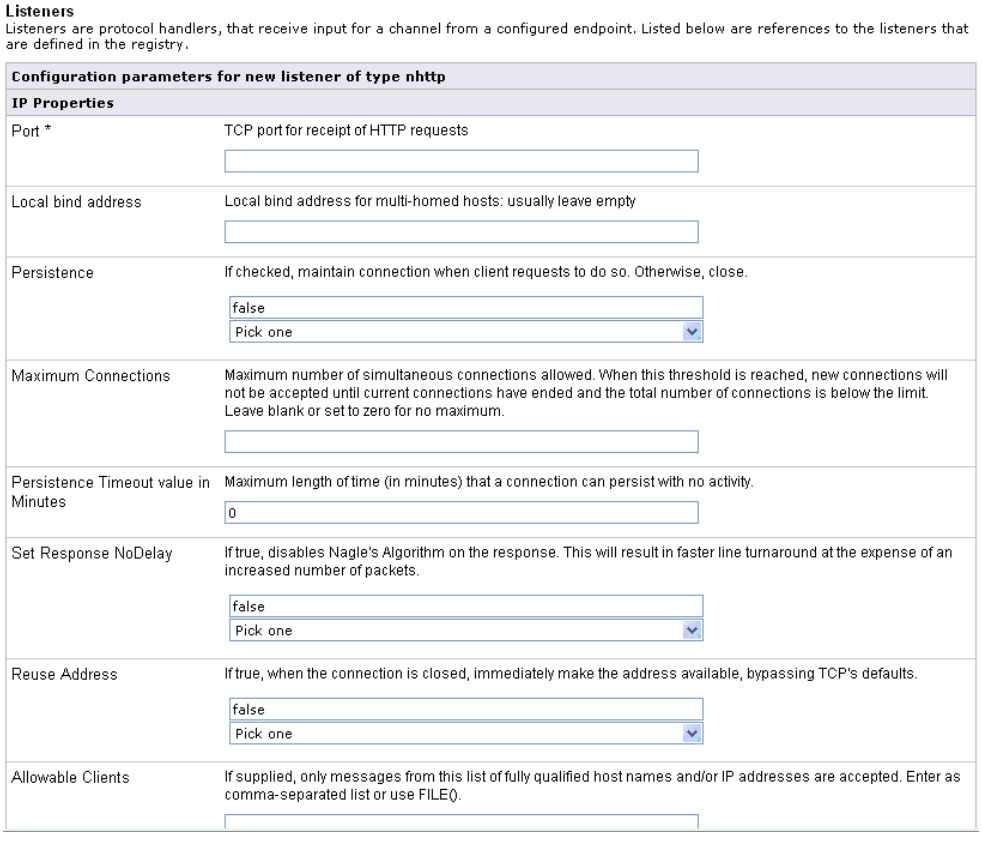
5. Click *Add*.

   The Select listener type pane opens.



6. Select *nhttp* from the Type drop-down list and click *Next*.

   The Configuration parameters for the nHTTP listener pane opens.



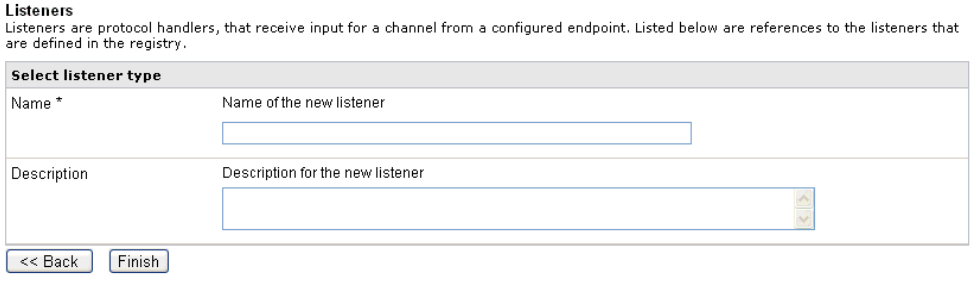
Note: The parameters prefixed with an asterisk (*) in the listener configuration pane are required.

7. Provide the appropriate values for the nHTTP listener parameters.

   For more information, see *nHTTP Listener Configuration Parameters* on page 187.

8. Click *Next*.

   You are returned to the Select listener type pane.



9. Enter a name for the nHTTP listener and description (optional).
10. Click *Finish*.

    You can now use this listener as part of your channel configuration where the business logic will be applied to the received messages.

Reference: nHTTP Listener Configuration Parameters

The following table lists and describes parameters for the nHTTP listener.

Parameter	Description
IP Properties	
Port	The TCP port for receipt of HTTP requests.
Local bind address	The local bind address for multi-homed hosts. This parameter value is usually not specified.
Persistence	If set to *true*, then the connection is maintained when the client requests to do so. If set to *false*, the connection is closed.

Parameter	Description
Maximum Connections	This defines the maximum number of simultaneous connections that are allowed. When this threshold is reached, new connections will not be accepted until current connections are closed and the total number of connections is below the limit. Leave this field blank (default) or set a value of zero to have no maximum limit of connections.
Persistence Timeout value in Minutes	The maximum length of time that a connection persists with no activity.
Set Response No Delay	If set to *true*, the Nagle Algorithm is disabled on the response. This will result in a faster line turnaround at the expense of an increased number of packets.
Reuse Address	If set to *true*, then when a connection is closed, it immediately makes the address available, bypassing the TCP defaults.
Allowable Clients	If supplied, then only messages from this list of fully qualified host names and/or IP addresses are accepted. Accepts comma-separated list or use the FILE() function.
Secure Connection (SSL)	
Secure Connection	If set to *true*, then a connection using secure HTTP (HTTPS) is made.
SSL Context Provider	The named iWay Security provider for SSL Context.
General Properties	
GET Handling	This determines how GET requests are handled. Options include: ❑ docroot. Attempts to serve a file from the document root directory. ❑ error. Returns an HTTP 405 Method Not Allowed. ❑ event. Generates an event message.

Parameter	Description
PUT and DELETE Handling	This determines how PUT and DELETE requests are handled. Options include: ❑ unavailable. Returns an HTTP 405 Method Not Allowed. ❑ event. Generates an event message.
Document Root	The base directory from which all HTTP pages are served through GET if GET Handling is enabled for page access.
Default Page	The default page displayed if no page is identified in the incoming HTTP[s] request.
Default Text	The default text sent with 200 OK, which will take configured ContentType.
Response content type	This overrides the content type of the response.

Parameter	Description

HTTP Response Code	Indicates the HTTP status code to send with the response. Usually this is left blank, which allows the channel to determine the appropriate response code.
	
	You can specify an iWay Functional Language (iFL) expression as a value for this parameter to be evaluated during the final emit process. If you specify an iFL expression, then ensure to include a leading backtick (`) character to prevent this expression from being evaluated until it is required. For example:
	
	`` `sreg(responsecode) ``
	
	If you decide to use a Special Register (SREG), then ensure to set it in the Message scope. The Message scope survives the process flow that elects to set the value. For example:
	
	`message:responsecode | 500 |`
	
	For more information on setting SREGs into scopes, see *SREG Service (com.ibi.agents.XDSREGAgent)* in the *iWay Service Manager Component Reference Guide*.
	
	The status code from nHTTP emits during the process flow are stored in a SREG called *httpstatus*, which is in the defined response SREG namespace of the emit service.

Authentication Scheme	The scheme to apply when authenticating HTTP requests. Select one of the following options from the drop-down list: ❑ Digest Auth ❑ Basic Auth ❑ None (Default)
Authentication Realm	If authentication is required, then this provides the name of the configured Realm provider to use.
Request Header Namespace	The special register namespace to which HTTP headers from the incoming requests are saved. The Default option creates HDR type special registers without a namespace prefix.

Parameter	Description
Response Header Namespace	The special register namespace from which HTTP headers for the outgoing response are taken. The Default option sends HDR type registers with no namespace prefix. If none is selected, no special registers are sent as HTTP headers.
Response Main Part Header Namespace	The special register namespace from which MIME headers for the outgoing response are taken. Provide a prefix to control the response Main BodyPart headers in the presence of attachments. Selecting *none* means that no special registers are sent as MIME headers. An empty namespace prefix is treated as *none*.
Maximum Request Entity size	When a request document is received that is larger than the specified maximum size, the listener will return HTTP 413 Request Entity Too Large and close the connection. A value of 0 specifies no maximum. The default size is 256KB.
Excluded Headers	A comma delimited list (case-insensitive) of headers that should not be sent with the response, even if they are found in the response header namespace.
Compress Response	If set to *true*, then the response is compressed with gzip or deflate compression when the client indicates that it can accept compressed transfer encoding.
Other	
Whitespace Normalization	This specifies how the parser treats whitespace in element objects. Select *preserve* (default) to turn off all normalization as prescribed by the XML Specification. Select *condense* to remove extra whitespaces in pretty printed documents and for compatibility with earlier versions.
Accepts non-XML(flat) only	If set to *true*, the listener expects flat (non-XML) documents. Preparsers do not run.
Optimize Favoring	The selection of memory is useful for large input documents.
Multithreading	The number of documents that can process in parallel.

Parameter	Description
Execution Time Limit	The time limit for document execution (in seconds) before it is terminated.
Default Java File Encoding	The default encoding if incoming message is not self-declaring.
Agent Precedence	Changes the order by which iSM selects agents. This is normally set to *Document overrides listener*.
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined replies.
Error Documents treated normally	If set to *true*, the error documents are processed by any configured pre-emitters.
Listener in Transaction Manager	If set to *true*, the agents run within a local transaction managed by the listener.
Record in Activity Log(s)	If set to *true*, activity on this channel will be recorded in the activity logs. If set to *false*, the activity will not be recorded.

Note: The nHTTP listener supports streaming. Streaming is used for large documents or documents for which the application needs to split the input into sections under the same transaction. For more information on streaming and configuring streaming preparsers, see the *iWay Service Manager Component and Functional Language Reference Guide*.

Reference: Special Registers for the nHTTP Listener

The following table lists and describes the special registers for the nHTTP listener.

Special Register	Level	Description
...	Header	Each header value from the message.
action	Document	The action field of the post.
docroot	Document	The defined docroot from configuration.
ip	Document	The IP of the sending system.

Special Register	Level	Description
iwayconfig	System	The current active configuration name.
iwayhome	System	The base at which the server is loaded.
iwayworkdir	System	The path to base of the current configuration.
msgsize	Document	The physical length of the message payload.
name	System	The assigned name of the master (listener).
protocol	System	The protocol on which the message was received.
requestType	Header	The type of HTTP request (GET, POST, or HEAD).
source	Document	The host name of the sending system.
url	Header	The full URL of the HTTP request (GET, POST, or HEAD).
tid	Document	Unique transaction ID.

Associating Session Information With an HTTP Interaction

In HTTP, a session is a sequence of network request-response transactions. A session may encompass a group of console screens or web interactions for a specific purpose. In transactional HTTP, such as REST or web services, a session may represent one or more request-response activities, such as sending a group of related shipping operations.

Applications can associate session information with an HTTP interaction. The session information is not actually carried between the client and the server. Rather, a token is assigned by iSM, which is carried between interactions. The token identifies the current session, much as a transaction ID represents the action of a single transaction within the session. By not carrying the session information between interactions, security is enhanced and network traffic is reduced.

Session information is carried in Special Registers (SREGs), which are created by the application and available in the later steps where they can be referenced and changed as required. The session SREGs are carried in a SREG scope called session. The session scope is not a namespace, although namespaces can be used within the session.

Registers in a specific scope can be set using the Special Register Setting Service (com.ibi.agents.XDSREGAgent). These registers can be referenced by the syntax *session:name*. For example, a database key carried in the dbkey SREG would be referenced as *session:dbkey*. The colon identifies the register as being in a named scope.

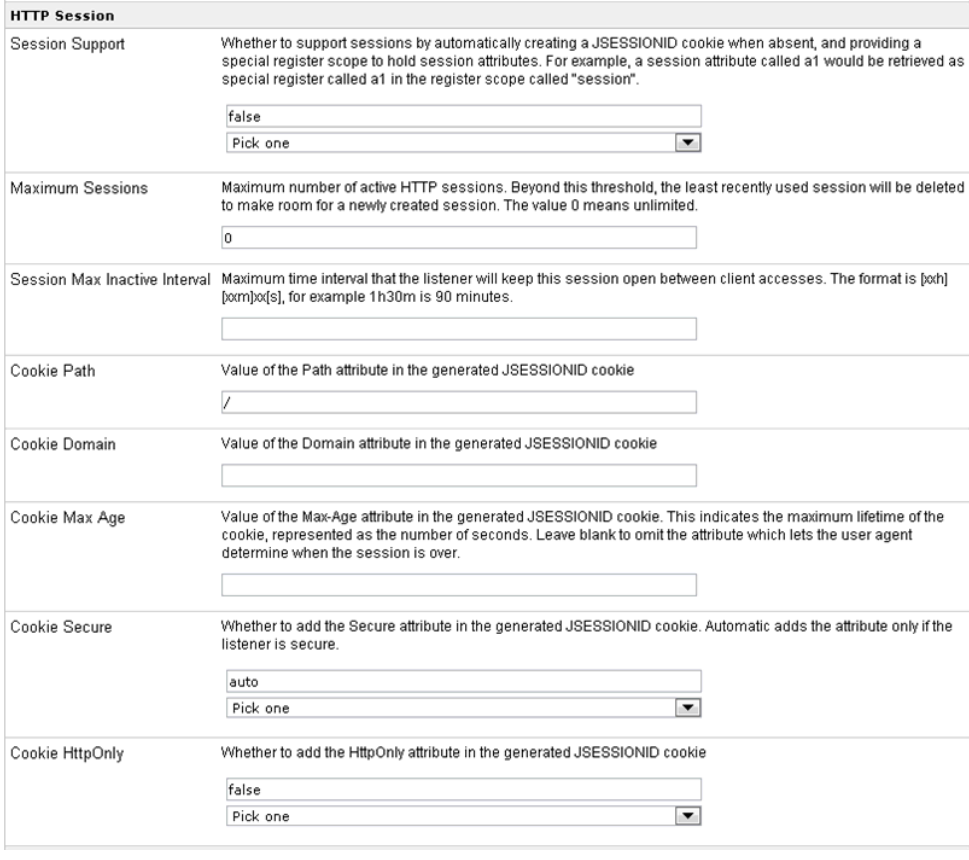
Note: Users are cautioned that scopes (denoted by the colon) are not namespaces. It is possible to use namespaces within the session scope (as it is in any scope), but usually in session scope namespaces do not add facility.

The session registers can be assigned to a type, such as USER, METRIC, or HDR. User registers can optionally be carried between channels within a transaction. All register attributes, including marshalling control and context recording can apply to session registers.

The session key is exchanged with the client by using the standard JSESSIONID cookie. As a result, management of the session involves dealing with the treatment of this cookie.

Configuring Sessions on an nHTTP Listener

Sessions are configured on the nHTTP listener in the HTTP Session section. Set the Session Support parameter to *true* to enable sessions.



The details of the session definition extend beyond the scope of this document. For more information, application designers should refer to other HTTP documentation.

The values specified for the Maximum Sessions and Session Max Inactive Interval parameters are important. Either of these settings can result in the loss of the session information for the session, even though a session token is received from the client. In such a case, the application must be designed to handle the loss of the session and session restart. For example, a shopping cart might be maintained in the session information. If the session expires, the shopping cart would be deleted and the application must reacquire the information that has been eliminated. While setting the listener to prevent expiration seems to avoid this situation, application designers must be aware that clients can be abandoned, resulting in the loss of the resources used to hold the session information in the server.

Because the session information is exchanged as a cookie (JSESSIONID), the cookie attributes can be applied. The cookie attributes are a cookie domain, a path, expiration time or maximum age, secure flag, and HttpOnly flag. Browsers will not return cookie attributes to the server. They will only send the name-value pair of the cookie. Cookie attributes are used by browsers to determine when to delete a cookie, block a cookie, or whether to send a cookie (name-value pair) to the server. The default entries for these attribute fields allow the cookie (session information) to be exchanged for all interactions.

The following table lists and describes the cookie parameters in the HTTP Session section.

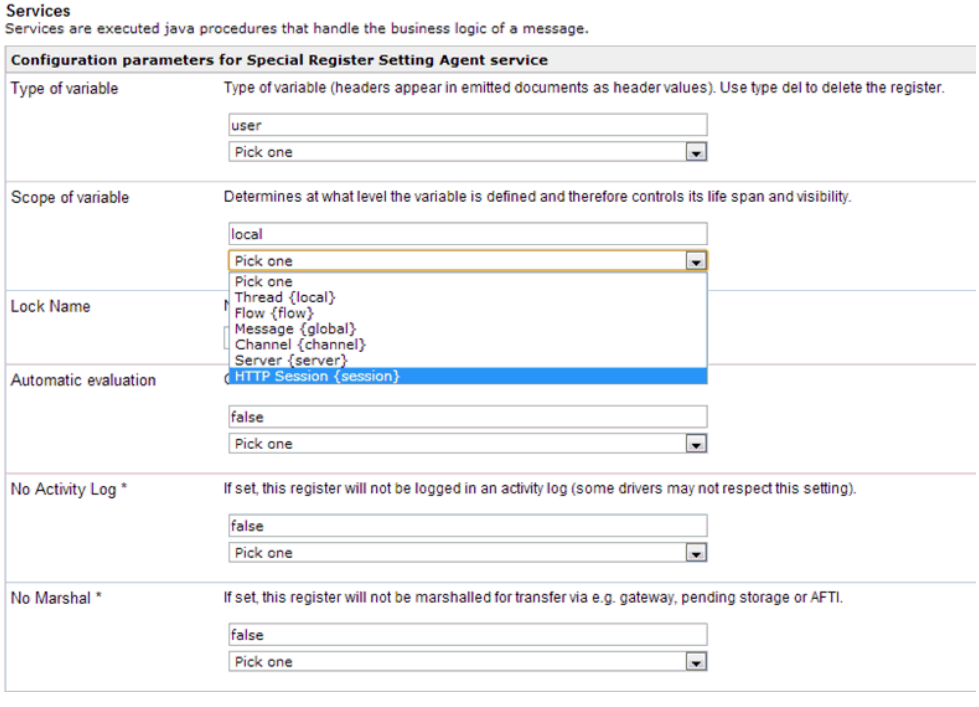
Parameter	Description
Cookie Path	Restricts the exchange of the cookie to the identified domain and optionally its subdomains. Most applications for iSM sessions will not need these settings.
Cookie Domain	
Cookie Max Age	The number of seconds that the JSESSIONID cookie can exist. This is the total time the session can exist once it is created.
Cookie Secure	The Secure attribute is meant to keep cookie communications limited to encrypted transmissions, directing browsers to use cookies only through secure and encrypted connections.

Parameter	Description
Cookie HTTP Only	Directs browsers (or other clients) to use the JSESSIONID cookie through the HTTP protocol only (this includes HTTPS. HttpOnly is not the opposite of Secure). An HttpOnly cookie is not accessible through non-HTTP methods, such as calls using JavaScript (for example, referencing document.cookie), and therefore cannot be stolen easily through cross-site scripting

Using Session Information in an Application

Setting the session information for an application is configured by using the Special Register Setting Service (com.ibi.agents.XDSREGAgent).

From the Scope of variable drop-down list, select *HTTP Session {session}*, as shown in the following image.



You must have the session support configured for the channel in order for the register to automatically appear in the next client interaction.

To reference the value of the register, you can use the following function from the iWay Functional Language (iFL):

```
_sreg('session:dbkey')
```

The HTTP Session Invalidator Service (com.ibi.agents.XDHttpSessionInvalidator) can be used in a process flow to invalidate the current session. This deletes all information in the session and prevents the session from being exchanged in subsequent client-server interactions. As a best practice, you can call for invalidating (deleting) the session once it is no longer required. An example would be a console logout or a determination of a catastrophic error situation that requires the user to restart an operation.

An application might keep an event count in the session, such that a count of 0 means this is the start of a session. Using 0 as the default value of an _sreg() lookup in a process flow test, the process flow can take whatever action is needed to begin the application session. Following the test, a Special Register Setting Service (com.ibi.agents.XDSREGAgent) might set the event count to the command shown in the following table:

eventcount	`_sreg('session:eventcount','0')+1`

HTTP Session Invalidator Service (com.ibi.agents.XDHttpSessionInvalidator)

This service is used to terminate the session. The following table lists and describes its parameter.

Parameter	Description
Expire Cookie	Determines how the termination is to be effected. If set to *false*, it removes the session information from the server. No further action is taken. If set to *true*, in addition to removing the session information, it sends an instruction to the client to delete the JSESSIONID cookie itself.

The service returns the input document on the success edge.

As a good practice, you can use this service when the application has completed the session, so as to reduce server resources.

Configuring Emit Services

You can configure outbound processing of HTTP messages as a service that can be used within a process flow, which will become part of the route configuration or directly as a service assigned to a route. In this case, a business process can continue after an HTTP message has been sent out to the client. The following section describes how to configure an HTTP nonblocking emit service. For more information on configuring outlets and routes, see the *iWay Service Manager User's Guide*.

Procedure: How to Configure an HTTP Nonblocking Emit Service

To configure an HTTP nonblocking emit service:

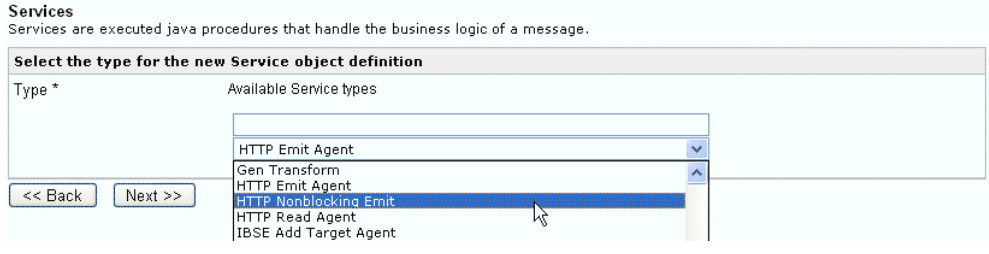
1. Click *Registry* in the top pane, and then click *Services* in the left pane.

   The Services pane opens.

   The table that is provided lists all the previously configured services and a brief description for each.

2. Click *Add*.

   The Select Service type pane opens.



3. Select *HTTP Nonblocking Emit* from the Type drop-down list.
4. Click *Next*.

   The configuration parameters pane for the HTTP nonblocking emit service opens.

5. Provide the appropriate values for the HTTP nonblocking emit service parameters.

   For more information, see *HTTP Nonblocking Emit Service Configuration Parameters* on page 200.

6. Click *Next*.

   The name and description pane opens.

7. Enter a name for the service and description (optional).

8. Click *Finish*.

Reference: HTTP Nonblocking Emit Service Configuration Parameters

The following table lists and describes parameters for the HTTP nonblocking emit service.

Parameter	Description
Configuration Parameters	
Destination (required)	The destination URL to post information that uses the following format: `http[s]://host[:port]/action`
HTTP Client Provider (required)	The HTTP client Provider that is used to manage connections for this emitter.
Action Method	Select one of the following supported methods from the drop-down list: ❏ GET method with data on the URL and URL Encoded. ❏ HEAD method. ❏ POST method with a Content-Length header. This value is selected by default.
Request Content Type	The content type for the HTTP request sent by this emitter. Select a value from the drop-down list or provide your own. Available values from the drop-down list include: ❏ application/EDI-X12 ❏ application/EDIFACT ❏ application/XML ❏ text/html ❏ text/plain
User ID	The user ID for Basic Authentication challenges.
Password	The password for Basic Authentication challenges.

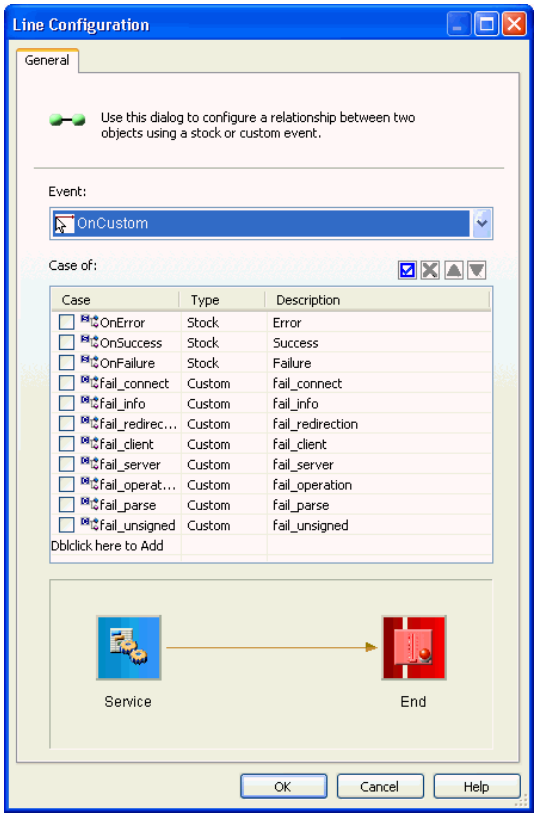
Parameter	Description
Domain	The domain for NTLM authentication challenges. Note that to use NTLM, you must enable connection persistence.
Request Header Namespace	The special register namespace from which HTTP headers for the outgoing request will be taken. Choose Default Namespace to send HDR type registers with no namespace prefix, or supply a namespace prefix here. None means that no special registers will be sent as HTTP headers. ❏ Default Namespace to send HDR type registers with no namespace prefix. ❏ Supply a namespace prefix here to indicate which headers to send. ❏ none means that no special registers will be sent as HTTP headers.
Request Main Part Header Namespace	The special register namespace from which MIME headers for the outgoing request will be taken. Provide a prefix to control the request Main BodyPart headers in the presence of attachments. Selecting *none* means that no special registers will be sent as MIME headers.
Response Header Namespace	The special register namespace into which HTTP headers from the incoming response will be saved. Choose Default Namespace to create special registers with no namespace prefix, or supply a namespace prefix here. None means that no special registers will be created. ❏ Default Namespace to create special registers with no namespace prefix. ❏ Supply a namespace prefix here to indicate header namespace. ❏ Empty namespace prefix will be treated as default.

Parameter	Description
Excluded Headers	A comma delimited list (case-insensitive) of headers that should not be sent with the request, even if they are found in the request header namespace.
Ask for Compressed Response	If set to *true*, the request will set the accept-encoding to indicate that the client can accept a compressed response. If the response has a compressed content encoding, the client will automatically inflate the response.
Compress Request	If set to *true*, the request entities will be compressed using the selected encoding and the content-encoding header will be set accordingly.
Replace Connection?	If set to *false*, the connection is not returned to the connection pool immediately. The identifier of the connection will be stored in the httpclient-key special register and the connection can be handled by the HTTP Client Manager agent.
Maximum HTTP Client Manager Delay	The maximum time for the HTTP Client Manager to handle a particular connection before it is automatically aborted. The format is [xxh][xxm]xx[s]. The default is 60 seconds.
Try Expect/Continue Handshake?	If set to *true*, the client will send the HTTP Expect: 100-continue header and await HTTP 100 response before sending the request body. By default, *false* is selected.
Chunk Encoded Request?	If set to *true*, the request entity will be sent with chunk encoding. By default, *false* is selected.
Maximum Request Size	The maximum size (after compression) of a request entity that can be sent with this emitter. A value of zero (0) means there is no maximum size limit and if no value is specified, the default value of 256KB is applied.
Maximum Response Size	The maximum size of a response entity that can be received by this emitter. A value of zero (0) means there is no maximum size limit and if no value is specified, the default value of 256KB is applied.
IP Properties	

Parameter	Description
Persistence	If set to *true*, the server is requested to maintain the connection.
Response Timeout value in seconds	The value in seconds to wait for a response before generating an error. The default value is 60 seconds.
Agent Specific Parameters	
Return	The type of return from this agent. Select *input* to return input document, *status* for an XML document with transaction parameters and status, or *response* to capture the output from the server.
Preemitter	If set to *true*, the preemitters will not run.
Response Wrapper Tag	The tag name with which to wrap the response if the response is non-XML and must be XML.
Response Base64 Encoded	If set to *true*, the response will use Base64 encoding.

Available Response Edges for nHTTPEmitAgent

When you connect the nHTTPEmitAgent object to an End object using the *OnCustom* build relation in a process flow, the available line edges are provided in the Line Configuration dialog box.



The following table lists and describes the available line edges for the nHTTPEmitAgent object.

Line Edge	Description
OnError	Error
OnSuccess	Success
OnFailure	Failure
fail_connect	fail_connect

Line Edge	Description
fail_info	fail_info
fail_redirection	fail_redirection
fail_client	fail_client
fail_server	fail_server
fail_operation	fail_operation
fail_parse	fail_parse
fail_unsigned	fail_unsigned

nHTTP Samples

This section provides additional information about the nHTTP listener and includes samples you can use as a reference.

nHTTP Listener Event Schema

The nHTTP listener allows you to configure the handling of incoming HTTP requests. For example, the available options for the GET Handling parameter include docroot, error, and event. If you select event, an event document is created for the incoming request. This document can then be used in your process to determine an action for the request. The event document corresponds to the following structure:

```
<http user="auto" type="GET">
  <parms>
    <parm name="ibse-port">9000</parm>
    <parm name="Host">clientbox:10000</parm>
    <parm name="Connection">Keep-Alive</parm>
    <parm name="pdm">0</parm>
    <parm name="version">1.1</parm>
    <parm name="Accept-Language">en-AU</parm>
    <parm name="action">user.req?
user=999999999&account=1234567890123456&tranid=tid1234</parm>
    <parm name="Accept">text/html, application/xhtml+xml, */*</parm>
    <parm name="User-Agent">Mozilla/5.0 (compatible; MSIE 9.0; Windows NT
6.1; WOW64; Trident/5.0)</parm>
    <parm name="url">/user.req?
user=999999999&account=1234567890123456&tranid=tid1234</parm>
    <parm name="ip">127.0.0.1</parm>
    <parm name="source">hostname unknown</parm>
    <parm name="Accept-Encoding">gzip, deflate</parm>
    <parm name="reqType">GET</parm>
  </parms>
  <body />
  <url secure="false">
    <host>clientbox</host>
    <port>10000</port>
    <path>/user.req</path>
    <query>user=999999999&account=1234567890123456&tranid=tid1234</query>
    <queryparms>
      <queryparm name="user">99999999</queryparm>
      <queryparm name="account">1234567890123456</queryparm>
      <queryparm name="tranid">tid1234</queryparm>
    </queryparms>
    <incomingurl>    http:clientbox:1000/
user=999999999&account=1234567890123456&tranid=tid1234
   </incomingurl>
  </url>
  <version>1.1</version>
</http>
```

The following syntax is a sample document for the GET event:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<http user="unknown" type="GET">
<parms>
   <parm name="version">1.1</parm>
   <parm name="source">beck-xp.ibi.com</parm>
   <parm name="ua-cpu">x86</parm>
   <parm name="reqType">GET</parm>
   <parm name="accept-encoding">gzip, deflate</parm>
   <parm name="accept-language">en-us</parm>
   <parm name="connection">keep-alive</parm>
   <parm name="url">/TEST?A=B&C=D</parm>
   <parm name="user-agent">Mozilla/4.0 (compatible; Win 5.1</parm>
   <parm name="host">theservercom:7777</parm>
   <parm name="ip">172.19.22.60</parm>
</parms>
<body />
<url secure="true">
   <host>theserver.com</host>
   <port>7777</port>
   <path>/TEST</path>
   <query>A=B&C=D</query>
</url>
<version>1.1</version>
</http>
```

Supported nHTTP Requests

The following table lists the supported HTTP requests that can be processed by the nHTTP listener. Flow refers to the generation of the event signal document that can be processed within a process flow. A Reject action causes the client request to be rejected with an HTTP 405 Method Not Allowed response.

Request Type	Available Actions
GET	From File, Flow, or Reject
POST	Flow
HEAD	Like GET
PUT	Flow or Reject
DELETE	Flow or Reject
TRACE	Echoes request as per RFC
OPTION	Reject

Maximum Allowed Connections

The nHTTP listener has a parameter that can be configured to limit the number of simultaneous connections. However, this is not related to pool sizes or persistent connections. This parameter simply limits the number of clients that can connect to the server at once, persistent or not.

The listener tracks the current number of connections. When a new connection is accepted, the count is raised. When a connection closes, the count is lowered. Before accepting a new connection, the listener checks the current number of connections against the max connections parameter. If the current number is at the threshold, the new connection is not accepted and the following error message is written to the log:

```
ERROR (nh2) max connection threshold exceeded
```

Please note that the client has no knowledge of this back-end functionality. From its point of view, the connection might just be slow. The client will continue making connection attempts until it times out. As a result, it is normal to see multiple instances of this error message when simultaneous connections are over the limit. If the client does not time out and another connection closes, the new connection will be accepted and normal processing is continued

A blank value or 0 specified for the parameter indicates no limit.

SSL Host Verification

When SSL Host Verification is enabled, the client verifies that the certificate the server is presenting in the handshake matches the server hostname.

So, in the keystore of the server SSL context, there needs to be a key pair with CN == server hostname. If there is more than one private key in this keystore, you need to specify the server key alias to point to this key.

The client needs to add the <certificate of the CA that signed the server certificate> to its truststore. In the case of a self-signed certificate, this is the server certificate itself. The server never verifies the client hostname, even if SSL client authentication is enabled.

The following shows some of the information in a self-signed certificate with the CN in the subject Distinguished Name set to the host and port as required by the host name verifier.

```
Owner CN=myMachine.ibi.com:7777, OU=iWay, O=IBI, L=Cranston, ST=Rhode
Island, C=US
Issuer CN=myMachine.ibi.com:7777, OU=iWay, O=IBI, L=Cranston, ST=Rhode
Island, C=US
Serial number 46141cb7
Valid from Wed Apr 04 17:46:31 EDT 2007 until: Mon Oct 01 17:46:31 EDT
2007
Certificate fingerprints
MD5 61:02:2E:F2:D6:C2:0B:A8:AF:1F:6F:86:64:23:C9:17
SHA1 5F:7B:6C:A5:0E:FC:0C:33:F6:4C:4D:48:1B:C9:07:A4:DD:EF:54:62
```

SOAP

SOAP (Simple Object Access Protocol) is a lightweight protocol for exchanging information in a decentralized, distributed environment. SOAP is an XML-based protocol that consists of three parts. An envelope that defines the framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses.

SOAP codifies the existing practice of using XML and HTTP as a method invocation mechanism and uses an XML vocabulary for representing method properties, return values, and exceptions.

To receive messages from SOAP, you must configure a SOAP listener. The iWay protocol adapter base configuration includes a preconfigured SOAP listener.

Reference: SOAP Listener Properties

The following table lists and describes the SOAP listener properties. For instructions on creating a listener, see *Configuring Listeners* on page 18.

Property Name	Property Description
Port (required)	The port on which SOAP requests are accepted. 9000 is the iWay default SOAP port.
Local bind address	The local bind address for multi-homed hosts. This value is usually left empty.
Asynch Directory	The stores the responses (output) from previously issued asynchronous SOAP requests. Also stores event messages destined for remote systems.
Timeout	The timeout interval for the TCP socket.

Property Name	Property Description
LingerTime	The linger-on-close period in seconds. Use if TCP loses links when sending documents.
Keystore	The path to the keystore, which contains the key.
Keystore Password	The password required to recover private keys from the keystore.
Keystore Type	The keystore file type. JKS is the default value.
Truststore	The file that provides the trust certificates for authenticating clients. Use if client authentication is required. Leave blank to use the default JVM truststore.
Truststore Type	The type of truststore.
Security Provider Class	The name of the class that implements some or all parts of Java security, including algorithms, key generation, conversion, and management facilities. The default Sun provider is: `com.sun.net.ssl.internal.ssl.Provider`
Security Protocol	The protocol used to enable security. Security protocol values include: ❑ **SSL.** Supports some version of SSL. May support other versions. ❑ **SSLv3.** Supports SSL version 3. May support other versions. ❑ **TLS.** Supports some version of TLS. May support other versions. ❑ **TLSv1.** Supports TLS version 1. May support other versions.
Security Algorithm	The algorithm that enables security. SunX509 is the default value.
Client Authentication	If set to *true*, the client is required to authenticate itself. The client must have a keystore of its own.
Whitespace Normalization	This specifies how the parser treats whitespace in element objects. Select *preserve* (default) to turn off all normalization as prescribed by the XML Specification. Select *condense* to remove extra whitespaces in pretty printed documents and for compatibility with earlier versions.

Property Name	Property Description
Accepts non-XML (flat) only	If set to *true*, the input data is sent directly to the business logic step. The data is not preparsed, parsed, or validated. This flag is used primarily to send non-XML to the business logic or replyTo without processing it.
Optimize Favoring	Use this option to customize how the listener performs. For smaller transactions, select *performance*. For large input documents that could monopolize the amount of memory used by iWay Service Manager, select *memory*.
Multithreading	This indicates the number of worker threads (documents or requests) that iWay Service Manager can handle in parallel. Setting this to a value of greater than 1 enables the listener to handle a second request while an earlier request is still being processed. The total throughput of a system can be affected by the number of threads operating. Increasing the number of parallel operations may not necessarily improve throughput. Default: 1 Max Value: 99
Maximum threads	The parallel threads can grow to this count automatically on demand. Over time, the worker count will decrease back to the multithreading level. Use this parameter to respond to bursts of activity.
Execution Time Limit	The maximum time that a request may take to complete. Used to prevent runaway requests. Any request that takes longer to complete than this value will be attempted to be terminated. (See the kill interval for related information in Global Options.)

Property Name	Property Description
Agent Precedence	This sets the order by which iWay Service Manager selects agents. iWay Service Manager selects the agent or agents to process the document by searching through the configuration dictionary. Usually, it looks for a document entry in the configuration and when a match is found, the agent specified in that document entry is selected. If a matching document entry is not found, or no agent is specified, the engine looks in the input protocol configuration (listener). To have the processing agent taken directly from the listener (thus ignoring the document entry), use <listener> overrides <document>. Possible values are <document> overrides <listener> and <listener> overrides <document>. The default value is: <document> overrides <listener>
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined reply-to and error-to destinations.
Error Documents treated normally	If set *true*, the error documents are processed by any configured preemitters.
Listener is Transaction Manager	If set to *true*, the agents run in a local transaction. Agents can roll back uncompleted transactions.
Record in Activity Log(s)	If set to *true*, then the activity on this channel will be recorded in the activity logs. If set to *false*, the activity will not be recorded.

Reference: SOAP Listener Special Registers

The following table lists and describes the special registers (SREGs) available on the SOAP listener.

Name	Level	Type	Description
ip	Document	String	The IP address of the sender.
iwayconfig	System	String	The current active configuration name.

Name	Level	Type	Description
msgsize	Document	Integer	The physical length of the message payload.
name	System	String	The assigned name of the master (listener).
protocol	System	String	The protocol on which the message was received.
source	Document	String	The originator of the message.
tid	Document	String	Unique transaction ID.

TCP

Today, the Internet and World Wide Web (WWW) are familiar terms to millions of people all over the world. Many people depend on applications enabled by the Internet, such as electronic mail and web access. In addition, the increased popularity of business applications places additional emphasis on the Internet.

The Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite is the engine for the Internet and networks worldwide. Its simplicity and power makes it the single network protocol of choice in the world today.

This topic provides an overview of the TCP/IP protocol suite. Another name for it is the Internet Protocol Suite, which is the phrase used in the official Internet standards documents. iWay Software uses the more common acronym, TCP/IP, to refer to the entire protocol suite.

The main goal of TCP/IP was to build an interconnection of networks, referred to as an internetwork, or internet, that provided universal communication services over the internet.

The iWay Adapter for TCP provides bidirectional capability to and from TCP/IP destinations. It enables the iWay Protocol Adapter to receive messages on a TCP/IP port and route those messages back to a client or to another non-TCP destination. Similarly, messages received through the other iWay Service Manager listeners can be directed to a TCP/IP port. During the message flow, iWay Service Manager can apply standard document analysis, validation, transformation, and business logic capabilities to the document.

Reference: TCP Listener Properties

The following table lists and describes the TCP listener properties. For instructions on creating a listener, see *Configuring Listeners* on page 18.

Property Name	Property Description
Port (required)	Socket number on which iWay Service Manager listens for incoming messages from a client application. The iWay Service Manager configuration tool requires that the port be numeric.
Local bind address	Local bind address for multi-homed hosts. Usually leave this value empty.
Allowable Client	If set, accepts messages only from this HOSTNAME or IP address.
Timeout	Timeout interval for the TCP socket.
Stream Length Encoded	The protocol needs to be able to recognize a complete message. Several methods are provided: ❏ **None (XML encoded).** Looks for the end of the XML document. ❏ **4 byte MSB.** An integer precedes the message. ❏ **2 byte LSB.** An integer in LSB form precedes the message. This is often used for messages from mainframes. ❏ **6 byte ASCII.** Character encoded length precedes the message. ❏ **Non-XML.** Complete message to socket close. ❏ **Read characters to End Of Line (EOL).** Reads one line at a time until en EOL is encountered. For example: `DATA1/r/nDATA2/r/nDATA3/r/n...` where: */r/n* Is the data separator.
Persistent connection	If set to *true*, the connection is maintained until the client closes. **Note:** If you use this option, length encoding must be enabled.
Set TCP No Delay	If set to *true*, Nagle's Algorithm is disabled on the client socket. This will result in faster line turnaround at the expense of an increased number of packets.

Property Name	Property Description
Defer Close of Socket	If set to *true*, closing the client socket is deferred for one second after the response is written. This compensates for an issue seen on some older versions of z/OS.
Tuning	
Multithreading	This indicates the number of worker threads (documents or requests) that iWay Service Manager can handle in parallel. Setting this to a value of greater than 1 enables the listener to handle a second request while an earlier request is still being processed. The total throughput of a system can be affected by the number of threads operating. Increasing the number of parallel operations may not necessarily improve throughput. Default: 1 Max Value: 99
Maximum Threads	Parallel threads can grow to this count automatically on demand. Over time, the worker count will decrease back to the multithreading level. Use this parameter to respond to bursts of activity.
Optimize Favoring	Use this option to customize how the listener performs. For smaller transactions, select *performance*. For large input documents that could monopolize the amount of memory used by iWay Service Manager, select *memory.*
Events	
Failed ReplyTo Flow	Specify the name of a published process flow to run if a message cannot be emitted on an address in its reply address list.
Dead Letter Flow	Specify the name of a published process flow to run if an error cannot be emitted on an address in its error address list.
Channel Failure Flow	Specify the name of a published process flow to run if this channel cannot start or fails during message use. iSM will attempt to call this process flow during channel shutdown due to the error.
Parse Failure Flow	Specify the name of a published process flow to run if XML parsing fails for the incoming message.

Property Name	Property Description
Channel Startup Flow	Specify the name of a published process flow to run prior to starting the channel.
Channel Shutdown Flow	Specify the name of a published process flow to run when the channel is shut down.
Other	
Whitespace Normalization	This specifies how the parser treats whitespace in element objects. Select *preserve* (default) to turn off all normalization as prescribed by the XML Specification. Select *condense* to remove extra whitespaces in pretty printed documents and for compatibility with earlier versions.
Accepts non-XML (flat) only	If set to *true*, the input data is sent directly to the business logic step. The data is not preparsed, parsed, or validated. This flag is used primarily to send non-XML to the business logic or replyTo without processing it.
Execution Time Limit	The maximum time that a request may take to complete. Used to prevent runaway requests. Any request that takes longer to complete than this value will be attempted to be terminated. (See the kill interval for related information in Global Options.)
Default Java File Encoding	The default encoding if an incoming message is not self-declaring (that is, XML).
Agent Precedence	This sets the order by which iWay Service Manager selects agents. iWay Service Manager selects the agent or agents to process the document by searching through the configuration dictionary. Usually, it looks for a document entry in the configuration and when a match is found, the agent specified in that document entry is selected. If a matching document entry is not found, or no agent is specified, the engine looks in the input protocol configuration (listener). To have the processing agent taken directly from the listener (thus ignoring the document entry), use <listener> overrides <document>. Possible values are <document> overrides <listener> and <listener> overrides <document>. The default value is: <document> overrides <listener>

Property Name	Property Description
Always reply to listener default	If set to *true*, the default reply definition is used in addition to define reply-to and error-to destinations.
Error Documents treated normally	If set to *true*, the error documents are processed by any configured preemitters.
Listener is Transaction Manager	If set to *true*, the agents run in a local transaction. Agents can roll back uncompleted transactions.
Record in Activity Log(s)	If set to *true*, then the activity on this channel will be recorded in the activity logs. If set to *false*, else the activity will not be recorded.
AES Key	If the channel is expecting to receive encrypted AFTI messages, specify the AES key (maximum 16 characters) to be used for decryption purposes.
Startup Dependencies	Specify a comma-separated list of channel names that must be started before the current channel is started.

Reference: TCP Listener Special Registers

The following table lists and describes the special registers (SREGs) available on the TCP listener.

Name	Level	Type	Description
ip	Document	String	The IP address of the sender.
iwayconfig	System	String	The current active configuration name.
msgsize	Document	Integer	The physical length of the message payload.
name	System	String	The assigned name of the master (listener).
protocol	System	String	The protocol on which message was received.
source	Document	String	The originator of the message. Same as IP.

Name	Level	Type	Description
tid	Document	String	Unique transaction ID.

Configuring a TCP Emitter

Messages are sent to particular destinations at the completion of a workflow. The state of the document determines which destination is used. The order in which they are used cannot be predicted.

Note: Configuring a TCP emitter is not required if the emitter protocol is the same as the listener protocol.

To route an output document or error message to a protocol other than that of the outlet (listener), you must configure an emitter. For example, an application may send input over FTP but want to route the output to a TCP listener.

TCP Emitter Properties

The following table lists and describes the TCP emitter properties. For instructions on creating an emitter, see *Configuring Emitters* on page 29.

Property Name	Property Description
Destination (required)	The TCP host and port.
Stream Length Encoded	A form of length encoding for the TCP stream. Values include: ❑ **None (XML encoded).** Looks for the end of the XML document. ❑ **4 byte MSB.** An integer precedes the message. ❑ **2 byte LSB.** An integer in LSB form precedes the message. This is often used for messages from mainframes. ❑ **6 byte ASCII.** Character encoded length precedes the message. ❑ **Non-XML.** Complete message to socket close.

Property Name	Property Description
Set TCP No Delay	If set to *true*, then this parameter disables the Nagle Algorithm on the client socket. This results in a faster line turnaround at the expense of an increased number of packets.

UDP

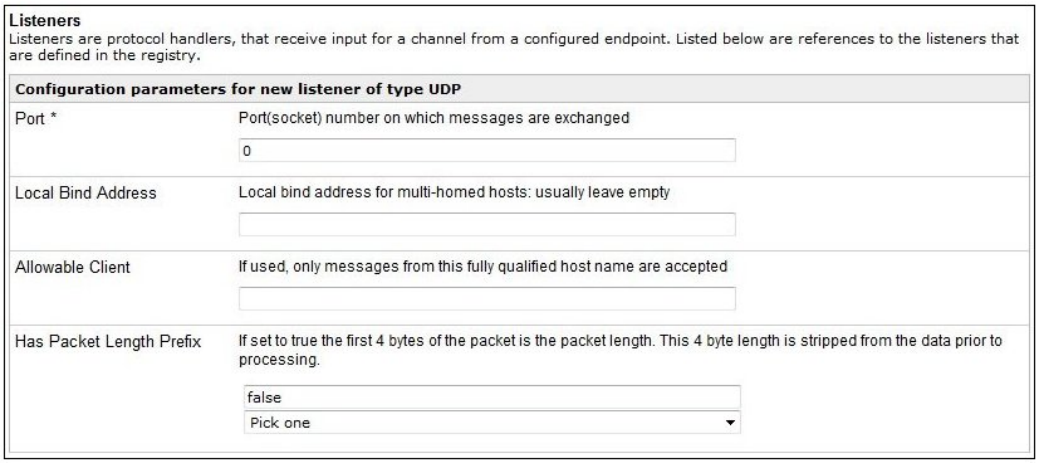
The User Datagram Protocol (UDP) is a transport layer protocol defined for use with the Internet Protocol (IP) network layer protocol. UDP is one of the core members of the IP suite. UDP uses a simple connectionless transmission model with a minimum of protocol mechanism. Unlike TCP/IP, UDP/IP provides very few error recovery services, offering instead a direct way to send and receive messages over an IP network. It is used primarily for broadcasting messages over a network.

UDP performs no handshaking between the sending and receiving parties and is therefore said to be connectionless. This protocol does not maintain state. As a result, it is fast and light-weight in terms of system resource usage. Lack of error handling and packet organization (packets can arrive in any order) make UDP useful for implementing very fast, lightweight services, such as lookups. In fact, the DNS service of the Internet is implemented using UDP.

You can listen for incoming UDP message packets by using the UDP listener, and can emit UDP message packets through the UDP Emitter or the UDP Emit service (com.ibi.agents.XDUDPEmitAgent).

Configuring the UDP Listener

You can listen for inbound UDP requests by configuring the UDP listener.



The following table lists and describes the configuration parameters for the UDP listener.

Parameter	Description
Port *	The active port on which the UDP message packet will be received.
Local Bind Address	This parameter reserved for instances of iSM that are installed on machines where multiple network adapters are installed. This parameter allows you to configure the UDP listener to use a different IP address than the default address of the system.

Parameter	Description
Allowable Client	This parameter allows you to restrict the source of the UDP message to a specific host address. This field is a comma-separated listing of host names and IP address filters that will be applied to the source address of the UDP packet. If the packet source does not match one of the filters, then the packet is rejected. The filters support wild card characters, for example, an asterisk (‘*’), any grouping of one or more characters, and any single character (‘?’). **Sample Filters:** ❏ `*.ibi.com` Will accept any host name within the ibi.com domain. ❏ `172.16.*` Will accept any IP address in the range of 172.16.0.0 through 172.16.255.255. ❏ `ab?c.ibi.com` Will accept a host whose name begins with the string "ab" followed by any single alpha-numeric character followed by the string "c.ibi.com". **Note:** Using a host name string pattern will force a DNS lookup on the sender name of the UDP message packet. This may be time consuming if your DNS cache is not up to date. However, using an IP Address string pattern does not require a DNS lookup and should be faster.

Parameter	Description
Has Packet Length Prefix	Determines whether to prepend the data packet length. Select one of the following values from the drop-down list. ❏ **true.** The UDP message packet contains a four byte length prefix prior to the start of the message data. This length prefix is a binary encoded big endian integer (for example, hex 00000C0 = decimal length of 192). ❏ **false.** The UDP message packet does not contain a four byte length prefix. By default, *false* is selected.
Tuning	
Multithreading	Indicates the number of worker threads (documents or requests) that iWay Service Manager can handle in parallel. Setting this to a value of greater than 1 enables the listener to handle a second request while an earlier request is still being processed. The total throughput of a system can be affected by the number of threads operating. Increasing the number of parallel operations may not necessarily improve throughput. The default is 1. The max value is 99.
Optimize Favoring	Use this option to customize how the listener performs. For smaller transactions, select *performance*. For large input documents that could monopolize the amount of memory used by iWay Service Manager, select *memory*.
Polling Interval	The maximum wait interval (in seconds) between checks for new requests or commands. The higher this value, the longer the interval, and the fewer system resources that are used. The side effect of a high value is that the worker thread will not be able to respond to a stop command. The default is 2.0 seconds.

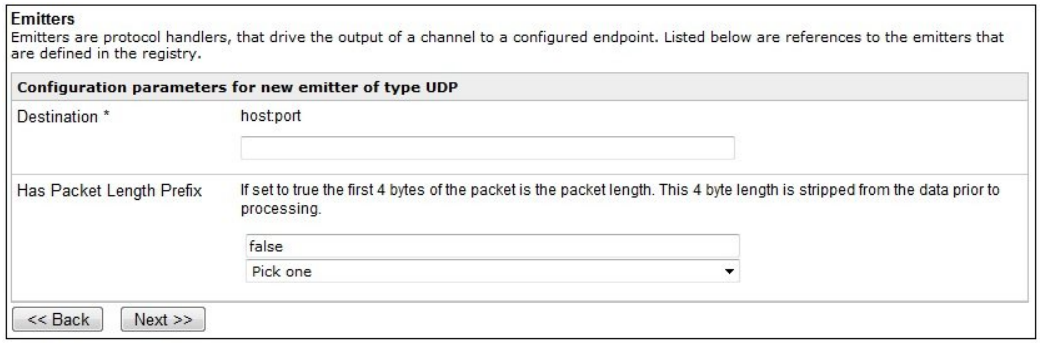
Parameter	Description
Events	
Expired Retry Flow	Name of a published process flow to run if a message on the retry queue has expired.
Failed ReplyTo Flow	Name of a published process flow to run if a message cannot be emitted on an address in its reply address list.
Dead Letter Flow	Name of a published process flow to run if an error cannot be emitted on an address in its error address list.
Channel Failure Flow	Name of a published process flow to run if this channel cannot start or fails during message handling. iWay Service Manager will attempt to call this process flow during channel shut down due to the error.
Parse Failure Flow	Name of published process flow to run if XML parsing fails for incoming message.
Channel Startup Flow	Name of published process flow to run prior to starting the channel.
Channel Shutdown Flow	Name of published process flow to run when the channel is shut down.
Other	
Whitespace Normalization	Specifies how the parser treats whitespace in Element content. Choose *preserve* to turn off all normalization as prescribed by the XML Specification. Choose *condense* to remove extra whitespace in pretty printed documents and for compatibility with earlier versions.
Accepts non-XML (flat) only	If set to *true*, the input data is sent directly to the business logic step. The data is not preparsed, parsed, or validated. This flag is used primarily to send non-XML to the business logic or replyTo without processing it.

Parameter	Description
Execution Time Limit	The maximum time that a request may take to complete. Used to prevent runaway requests. Any request that takes longer to complete than this value will be attempted to be terminated.
Default Java File Encoding	The default encoding if incoming message is not self-declaring (that is, XML).
Agent Precedence	Sets the order by which iWay Service Manager selects agents. iWay Service Manager selects the agent or agents to process the document by searching through the configuration dictionary. Usually, it looks for a document entry in the configuration and when a match is found, the agent specified in that document entry is selected. If a matching document entry is not found, or no agent is specified, the engine looks in the input protocol configuration (listener). To have the processing agent taken directly from the listener (thus ignoring the document entry), use <listener> overrides <document>. Possible values are <document> overrides <listener> and <listener> overrides <document>. The default value is <document> overrides <listener>.
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined reply-to and error-to destinations.
Error Documents treated normally	If set to *true*, error documents are processed by any configured preemitters.
Listener is Transaction Manager	If set to *true*, agents run within a local transaction.
Record in Activity Log(s)	If set to *true*, activity on this channel will be recorded in the activity logs, otherwise the activity will not be recorded.

Parameter	Description
AES Key	If the channel will receive encrypted AFTI messages, set the AES key (maximum 16 characters) to be used for decrypting.
Startup Dependencies	A comma-separated list of channel names that must be started before this one is called.

Configuring the UDP Emitter

You can emit UDP replies or requests by configuring the UDP emitter.



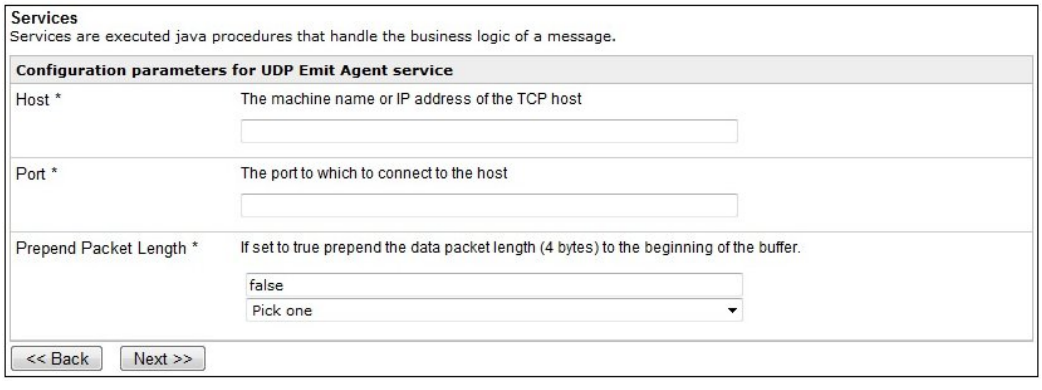
The following table lists and describes the configuration parameters for the UDP emitter.

Parameter	Description
Destination *	The host and port destination where the UDP message packet will be transmitted. The entry is in the form of *host:port*. For example, *reporter:1234* to route messages to the reporter machine on port 1234.

Parameter	Description
Has Packet Length Prefix	Determines whether to prepend the data packet length. Select one of the following values from the drop-down list. ❏ **true.** The UDP message packet will contain a length prefix. The emitter calculates the binary length of the message. This length is exclusive of the four byte packet length. The length is then prepended to the packet. ❏ **false.** The UDP message packet will not contain a length prefix. By default, *false* is selected.

Configuring the UDP Emit Service

The UDP Emit service allows you to configure a service to transmit a User Datagram Protocol (UDP) message packet to a specified host.



Syntax:

```
com.ibi.agents.XDUDPEmitAgent
```

Parameters:

Parameter	Description
Host	The host name or IP address of the machine to which the UDP message packet will be sent.
Port	The port number of the host to which the UDP message packet will be sent.
Prepend Packet Length	Determines whether to prepend the data packet length. Select one of the following values from the drop-down list. ❑ **true.** The UDP message packet will contain a length prefix. The emitter calculates the binary length of the message. This length is exclusive of the four byte packet length. The length is then prepended to the packet. ❑ **false.** The UDP message packet will not contain a length prefix. By default, *false* is selected.

Note: Due to the nature of UDP, the only error checking that is performed is whether the host machine can be reached. Message packets are sent to the host with no guarantee that the packet was received or was processed by the specified host system.

iWay Command Extension

The iWay Telnet extension is used to access the iWay Service Manager (iSM) command line console through a remote session. A command console client session can connect to any iSM instance running in the foreground or background. The command session can be created or activated as required through the iSM Administration Console.

For more information on configuring and using the Telnet listener, see the *iWay Service Manager Command Reference Guide*.

iWay RVI Proxy Extension

The iWay RVI Proxy (also called RVI Gateway) extension links two or more iWay Service Manager instances in a message receiver or a message executor relationship to tunnel through secure firewalls.

To install the iWay RVI Proxy extension, you must:

1. Install the iWay Gateway extension on the iWay Proxy server and the execution engine.
2. Configure the RVIAttach listener on the iWay Proxy server.
3. Add the Relay agent to the appropriate listener(s) configured on the iWay Proxy server.
4. Configure the RVIGateway listener on the execution engine.

For more information on configuring and using the RVIAttach and RVIGateway listeners, see the *iWay Service Manager Extensions User's Guide*.

Chapter 6

Queuing Protocol Adapters

The iWay queuing protocol adapters employ the Java Message Service Queue (JMSQ) application programming interface (API) as the underlying technology for inter-client communication between distributed applications. JMSQ provides a common interface that wraps around the underlying message delivery systems of a number of vendors.

In this chapter:

Introducing Queuing Protocol Adapters

Message queuing is a method by which a process can exchange or pass data using an interface to a system-managed message queue. Messages can vary in length and can be assigned different types or uses.

A message queue can be created by one process and used by multiple processes that read and/or write messages to the queue. For example, a server process can read messages from and write messages to a queue created for client processes. The message type can be used to associate a message with a particular client process although all messages are in the same queue.

An architecture for distributed systems is based on reliable message queuing. Messages are queued asynchronously between applications and systems.

The benefits of a message queuing system are:

- Multiple providers can post messages to a queue.
- Multiple message consumers can be attached to a single queue.
- The queuing infrastructure ensures messages are delivered only once.
- Messages can be submitted to a queue even when the message consumer(s) are not running or are unreachable.
- Multiple posters/readers ensure scalability.
- Systems that can make queues persistent provide reliability.
- Because producers and consumers are not interconnected, abstraction is provided (similar to a loosely coupled pipes and filters architecture).

JMSQ

Enterprise messaging is recognized as an essential tool for building enterprise applications and e-commerce systems. Java Message Service Queue (JMSQ) provides a common way for Java programs to create, send, receive, and read enterprise messaging system messages. The JMSQ application programming interface (API) is a Java technology API for inter-client communication among distributed applications.

JMSQ provides a common interface that wraps around the underlying message delivery system. The delivery system can be provided by any number of vendors. JMSQ is a service-oriented API specification, that is, the JMSQ API prescribes messaging functionality in terms of interfaces, which JMSQ vendors then implement. Therefore, programmers work with JMSQ through these interfaces.

JMSQ is a set of interfaces and associated semantics that define how a JMSQ client accesses the facilities of an enterprise-messaging product. Rather than communicate directly with each other, the components in an application, based around a message service, send messages to a message server. The message server, in turn, delivers the messages to the specified recipients.

iWay Service Manager can read messages arriving on a JMS queue and route the response to either another queue or to a non-JMSQ destination. Similarly, Service Manager can direct messages received through its listeners to a JMS queue. During the message flow, Service Manager can apply the standard document analysis, validation, transformation, and business logic capabilities to the document.

The iWay Adapter for JMS is a generic listener/emitter that provides an interface to various message-oriented middleware products that adopted the JMSQ standard. Each product must supply the required interface libraries that must be registered in the Service Manager configuration. Therefore, the properties listed differ depending on the JMSQ vendor, and you must add the required JMS JAR files for each to the class path.

Registering JMS JAR Files

JMSQ architecture is closely allied with a web server and JNDI (Java Naming Directory Interface), which locates the drivers. Therefore, the JMSQ listener requires that you register the specific JAR files supplied by the JMSQ vendor.

The JMSQ listener requires that you register:

Server/Software	JAR Files
WebLogic Server	weblogic.jar
TIBCO Enterprise for JMSQ software	❑ tibjms.jar ❑ tibjmsnaming.jar ❑ jms.jar ❑ tibjmsadmin.jar
OpenJMS	❑ concurrent-1.3.4.jar ❑ openjms-0.7.7-beta-1.jar ❑ openjms-common-0.7.7-beta-1.jar ❑ openjms-net-0.7.7-beta-1.jar ❑ spice-jndikit-1.2.jar

For instructions on registering JAR files, see *Registering Library Files and Setting JVM Options* on page 40.

Reference: JMSQ Listener Properties

The following table lists and describes the JMSQ listener properties. For instructions on creating a listener, see *How to Create a Listener* on page 18.

Property Name	Property Description
Connection (required)	The resource that contains information about the JMS Server.
JNDI Factory	The name of the JNDI factory. Provided by the JNDI service provider.
JNDI URL	The path to the JNDI server for the initial JNDI context handle to use.
Form of Input	The form of input. **TOPIC:** Used for a TopicSession in the Publish and Subscribe domain. **QUEUE:**U sed for a QueueSession in the PTP domain.
Form of Acknowledgment	The acknowledgment mode support. The session acknowledgment mode is either transactional (to send and receive a series of messages and then explicitly commit or rollback the group) or one of the following acknowledgment modes. **Client Provides:** An explicit acknowledge on a message acknowledges the receipt of all messages produced and consumed by the session that gives the acknowledgment. When a session is forced to recover, it restarts with its first unacknowledged message. **Duplicates Permitted:** The session lazily acknowledges the delivery of messages to consumers, possibly allowing duplicate messages after a system outage. **Auto Acknowledge:** The session automatically acknowledges the client receipt of a message. By successfully returning from a call to receive (synchronous mode) or when the session message listener successfully returns (asynchronous mode), the last message can be delivered again.
External Transaction Rollback	Use external rollback facilities to manage transactions, for example, the BEA WebLogic pending facilities.
Receiver Name (required)	The name of the queue on which input documents will be received for processing by Service Manager.

Property Name	Property Description
Set Reply Correlation Id	The entered value is used as the correlation ID of the response.
Default Reply	The default queue or topic to which the response document will be written.
Default Output Form	You can select a topic or queue target.
Pending Queue	The name of the queue where Service Manager keeps failed requests that are retried.
User	A valid user ID defined to the JMS Server.
Password	A valid password defined to the JMS Server.
Duration	The maximum time, in milliseconds, that a document can remain in the retry pending queue.
Retry	The interval between retrying pending requests contained in a message.
Message Selector	In PTP domains, message selection occurs on the server. However, in Pub/Sub domains, by default, messages for a subscribed topic are delivered to the subscriber, and then, the filter is applied to determine what is consumed. Message selectors can consist of: ❏ Literals and indefinites ❏ Operators and expressions ❏ Comparison tests ❏ Parentheses ❏ White space

Property Name	Property Description
Message Priority	A priority is suggested to the JMSQ provider and can only order the delivery of undelivered messages. JMSQ defines ten levels of message priority with values 0 through 9, where 0 is the lowest and 9 is the highest. Zero through four are considered normal settings, and five through nine, expedited. The Message Priority field is the default priority value set in the JMSQ header.
Output Message Type	Defines the type of the output message (bytes, dynamic, text, map, object, or stream).
JMSReplyTo	The queue or topic (used for JNDI lookup).
Get CorrelId As Bytes	Accounts for binary in correlation IDs.
Request Header Namespace	The special register namespace into which protocol headers from the incoming request will be saved.
Response Header Namespace	The special register namespace from which protocol headers for the outbound response will be taken.
Whitespace Normalization	Specifies how the parser treats whitespace in Element content. Select *preserve* to turn off all normalization as prescribed by the XML Specification. Select *condense* to remove extra whitespace in pretty printed documents and for compatibility with earlier versions.
Accepts non-XML (flat) only	If set to *true*, then non-XML input is expected. If enabled, XML input still can be passed to the listener. Preparsers do not run.
Optimize Favoring	Use this option to customize how the listener performs. For smaller transactions, select *performance*. For large payloads that could monopolize the amount of memory used by Service Manager, select *memory*.

Property Name	Property Description
Multithreading	Indicates the number of worker threads. (Equivalent to the number of requests that Service Manager can handle in parallel.) Setting this to greater than 1 enables the listener to handle a second request while an earlier request is still being processed. The total throughput of a system can depend on the number of threads operating. Default: 1 Max value: 99
Execution Time Limit	The maximum time a request can take to complete. A request that takes longer to complete terminates. Prevents runaway requests.
Polling Interval	The interval (in seconds) at which the listener checks for new input. The listener is constantly connected to the queue to retrieve incoming messages. The default value is 2.0.
Default Java File Encoding	The default encoding if an incoming message is not self-declaring (that is, XML).
Agent Precedence	Sets the order by which Service Manager selects agents. Service Manager looks for a document entry in the configuration library and when a match is found, the agent specified in that document entry is selected. If no matching document entry is found or agent specified, the engine looks in the input protocol configuration (listener). To have the processing agent taken directly from the listener (thus ignoring the document entry), use <listener> overrides <document>. <document> overrides <listener> is the default value.
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined destinations.
Error Documents treated normally	If set to *true*, error documents are processed by configured preemitters.
Listener is Transaction Manager	If set to *true*, the agents run in a local transaction. Agents can roll back uncompleted transactions.

Property Name	Property Description
Record in Activity Log(s)	If set to *true*, activity on this channel will be recorded in the activity logs. If set to *false*, the activity will not be recorded.

Note: The JMSQ listener supports streaming. Streaming is used for large documents or documents for which the application needs to split the input into sections under the same transaction. For more information on streaming and configuring streaming preparsers, see the *iWay Service Manager Component and Functional Language Reference Guide*.

Reference: JMSQ Listener Special Registers

The following table lists and describes the special registers (SREGs) available on the JMSQ listener.

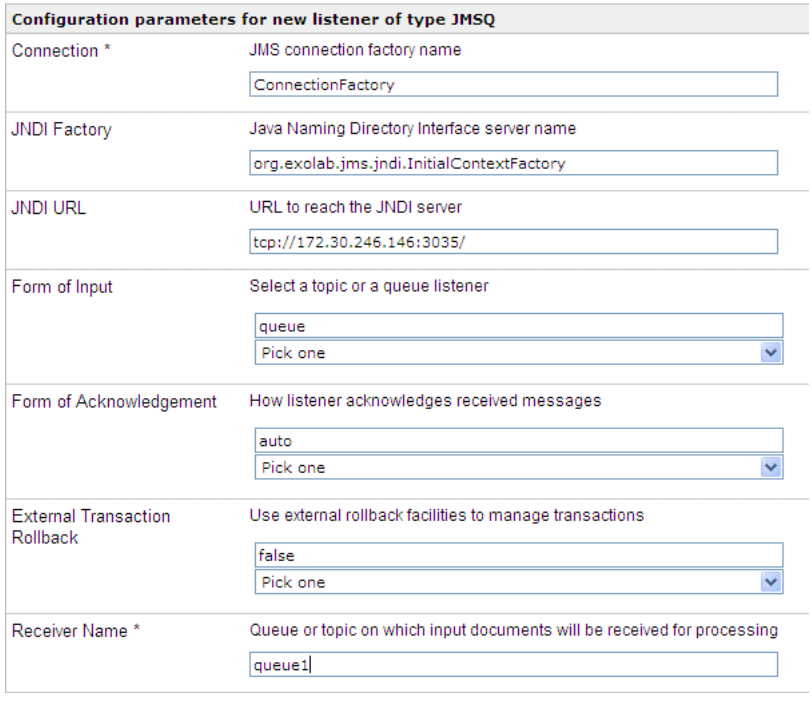
Name	Source	Level	Type	Description
correlid	Listener	Document	String	The correlation ID.
destination	Listener	Document	String	The default destination for reply (from message).
iwayconfig	Listener	System	String	The current active configuration name.
msgid	Listener	Document	String	The message ID.
msgsize	Listener	Document	Integer	The physical length of the message payload.
name	Listener	System	String	The assigned name of the master (listener).
priority	Listener	Document	String	The priority of this message.
protocol	Listener	System	String	The protocol on which this message was received.
source	Listener	Document	String	The queue or topic on which message was received.
tid	Listener	Document	String	Unique transaction ID.

JMSQ Listener Configuration Example

You are receiving documents on a JMS queue, named queue1, located at the following host: tcp://172.30.246.146:3035. You would like to configure a listener on this queue. Specify the values as follows:

- **Connection:** ConnectionFactory
- **JNDI Factory:** org.exolab.jms.jndi.InitialContextFactory
- **JNDI URL:** tcp://172.30.246.146:3035/
- **Receiver Name:** queue1
- **User logon ID/password:** Supply the correct logon credentials.

Note: For more information on configuration parameters, see the OpenJMS documentation at *http://openjms.sourceforge.net*.



Configuring a JMS Emitter

Messages are sent to the destinations at the completion of a workflow. The state of the document determines which destination is used. The order in which the destinations are used cannot be predicted. Destinations cannot accept responses. These configurations also are used to send a message over a different protocol.

Note: Configuring a JMS emitter is not required if the emitter protocol used in the outlet of the channel is the same as the listener protocol used in the inlet of the channel. For more information on inlets and outlets, see *Defining an Inlet* on page 15 and *Defining an Outlet* on page 25.

To route an output document or error message to a protocol other than that of the listener, you must configure an emitter. For example, an application may send input over TCP but want to route the output to a JMS queue.

JMS Emitter Properties

The following table lists and describes the JMS emitter properties. For instructions on creating an emitter, see *How to Create an Emitter* on page 29.

Property Name	Property Description
Destination (required)	Queue@connectionFactory. A connection factory is a resource that contains information about the JMS Server.
JNDI URL	The path to the JNDI Server to find the initial context handle to use JNDI.
JNDI Factory	InitialContext is a bound handle to a directory server, which looks up administered resources. The JNDI factory is specific to the implementation of the JNDI provider.

Form of Output	`TOPIC`
	Is for a TopicSession in the Publish and Subscribe domain.
	`QUEUE`
	Is for a QueueSession in the PTP domain.

Set Reply Correlation Id	If set to a value, this is used as the correlation ID of the response.

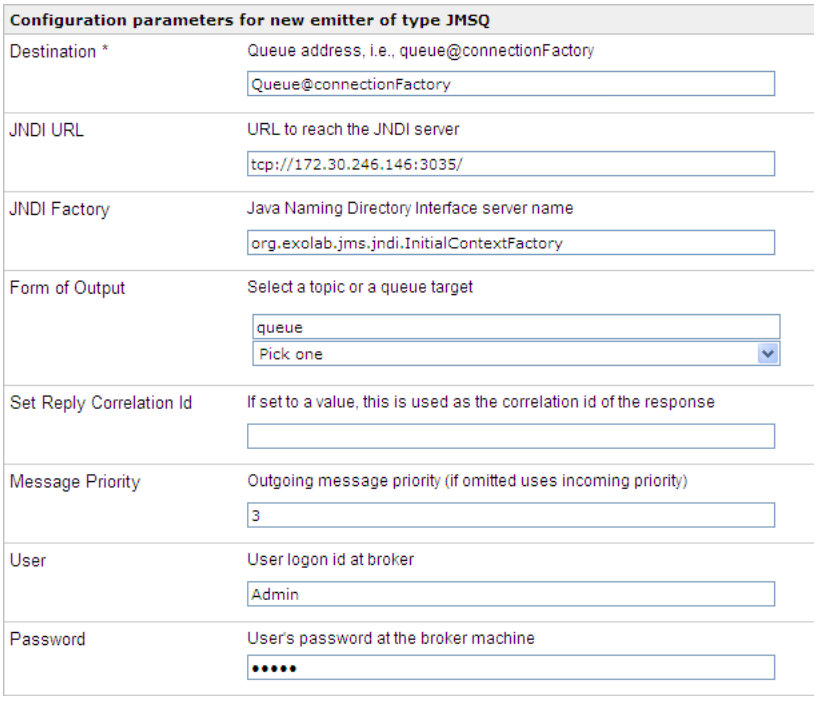
Property Name	Property Description
Message Priority	A priority is suggested to the JMSQ provider and can only order the delivery of undelivered messages. JMSQ defines ten levels of message priority with values 0 through 9, where 0 is the lowest and 9 is the highest. Zero through four are considered normal settings and five through nine, expedited. The Message Priority field is the default priority value set in the JMSQ header.
User	A valid user ID defined to the JMS Server.
Password	A valid password defined to the JMS Server.
Output Message Type	Select *Bytes*, *Text*, or *Dynamic*.
JMSReplyTo	The queue or Topic (used for JNDI lookup).
Request Header Namespace	The special register namespace from which protocol headers for the outbound request will be taken.

JMS Emitter Configuration Example

You want to route documents or error messages to a protocol other than a listener. You must configure an emitter to do this. Specify the values as follows:

- **Destination:** queue@ConnectionFactory
- **JNDI Factory:** org.exolab.jms.jndi.InitialContextFactory
- **JNDI URL:** tcp://172.30.246.146:3035/
- **User logon ID/password:** Supply the correct logon credentials.

Note: For more information on configuration parameters, see the OpenJMS documentation at http://openjms.sourceforge.net.



Configuring a JMS Emitter for Server for JMS

To route an output document or error message to a protocol other than that of the listener, you must configure an emitter. For example, an application may send input over TCP, but want to route to a JMS queue.

Note: This procedure is not required if the emitter protocol used in the outlet of the channel is the same as the listener protocol used in the inlet of the channel. For more information on inlets and outlets, see *Defining an Inlet* on page 15 and *Defining an Outlet* on page 25.

JMS Emitter Properties for Server for JMS

The following table lists and describes the JMS emitter properties for Server for JMS. For instructions on creating an emitter, see *How to Create a Listener* on page 18.

Property Name	Property Description
Destination (required)	Queue@connectionFactory. A connection factory is a resource that contains information about the JMS Server. If you attempt to emit a message to a queue name, TestQueue, the value in this field would be TestQueue@QueueConnectionFactory.
JNDI URL	This property is required to access the iWay Server for JMSQ. The iWay Server for JMSQ can listen on a variety of protocols. The default protocol is TCP. If connecting to the iWay Server for JMSQ through TCP, the JNDI URL is tcp://host:3035.
JNDI Factory	The InitialContext is a bound handle to a directory server, which looks up administered resources. The JNDI factory is specific to the implementation of the JNDI provider. The JNDI factory for the iWay Server for JMSQ is: org.exolab.jms.jndi.mipc.IpcJndiInitialContextFactory.

Form of Output	`TOPIC`
	Is for a TopicSession in the Publish and Subscribe domain.
	`QUEUE`
	Is for a QueueSession in the PTP domain.

Set Reply Correlation Id	If a value is set, that value is used as the correlation ID of the response.
Message Priority	The outgoing message priority (if omitted, uses incoming priority).
User	The user ID at broker machine.
Password	The password associated with user ID.
Output Message Type	Bytes, dynamic, or text.
JMSReplyTo	The queue or topic (used for JNDI lookup).

JMSQ Troubleshooting

The following table lists and describes errors that you may encounter when using a JMSQ listener.

Error	Reason	Solution
`Could not start: XD[FAIL] Can't create JMSQ session.: [line 6]`	The class path was not updated to include the required JAR files for the JNDI Server. Alternatively, although these libraries were included using this pane, the server was not stopped and started again to enable this library to be included in the configuration.	Use the Path Settings pane to register the appropriate JMSQ JAR files in the ClassPath section. The server must be stopped and restarted for this to take effect.
`Cannot emit reply to .XDJMSQEmit <no dead letter path>: XD[FAIL} in emit{} error create queue.`	The system was unable to find the queue(s) specified.	Either the queue was not defined, or it was spelled incorrectly. The JMSQ listener is case-sensitive. Verify that the queue name exists and is entered in the configuration pane in the correct case.

MSMQ

Microsoft Message Queuing (MSMQ) is a messaging infrastructure and a development tool for creating distributed, loosely-coupled, messaging applications for the Windows operating system. Applications developed for Message Queuing send messages to queues. Queues are temporary storage locations used to ensure that messages reach their destination. These applications can communicate across heterogeneous networks and with computers that are offline.

Message Queuing provides guaranteed message delivery, efficient routing, security, transactional support, and priority-based messaging. Software that includes these features often is referred to in the industry as message-queuing software, store-and-forward software, or message-oriented middleware.

With Message Queuing, you can communicate across networks and with computers that are offline independent of the current state of the network and computers. System administrators can use Message Queuing to efficiently manage large, complex networks of computers and message queues.

Message Queuing provides applications with reliable communication and efficient use of network resources. Developers can focus on business logic instead of networking issues, because Message Queuing guarantees network communication.

The iWay Adapter for MSMQ provides bidirectional functionality and transactional support. The adapter facilitates the exchange of messages between Microsoft Message Queues and other enterprise systems.

The iWay Adapter for MSMQ enables Service Manager to read messages arriving on a named queue and route these messages to either another queue or to a non-MSMQ destination. Similarly, messages received through other Service Manager listeners can be directed to an MSMQ queue. During the message flow, the standard document analysis, validation, transformation, and business logic capabilities offered by the server can be applied to the document.

MSMQ Listener Properties

The following table lists and describes the MSMQ listener properties. For instructions on creating a listener, see *How to Create a Listener* on page 18.

Property	Property Description
Request Queue (required)	Queue on which request documents are received. The name is case-sensitive and conforms to the following format *Host\queuetype$\qName* where: *Host* Is the machine name where the Microsoft Queuing system is running. *queuetype* Private queues are queues that are not published in Active Directory and appear only on the local computer where they reside. Private queues are accessible only by Message Queuing applications that recognize the full path name or format name of the queue. *qName* Is the name of the queue where messages are placed, for example: `iwaykxc1\Private$\siebel`

Property	Property Description
Default Reply	Default queue where responses are directed unless specified otherwise, and where the message is routed. Conforms to the following format *Host**queuetype*$*qName* where: *Host* Is the machine name where the Microsoft Queuing system is running. *queuetype* Private queues are queues that are not published in Active Directory and appear only on the local computer where they reside. Private queues are accessible only by Message Queuing applications that recognize the full path name or format name of the queue. *qName* Name of the queue where messages are placed.
Filter by Msg Id	If values are entered, Service Manager accepts only messages with this message header value.
Filter by Correlation Id	If values are entered, Service Manager accepts only messages with this message header value.
Message Priority	Microsoft defines eight levels of message priority with values 0 through 7, where 0 is the lowest and 7 is the highest. Zero through three are considered normal settings, and four through seven, expedited. The Message Priority field is the default priority value set in the message header.
Transactional	If set to *true*, the queue will be treated as transactional.
Pending Queue	Name of the queue where Service Manager keeps requests that have not been processed (due to the back-end data server not being available).

Property	Property Description
Duration	The maximum time, in seconds, that a document can remain in the Retry pending queue.
Retry	The interval between retrying pending requests.
User Headers	The include user attributes in RFH2 header.
Whitespace Normalization	Specifies how the parser treats whitespace in Element content. Choose *preserve* to turn off all normalization as prescribed by the XML Specification. Choose *condense* to remove extra whitespace in pretty printed documents and for compatibility with earlier versions.
Accepts non-XML (flat) only	Select *true* if non-XML input is expected. If enabled, XML input still can be passed to the listener. Preparsers do not run.
Optimize Favoring	For small documents, select *performance*. For large volumes of data, select *memory*.
Multithreading	The number of worker threads. Equivalent to the number of document requests the engine can handle in parallel. Setting this to a value greater than 1 enables the listener to handle a second request while an earlier request is processed. The total throughput of a system can be affected by the number of threads operating. Default = 1 Max value = 99
Execution Time Limit	The maximum time a request can take to complete. A request that takes longer to complete terminates. Prevents runaway requests.
Polling Interval	The maximum wait interval between checks of the MSM queues. The higher the value, the longer the interval, and the fewer system resources used. The side effect of a high value is that the worker thread cannot respond to a stop command. Default = 2.0
Default Java File Encoding	The default encoding if incoming message is not self-declaring (that is, XML).

Property	Property Description
Agent Precedence	Sets the order for selecting the Execution Agent. The system selects the agent(s) to use to process the document by searching through the configuration dictionary. Usually, it looks for a document entry in the configuration and when a match is found, the agent specified in that document entry is selected. If a matching document entry is not found or no agent is specified, the engine looks in the input protocol configuration (listener). For the processing agent to be taken directly from the listener (thus ignoring the document entry), select *Listener Overrides Document*. An alternative is Document Overrides Listener. <document> overrides <listener> is the default value.
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined destinations.
Error Documents treated normally	If set to *true*, error documents are processed by configured preemitters.
Listener is Transaction Manager	If set to *true*, agents run in a local transaction. Agents can roll back uncompleted transactions.
Record in Activity Log(s)	If set to *true*, activity on this channel will be recorded in the activity logs. If set to *false*, the activity will not be recorded.

Reference: MSMQ Listener Special Registers

The following table lists and describes the special registers (SREGs) available on the MSMQ listener.

Name	Level	Type	Description
destination	System	String	The name of the destination queue.
msgsize	Document	Integer	The physical length of the message payload.
msmqlabel	Document	String	The message label of the received message.

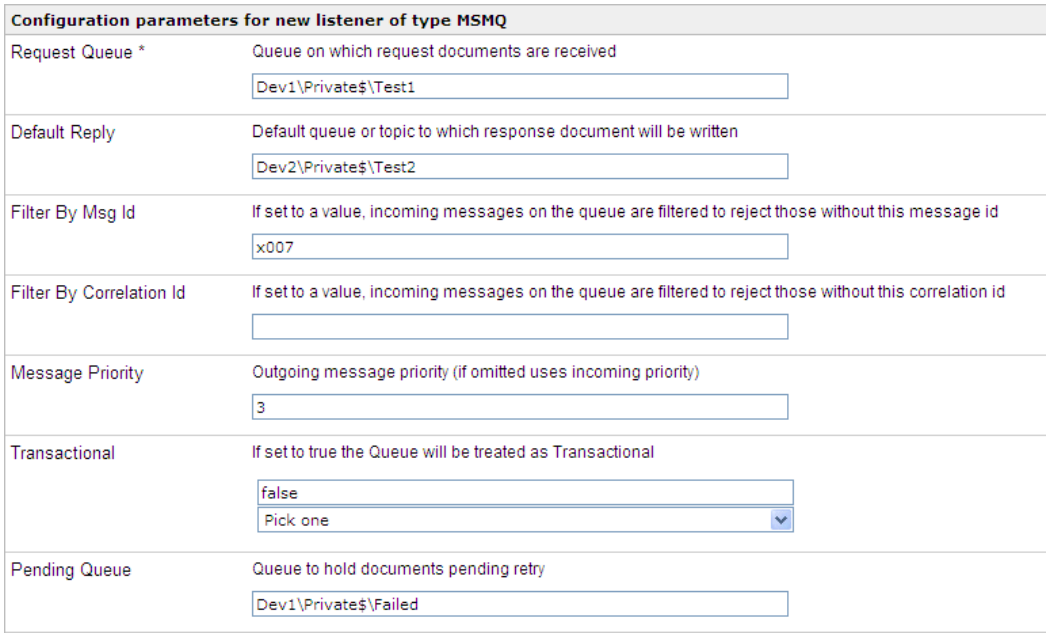
Name	Level	Type	Description
name	System	String	The assigned name of the listener.
priority	Document	Integer	The priority of the incoming message.
protocol	System	String	The protocol on which this message was received.
source	System	String	The queue on which the message was received.
tid	Document	String	Unique transaction ID.

MSMQ Listener Configuration Example

You are receiving documents on a private queue, named Test1, not listed in Active Directory, located on a development machine named Dev1. You would like incoming messages with a specific message header value directed to another queue on a second development box called Dev2. You also want Service Manager to keep requests that failed due to a back end data server not being available. Specify the values as follows:

- **Request Queue:** Dev1\Private$\Test1
- **Default Reply:** Dev2\Private$\Test2
- **Filter by Msg Id:** x007

- **Pending Queue:** Dev1\Private$\Failed



Configuring an MSMQ Emitter

To route an output document or error message to a protocol other than that of the listener, an emitter must be configured. An emitter is used to send documents outbound either on the same protocol that the document arrived on or alternatively, on a cross protocol.

You can return a processed document to one or more alternate destinations. By default, an output document is returned using the same protocol on which it was received. For example, an application may send input over TCP but want to route the output to MSMQ. In such a case, an emitter must be configured.

Note: Configuring an MSMQ emitter is not required if the emitter protocol used in the outlet of the channel is the same as the listener protocol used in the inlet of the channel. For more information on inlets and outlets, see *Defining an Inlet* on page 15 and *Defining an Outlet* on page 25.

MSMQ Emitter Properties

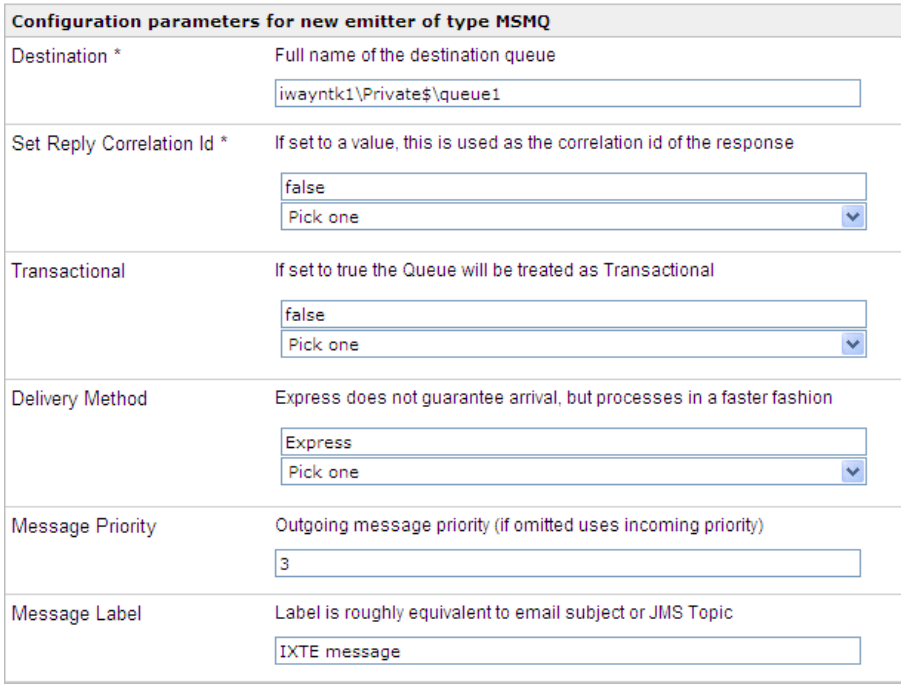
The following table lists and describes the MSMQ emitter properties. For instructions on creating an emitter, see *How to Create an Emitter* on page 29.

Property	Description
Destination (required)	Queue where the message is routed, in the format of *Host\queuetype\qName* where: *Host* Is the machine name where the Microsoft Queuing system is running. *queuetype* Private queues are queues that are not published in Active Directory and appear only on the local computer where they reside. Private queues are accessible only by Message Queuing applications that recognize the full path name or format name of the queue. *qName* Is the name of the queue where messages are placed, for example: `iwayntk1\Private$\queue1`
Set Reply Correlation Id (required)	If set to a value, this value sets the correlation ID of the response.
Transactional	If set to *true*, the queue will be treated as transactional.
Delivery Method	Select Express Delivery or Recoverable Delivery. Express Delivery does not guarantee arrival, but processing is faster.
Message Priority	Microsoft defines eight levels of message priority with values 0 through 7, where 0 is the lowest and 7 is the highest. Zero through three are considered normal settings, and four through seven, expedited. The Message Priority field is the default priority value set in the message header.
Message Label	Label is roughly equivalent to email subject or JMSQ Topic.

MSMQ Emitter Configuration Example

You want to route documents or error messages to a protocol other than a listener. You must configure an emitter to do this. Specify the values as follows:

- **Destination:** iwayntk1\Private$\queue1
- **Set Reply Correlation Id:** false



Oracle Advanced Queuing

Oracle Advanced Queuing (AQ) is the message queuing function of the Oracle database system. With Oracle AQ, you can perform message queuing operations in a manner similar to that of SQL operations.

The message queuing function of AQ enables asynchronous communication, using queues, between applications and users on Oracle databases. It offers:

- Multiple ways for applications to place a message in a queue.
- Multiple ways for applications to retrieve a message from a queue.
- Ways to distribute messages to appropriate queues.

- Guaranteed delivery of messages, with exception handling (when messages cannot be delivered).

AQ offers ways to prioritize the messages and offers time properties for messages such as expiration and delays. Notifications also are provided for immediate attention.

With AQ, message queuing operations benefit from the reliability, integrity, high availability, security, and scalability of a database. All of the message queuing operations are transactional. After messages are committed, they are guaranteed to be delivered. You can perform multiple message queuing and database operations in the same transaction. A database offers disaster protection for the messages. You also can use the advanced security features of the Oracle database.

The integration of message queuing with a database also offers unique benefits. Message queuing can use the management functions of a database. All the AQ operations are automatically audited, and you can look up the messaging information using an SQL view. You can use these SQL views to extract additional intelligence about the messaging environment.

Message queuing can take advantage of the system type of the Oracle database. Each message can be of an Oracle object type. Queuing brings the structure to the messaging system, which brings benefits such as better querying and content-based subscriptions.

AQ is used extensively in application integration, e-Businesses for online operations, and B2B exchanges.

Queuing Messages With Oracle AQ

iWay Service Manager can monitor AQ queues and read messages as they appear on the queue. The message is then processed and deleted after processing is complete. Before using the AQ adapter, you must add the Oracle-provided AQ JAR files to the class path.

For instructions on registering JAR files, see *Registering Library Files and Setting JVM Options* on page 40.

The Oracle AQ listener requires that you register jar files as follows:

Queue	JAR Files
RAW	❑ aqapi.jar ❑ classes12.zip These Java AQ API JAR files are located in the following directory: *%ORACLE_HOME%*/rdbms/jlib/ where: *%ORACLE_HOME%* Is the Oracle home directory on your system.
CLOB	❑ aqapi12.jar ❑ odbc14.jar
RAC	❑ aqapi12.jar ❑ ojdbc14.jar ❑ ons.jar

For instructions on registering JAR files, see *Registering Library Files and Setting JVM Options* on page 40.

Oracle AQ Listener Properties

The following tables list and describe the Oracle AQ listener properties. The individual tables correspond to the basic, AQ, JMS, and Advanced sections of the configuration window. For instructions on creating a listener, see *How to Create a Listener* on page 18.

Property Name	Property Description
SID (required)	The name of the Oracle database service.
Host (required)	The IP address or DNS name where the Oracle Service ID is installed.

Property Name	Property Description
Port (required)	The Oracle listener port. The default value, if not changed at Oracle installation time, is 1521.
Driver (required)	The type of driver to use for server connection. Select OCI or Thin.
User (required)	A valid Oracle user ID that has AQ_USER_ROLE granted by the AQ administrator and executes a privilege on DBMS_AQ.
Password (required)	A valid password for the Oracle user ID previously specified.
Table Owner/Schema (required)	The schema name. Usually the same name as the login name.
Table Name	The named repository for a set of queues and their messages. A queue table may contain numerous queues, each of which may have many messages. However, a given queue and its messages may exist in only one queue table. This queue table resides in the previously specified Oracle Service ID.
Receiver Name (required)	The name of the queue on which input documents will be received for processing by Service Manager.
Connection Type (required)	You can select Oracle JMS or Oracle AQ.

AQ

Property Name	Property Description
Message Type	Select CLOB or RAW type of message to send or create type if creating dynamic message queue on listener.
Payload Class	When using custom defined objects, specify a class that is in the classpath.
Payload Object	When using custom defined objects, specify the Oracle object to which the class will map. The owner of the schema will be prepended.
Object Owner	Usually the same as the login value.

RAC

Property Name	Property Description
RAC Hosts	Enter additional hosts for RAC failover/load balancing using the following format: `host1:port,host2:port,host3:port`
RAC Failover	Select *true* if you want to use a RAC connection for failover. By default, *false* is selected.
RAC Loadbalance	Select *true* if you want to use a RAC connection for load balancing. By default, *false* is selected.
ONS Hosts	Hosts emitting ONS information. For example: `host:port,host:port`
Cache Minimum	The minimum number of connections in the queue to be held in the cache.
Cache Maximum	The maximum number of connections in the queue to be held in the cache.
Cache Initial	The initial number of connections to create in the cache.
Cache Inactive Timeout	The maximum time a cached physical connection can remain idle.
Cache Abandoned Timeout	The maximum time that a connection can remain unused before the connection is closed and returned to the cache.
Cache Maximum Statements	The maximum number of statements that a connection keeps open.
Cache Property Interval	Sets the time interval at which the cache manager inspects and enforces all specified cache properties.

JMS

Property Name	Property Description
Form of Input	Select a topic or a queue listener.
Form of Acknowledgment	Select auto acknowledge, client provides, or duplicates permitted to choose how the listener acknowledges received messages.
Set Reply Correlation Id	If set to a value, this is used as the correlation ID of the response.
Default Output Form	Select a topic or a queue target.
Message Type	Select bytes or text type of message to send or create type if creating dynamic queue on listener.
JMSReplyTo	Queue or Topic (used for JNDI lookup).

Advanced

Property Name	Property Description
Retry	The interval between retrying pending requests contained in a message.
Accepts non-XML (flat) only	Select *true* if non-XML input is expected. If enabled, XML input still can be passed to the listener. Preparsers do not run.
Optimize Favoring	Use this option to customize listener performance. For smaller transactions, select *performance*. For large payloads that could monopolize the amount of memory used by Service Manager, select *memory*.
Multithreading	The number of documents that can be processed in parallel.
Maximum Threads	The parallel threads can grow to this count automatically on demand.
Execution Time Limit	The maximum time a request can take to complete. A request that takes longer to complete terminates. Prevents runaway requests.

Property Name	Property Description
Polling Interval	The interval (in seconds) after which the listener returns control to Service Manager to determine if there is new input or a stop listener is requested. The listener is constantly connected to the queue to retrieve incoming messages. The default value is 2.0.
Whitespace Normalization	Specifies how the parser treats whitespace in Element content. Choose *preserve* to turn off all normalization as prescribed by the XML Specification. Choose *condense* to remove extra whitespace in pretty printed documents and for compatibility with earlier versions.
Default Java File Encoding	The default encoding if incoming message is not self-declaring (that is, XML).
Agent Precedence	Sets the order by which Service Manager selects agents. Service Manager usually looks for a document entry in the configuration library and when a match is found, the agent specified in that entry is selected. If no matching document is found or no agent specified, the engine looks in the input protocol configuration (listener). For the processing agent to be taken directly from the listener (thus ignoring the document entry), use <listener> overrides <document>. <document> overrides <listener> is the default value, which is used to manage EDA documents.
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined destinations.
Error Documents treated normally	If set to *true*, error documents are processed by configured preemitters.
Listener is Transaction Manager	If set to *true*, agents run in a local transaction. Agents can roll back uncompleted transactions.
Record in Activity Log(s)	If set to *true*, activity on this channel will be recorded in the activity logs. If set to *false*, the activity will not be recorded.

Configuring an Oracle AQ Emitter

Messages are sent to the destinations at the completion of a workflow. The state of the document determines which destination is used. The order in which the destinations are used cannot be predicted. Destinations cannot accept responses. These configurations also are used to send a message over a different protocol.

Note: Configuring an Oracle AQ emitter is not required if the emitter protocol used in the outlet of the channel is the same as the listener protocol used in the inlet of the channel. For more information on inlets and outlets, see *Defining an Inlet* on page 15 and *Defining an Outlet* on page 25.

To route an output document or error message to a protocol other than that of the listener, you must configure an emitter. For example, an application may send input over TCP, but want to route the output to an Oracle AQ queue.

Oracle AQ Emitter Properties

The following tables list and describe the Oracle AQ emitter properties. The individual tables correspond to the basic, AQ, and JMS sections of the configuration window. For instructions on creating an emitter, see *How to Create an Emitter* on page 29.

Property Name	Property Description
Destination (required)	Type queue@host. When using RAC, leave the host section blank.
Host	The host name of the server connection.
SID (required)	The Oracle database service ID.
Port (required)	The Oracle listener port. The default value, if not changed at Oracle installation time, is 1521.
Driver (required)	The type of driver to use for the server connection. Select OCI or Thin.
User (required)	A valid Oracle user ID that has AQ_USER_ROLE granted by the AQ administrator and executes a privilege on DBMS_AQ.
Password (required)	A valid password for the Oracle user ID previously specified.

Property Name	Property Description
Table Owner/Schema (required)	The schema name. Usually the same name as the login name.
Table Name	A named repository for a set of queues and their messages. A queue table may contain numerous queues, each of which may have many messages. However, a given queue and its messages may exist in only one queue table. This queue table resides in the previously specified Oracle Service ID.
Connection Type (required)	You can select the Oracle JMS or Oracle AQ.

AQ

Property Name	Property Description
Message Type	You can select either a CLOB or RAW type of message to send. You can also create a type if creating dynamic message queue on listener.
Payload Class	When using custom defined objects, specify a class that is in the classpath.
Payload Object	When using custom defined objects, specify the Oracle object that the class will map to. The owner of the schema will be prepended.
Object Owner	This property value is usually the same as the login value.

RAC

Property Name	Property Description
RAC Hosts	Enter additional hosts for RAC failover/load balancing using the following format: `host1:port,host2:port,host3:port`

Property Name	Property Description
RAC Failover	If set to *true* , the RAC connection is used for failover. By default, *false* is selected.
RAC Loadbalance	If set to *true*, the RAC connection is used for load balancing. By default, *false* is selected.
ONS Hosts	Hosts emitting ONS information. For example: `host:port,host:port`
Cache Minimum	The minimum number of connections in the queue to be held in the cache.
Cache Maximum	The maximum number of connections in the queue to be held in the cache.
Cache Initial	The initial number of connections to create in the cache.
Cache Inactive Timeout	The maximum time a cached physical connection can remain idle.
Cache Abandoned Timeout	The maximum time that a connection can remain unused before the connection is closed and returned to the cache.
Cache Maximum Statements	The maximum number of statements that a connection keeps open.
Cache Property Interval	Sets the time interval at which the cache manager inspects and enforces all specified cache properties.

JMS

Property Name	Property Description
Message Priority	The outgoing message priority (if omitted, uses incoming priority) as integer.
Message Type	Select bytes or text type of message to send or create type if creating dynamic queue on listener.

Property Name	Property Description
Set Reply Correlation Id	If set to a value, this value is used as the correlation ID of the response.
Delivery Mode	Select Non-Persistent or Persistent. Retention of a message at the destination until its receipt is acknowledged is not guaranteed by a Persistent delivery mode.
JMSReplyTo	Select Queue or Topic (used for JNDI lookup).
Time to Live	The lifetime of the message (in milliseconds) as float. The default will live forever.

Reference: **AQ Listener Special Registers**

The following table lists and describes the special registers (SREGs) available on the Oracle AQ listener.

Name	Level	Type	Description
correlid	Document	String	The correlation ID.
destination	Document	String	The default destination for reply (from message).
iwayconfig	System	String	The current active configuration name.
msgid	Document	String	The message ID.
msgsize	Document	Integer	The physical length of the message payload.
name	System	String	The assigned name of the master (listener).
priority	Document	String	The priority of this message.
protocol	System	String	The protocol on which this message was received.
source	Document	String	The queue or topic on which message was received.
tid	Document	String	Unique transaction ID.

Oracle AQ Troubleshooting

The following table lists and describes errors that you may encounter when using an Oracle AQ listener.

Error	Reason	Solution
`fetching message errorcode[174] and errormsg[JMS-174: Class must be specified for queues with object payloads Use dequeue(deq_option, payload_fact) or dequeue(deq_option, sql_data_cl)]`	Queue specified was configured for a user-defined message type. Currently, the listener can only accommodate the RAW message type.	Redefine the message queue as payload_type of RAW. For additional information, see the Oracle AQ documentation.
`Exception: java.sql.SQLExceptio n: ORA-00600: internal error code, arguments: [kpotcprc: depth exceeded], [16], [], [], [], [], [], []: [`	Server is unable to find the receive queue specified during configuration. Possible cause is a typographical error during configuration.	Verify that the message queue exists in the queue table.
`java.sql.SQLExceptio n: Io exception: The Network Adapter could not establish the connection:`	If the Oracle instance is not running, this error can appear at the start up (or restart) of Service Manager.	Check with your Oracle DBA to ensure that your Oracle system is running.
`OracleDriver.class not found`	The JDBC driver for Oracle is release specific.	Verify that the correct JDBC driver is included in your class path. To ensure that the correct classes12.zip is being used, check the Service Manager class path.

Sonic Message Queuing

Sonic Message Queuing delivers a high performance and high reliability messaging system and provides certified Java Message Service Queue (JMSQ) implementation that includes both Queue Point-to-Point messaging and Publish/Subscribe messaging. It has an extended client/server feature that enables hierarchical security management. Sonic Message Queuing guarantees message persistence over the Internet. It contains message security, encryption, and certificate management.

The Sonic patent-pending Dynamic Routing Architecture (DRA) enables enterprises to participate in global e-Business exchanges through a single broker. When trading domains come online, Sonic dynamically discovers destinations and delivers messages between the servers on an optimized routing path. The Sonic architecture sets a foundation for high-throughput e-Business integration by robustly providing the following e-Business essentials: scalability, availability, and reliability.

The iWay Adapter for Sonic MQ provides bidirectional capability to and from Sonic destinations.

This topic describes how to:

- Listen on Sonic queues.
- Redirect output from a non-Sonic destination to a Sonic destination.

To redirect output to a destination, see *iWay Adapter for Sonic Emitter Functionality* on page 295.

Queuing Messages With Sonic

The iWay Adapter for Sonic MQ enables Service Manager to read messages arriving on a named queue or topic and route the messages to either another queue or topic, or to a non-Sonic destination. Similarly, messages received through any of the other iWay protocol adapter listeners can be directed to a Sonic queue. During the message flow, Service Manager can apply standard document analysis, validation, transformation, and business logic capabilities to the document.

The iWay Adapter for Sonic MQ provides bidirectional support for Sonic queues (a listener and an emitter). The adapter provides support for topics for a TopicSession in the Publish and Subscribe domain, as well as queue support (for a QueueSession in the PTP domain). Support is provided for persistent and non-persistent messages. The persistent option prevents messages from being lost in the event of a network or system failure. The iWay Adapter for Sonic MQ can operate over TCP, SSL, or HTTP(S).

The iWay Adapter for Sonic MQ includes high availability features. The adapter monitors the connection for an exception and re-establishes the connection if it was dropped. The adapter also supports load balancing. With load balancing enabled, a connect request can be redirected to another broker with a Sonic cluster. Similarly, failover capability is implemented. The adapter can redirect the message to another broker if it is unable to connect to the original broker.

Registering Sonic Client JAR Files

The following tables list the required .jar files for registration.

For instructions on registering JAR files, see *Registering Library Files and Setting JVM Options* on page 40.

For Listeners:

Type	JAR Files
SSL	❑ Sonic_Client.jar Found at %Sonic_Home%/lib/. Add the Sonic_Client to the Path Settings PreClassPath section. ❑ Sonic_Crypto.jar ❑ jsafe.jar ❑ sonic_SSL.jar ❑ certj.jar ❑ sslj.jar Add the files to the Path Settings PreClassPath section.

Type	JAR Files
SSL Client Certificate	❑ Sonic_Client.jar Found at %Sonic_Home%/lib/. Add the Sonic_Client to the Path Settings PreClassPath section. ❑ Sonic_Crypto.jar ❑ jsafe.jar ❑ sonic_SSL.jar ❑ certj.jar ❑ sslj.jar

For Emitters:

Type	JAR Files
TCP or HTTP	Sonic_Client.jar Found at %Sonic_Home%/lib/. Add the Sonic_Client to the Path Settings PreClassPath section.
SSL	❑ Sonic_Client.jar ❑ jsafe.jar ❑ sonic_SSL.jar ❑ certj.jar ❑ sslj.jar Found at %Sonic_Home%/lib. Add the JAR files to the Path Settings PreClassPath section.

For instructions on registering JAR files, see *Registering Library Files and Setting JVM Options* on page 40.

iWay Adapter for Sonic MQ Listener Capability

Sonic supports standard Internet protocols including Secure Socket Layer (SSL), HTTP, encryption, TCP, and security. iWay Service Manager can listen on Sonic queues using TCP, SSL, HTTP, and HTTPS protocols. Each protocol requires setup specific to the unique protocol. This topic details configuration information for each individual protocol.

Procedure: How to Configure Reconnect Support

1. Click the *Server* link at the top left of the Service Manager console.
2. Under Settings in the left pane, click *General Settings*.
3. In the Retry Interval entry field, enter a value to enable the listeners to retry the connection if it fails for external causes.

   The default value is 120 seconds.

   The retry interval is a global property that controls the configured listeners. It marks the frequency in seconds in which Service Manager attempts to reconnect to a broker if the connection is lost or cannot be established.

Configuring a Sonic Listener Using TCP or HTTP

The Transport Control Protocol (TCP) is a connection-oriented protocol that establishes a connection with another host before it sends data. Before a connection is started between two hosts, control messages called a handshake are sent out to initiate the connection. TCP is the default protocol of a Sonic broker installation. Client applications that are Internet-enabled generally use TCP/IP protocols.

Hypertext Transfer Protocol (HTTP), the underlying protocol used by the World Wide Web, defines how messages are formatted and transmitted and the actions web servers and browsers must take in response to various commands.

HTTP operates over TCP connections. After a successful connection, the client transmits a request message to the server, which returns a reply message.

By server design or by company security policy, proxy servers and firewalls frequently allow only HTTP-based traffic to pass through. You can establish a direct connection to a Sonic broker using HTTP tunneling as the protocol. However, because the HTTP tunneling protocol is significantly slower than TCP or SSL, this option is recommended only when TCP and SSL protocols are not available.

You require JAR files supplied by Sonic to emit messages to a Sonic message queue.

Sonic Listener Properties for TCP or HTTP

The following table lists and describes the Sonic listener properties for TCP or HTTP. For instructions on creating a listener, see *How to Create a Listener* on page 18.

Property Name	Property Description

Form of Input	`TOPIC`
	Is for a TopicSession in the Publish and Subscribe domain.
	`QUEUE`
	Is for a QueueSession in the PTP domain.

Receiver Name (required)	The name of the queue on which input documents will be received for processing by Service Manager.
Default Output Form	The topic or queue target.
Standard Reply To	The queue or topic to which the system writes response messages. The property may be overwritten by the configuration properties.
Broker URL (required)	The URL (address) used by the listener to connect to the Sonic broker. The format of the URL is Protocol://host:port. If Service Manager is listening on a Sonic broker configured to listen on TCP, the format of the URL is tcp://host:port. The default TCP port on which Sonic listens is 2506. This value is configured in the Sonic broker.ini file. When Service Manager listens to Sonic, the URL is in the form of http://host:port where the HTTP port is defined in the Sonic broker.ini file. The Sonic listener supports failover if unable to reach a particular broker. You must provide a comma-separated list of broker URLs. The client attempts to connect to brokers in the list, for example, tcp://host:port, tcp://host:port.
Pending Queue	If system resources are down or temporarily unavailable, requests that cannot be completed can be placed into the pending system for later execution. When listening on a TOPIC, the iWay adapter uses a Sonic queue for pending.

Property Name	Property Description
Form of Acknowledgment	Acknowledgment mode support: the session acknowledgment mode is either Transactional (to send and receive a series of messages and then explicitly commit or roll back the group) or in one of the following acknowledgment modes: **Client Provides:** An explicit acknowledge on a message acknowledges the receipt of all messages that were produced and consumed by the session that gives the acknowledgment. When a session is forced to recover, it restarts with its first unacknowledged message. **Duplicates Permitted:** The session lazily acknowledges the delivery of messages to consumers, possibly allowing some duplicate messages after a system outage. **Auto Acknowledge:** The session automatically acknowledges the client receipt of a message by successfully returning from a call to receive (synchronous mode) or when the session message listener successfully returns (asynchronous mode). The last message can be delivered again.
Send Persistently	The support for persistent and non-persistent messages. In the event of a network or system failure, the persistent option prevents messages from being lost. In the event of a broker or Service Manager failure, non-persistent messages are volatile. Persistent messages are saved to disk.

Property Name	Property Description
Durable Topic Subscriber	The support for durable topic subscribers. In Publish and Subscribe messaging, messages are not stored for later delivery unless the client establishes a subscription name that is associated with its user identity on the message broker. **False (Non-Durable Subscribers).** A client uses a topic subscriber to receive messages that were published to a topic. A regular topic subscription is not durable. Messages are delivered when active but never retained. **True (Durable Subscribers).** If a client must receive all messages published on a topic including those published while the subscriber is inactive, it uses a durable TopicSubscriber. The message broker retains a record of this durable subscription and ensures that all messages from the topic publishers are retained until they are either acknowledged by this durable subscriber or expire. Sessions with durable subscribers must always provide the same client identifier. In addition, each client must specify a name that uniquely identifies (within the client identifier) each durable subscription it creates. Within a session, durable topic subscribers must be uniquely named.
Message Priority	The Sonic broker attempts to deliver messages with a higher priority ahead of messages with lower values. Priority is suggested to the JMSQ provider and can only order the delivery of undelivered messages. JMSQ defines ten levels of message priority with values 0 through 9, where 0 is the lowest, and 9 is the highest. Zero through four are considered normal settings and five through nine, expedited. The Message Priority field is the default priority value set in the JMSQ header.
Set Reply Correlation Id	If set to a value, that value is used as the correlation ID of the response.
Load Balance	If set to *true*, the load balancing option is enabled on the Service Manager listener. With load balancing enabled, a connect request can be redirected to another broker within a Sonic cluster, if load balancing was not disabled on the broker side.

Property Name	Property Description
Duplicates Detect	You can configure the broker to commit transactions to index a universally unique, 64-character identifier (UUID) supplied by the sender. The sender then uses a commit method that commits the transacted messages, unless a previously sent transaction identifier is still unexpired. Otherwise, the method forces a rollback of the transaction. For more information, see the INDEXED_TXN_ server properties in the installation section of the *Sonic MQ V5 Configuration and Administration Guide*.
User	A valid user ID defined to Sonic. You can use the Sonic Explorer Users option (found under message broker) to display users defined to Sonic. The user ID is required only when Sonic security is enabled.
Password	A valid user ID and password combination defined to Sonic. Use the Sonic Explorer Users option (found under message broker) to display users defined to Sonic.
Client ID	The broker retains messages for durable subscriber, using the userName and clientID of the connection plus the subscriptionName to index the subscription. The Client ID is a unique identifier that can prevent conflicts for durable subscriptions when many clients use the same user name and the same subscription name.
Connection ID	Identifies the connection. When combined with Client ID, must be unique at the broker.
Subscription Name	A subscription name always includes the name of the topic. To distinguish different message selectors used in subscriptions, you can include a string that helps identify the message selector. For example, you can use a subscription named Atlas_priority4 for a subscription to the Atlas topic with selector JMSPriority=4. This construct enables you to create many durable subscriptions that are easily understood and nonconflicting.

Property Name	Property Description
Message Selector	In PTP domains, all message selection takes place on the server. However, in Pub/Sub domains, all messages for a subscribed topic are delivered by default to the subscriber, and then, the filter is applied to decide what is consumed. Message selectors can consist of: ❏ Literals and indefinites ❏ Operators and expressions ❏ Comparison tests ❏ Parentheses ❏ White space
Output Message Type	Select *Bytes*, *Text*, or *Dynamic*.
Prefetch Count	Only applies to queue messages. Does not apply to topic messages. Prefetch count is the number of messages buffered locally. A worker (thread) prefetches messages to this limit. Injudicious use of this configuration can result in unbalanced thread use and appear to cause worker starvation. For example, if a prefetch count is set to 3 and two workers are defined and three messages are in the queue, the first worker prefetches all three messages. The second worker finds the queue empty and awaits the arrival of yet another message. Thus, it appears that the first worker processes three messages while the second does no work. To achieve balance, set the prefetch count to 1 and the prefetch threshold to 0. Use of prefetch count and threshold can result in improved performance through overall reduction in network traffic to the broker.
Prefetch Threshold	Fewer messages cause prefetch to be initiated. For additional information, see the description of the Prefetch Count property.
Duration	The maximum time a document can remain in the retry pending queue.
Retry	The interval between retrying pending requests.

Property Name	Property Description
Preserve Undelivered	The preserve messages that are not retrieved to a dead letter queue.
Notify Undelivered	Notify the broker administrator if undelivered messages are preserved.
Sequential	Determines the order in which the connection to the brokers is made when multiple brokers are listed. If this property is enabled, connection occurs sequentially. If disabled, connection occurs in a random manner.
Fault Tolerant	Determines whether the listener is connecting to the Sonic Broker as a fault tolerant client or not. The default value is false, that is, the listener is not connecting as a fault tolerant client. This feature is supported only when the Sonic Broker is installed with a Sonic Fault Tolerance license code.
Reconnect Fault Timeout	Sets the interval, in seconds, of how long to wait before attempting to reconnect to the broker if the session is lost.
Initial Connect Timeout	Sets the interval, in seconds, of how long to wait before attempting to connect to the broker, or to another broker on the list, if connection fails when the listener is first started.
Whitespace Normalization	Specifies how the parser treats whitespace in Element content. Choose *preserve* to turn off all normalization as prescribed by the XML Specification. Choose *condense* to remove extra whitespace in pretty printed documents and for compatibility with earlier versions.
Accepts non-XML (flat) only	If set to *true*, then non-XML input is expected. If enabled, XML input still can be passed to the listener. Preparsers do not run.
Optimize Favoring	Use this option to customize listener performance. For smaller transactions, select *performance*. For large payloads that could monopolize the amount of memory used by Service Manager, select *memory*.

Property Name	Property Description
Multithreading	The number of worker threads. (Equivalent to the number of requests that Service Manager can handle in parallel.) Setting this to a value of greater than 1 enables the listener to handle a second request while an earlier request is being processed. The total throughput of a system can depend on the number of threads operating. Default: 1 Max value: 99
Execution Time Limit	The maximum time a request can take to complete. A request that takes longer to complete terminates. Prevents runaway requests. **Note:** The kill interval property checks for runaway requests that have exceeded their maximum life. Default is 60 (seconds). Duration is xxhxxmxxs, for example, 1h2m3s = one hour, two minutes, and three seconds.
Polling Interval	Indicates frequency in seconds that the listener returns control to Service Manager to determine if a stop listener was requested. Listener is constantly connected to the queue to retrieve incoming messages. The default value is 2.0.
Default Java File Encoding	The default encoding if incoming message is not self-declaring (that is, XML).
Agent Precedence	Sets the order by which Service Manager selects agents. Service Manager selects the agent(s) to process the document by searching through the configuration dictionary. Usually, it looks for a document entry in the configuration and when a match is found, the agent specified in that document entry is selected. If a matching document entry is not found or no agent is specified, the engine looks in the input protocol configuration (listener). To have the processing agent taken directly from the listener (thus ignoring the document entry), use <listener> overrides <document>. <document> overrides <listener> is the default value.
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined destinations.

Property Name	Property Description
Error Documents treated normally	If set to *true*, error documents are processed by configured preemitters.
Listener is Transaction Manager	If set to *true*, agents run in a local transaction. Agents can roll back uncompleted transactions.
Record in Activity Log(s)	If set to *true*, activity on this channel will be recorded in the activity logs, else the activity will not be recorded.

Note: The Sonic listener supports streaming. Streaming is used for large documents or documents for which the application needs to split the input into sections under the same transaction. For more information on streaming and configuring streaming preparsers, see the *iWay Service Manager Component and Functional Language Reference Guide*.

Sonic TCP Listener Configuration Example

You would like to listen on a Sonic MQ queue named SampleQ2, hosted by a broker on machine iwxfoc, which listens on default Sonic MQ port 2056. You would also like to output messages to the queue SampleQ4 on the same broker. The broker does not have security enabled.

To configure the components needed to support this scenario, you need to configure the following in Service Manager:

Note: For information on the steps required to accomplish these tasks, see the *iWay Service Manager User's Guide*.

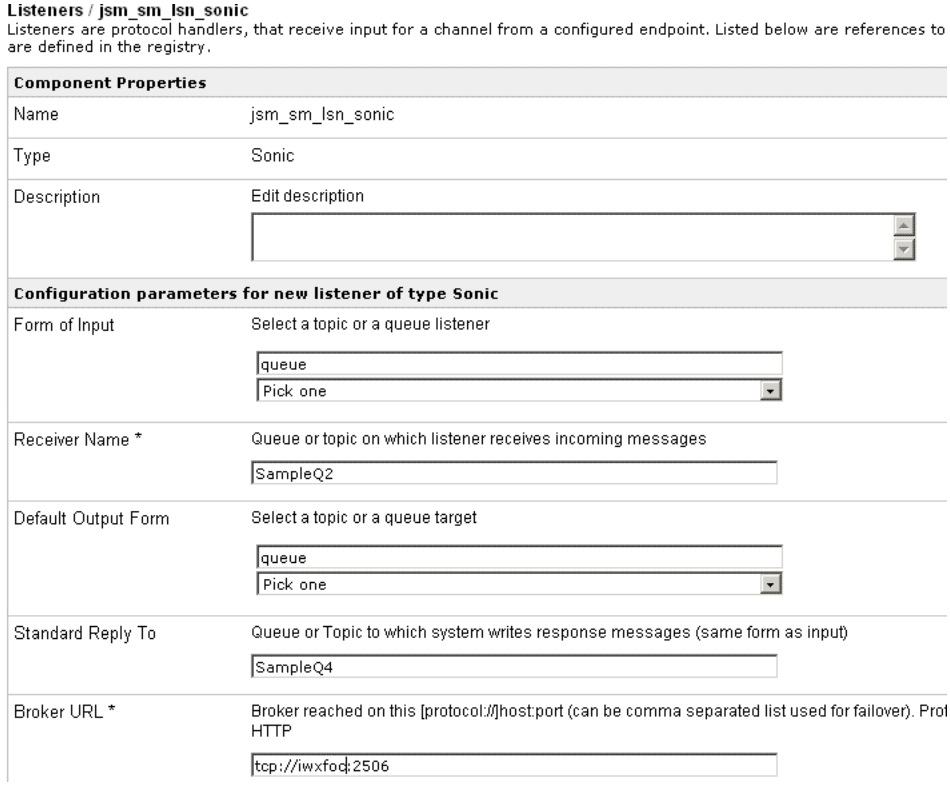
- ❑ Copy the Default Channel provided with SM installation and rename it as sonic_lsn_channel.
- ❑ Create a listener of type Sonic, and call it jsm_sm_lsn_sonic. The listener will connect to Sonic MQ on iwxfoc machine (tcp://iwxfoc:2506). It will get messages from SampleQ2 queue and output them into a queue named SampleQ4 at the same broker.
- ❑ Add the listener to an Inlet.
- ❑ Add the Inlet to the Channel.

The other parts of the channel will stay the same as default channel. After building and deploying the sonic_lsn_channel, you can see messages being transferred from SampleQ2 to SampleQ4 on the iwxfoc Sonic MQ broker.

Specify the listener values as follows. Leave default values for all other fields.

- **Type:** Sonic
- **Form of Input:** queue
- **Receiver Name:** SampleQ2
- **Standard Reply To:** SampleQ4
- **Broker URL:** tcp://iwxfoc :2506

The following image shows the Sonic listener configuration pane.



Configuring a Sonic Listener Using SSL

Secure Sockets Layer (SSL) provides authentication and encryption techniques that require the broker to set the appropriate properties and certificates on a security-enabled database. The client also must have the appropriate libraries and properties to call an SSL connection. A complete implementation sample is provided in the Protocols section of the *Sonic MQ V5 Configuration and Administration Guide*.

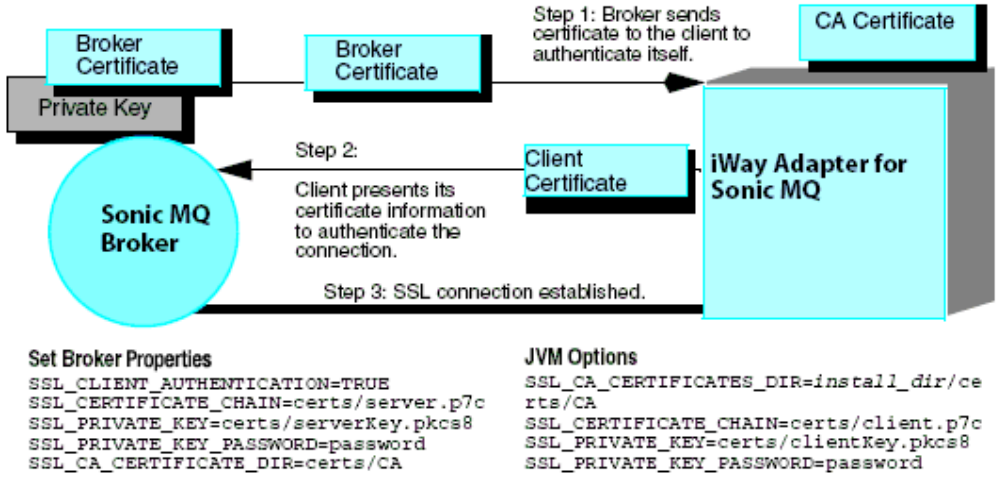
Note: You must have SSL libraries installed from RSA with the appropriate Sonic license or from supported third-parties such as Institute for Applied Information Processing and Communications (IAIK) or Java Secure Socket Extension (JSSE).

The manner in which you configure SSL for your application depends on whether you implement client authentication with a user name and password or with a client certificate. The following topics describe the different configurations between these implementations:

- *Configuring a Sonic Listener Using SSL Client Certificate* on page 285
- *Configuring Sonic Emitter Using SSL* on page 300

When the Sonic broker is configured with the SSL property SSL_CLIENT_AUTHENTICATION=FALSE in the broker.ini configuration file, no client certificate is required, and the client is authenticated with a user name and password. The client reads the user name and password and then passes them to the broker.

The following diagram shows Step 1, in which the broker sends the certificate to the client to authenticate itself; Step 2, in which the client presents its certificate information to authenticate the connection; and Step 3, in which an SSL connection is established.



You must add JAR files, supplied by Sonic, to emit messages to a Sonic message queue using the SSL protocol. For more information, see *Registering Sonic Client JAR Files* on page 264.

Setting Java System Properties for Sonic SSL

For instructions on setting Java system properties, see *Registering Library Files and Setting JVM Options* on page 40.

In the Property field, type *SSL_CA_CERTIFICATES_DIR*. In the Value field, type the location of the CA certificate to be used by Service Manager.

The default certificate shipped with the Sonic product is located in:

```
%SONIC_HOME%\certs\ca
```

Sonic Listener Properties for SSL

The following table lists and describes the Sonic listener properties for SSL. For instructions on creating a listener, see *How to Create a Listener* on page 18.

Property Name	Property Description

Form of Input	`TOPIC`
	Is for a TopicSession in the Publish and Subscribe domain.
	`QUEUE`
	Is for a QueueSession in the PTP domain.

Receiver Name (required)	The name of the queue on which input documents will be received for processing by Service Manager.
Default Output Form	The topic or queue target.
Standard Reply To	The queue or topic to which the system writes response messages. The property may be overridden by the configuration properties.
Broker URL (required)	The URL (address) used by the listener to connect to the Sonic broker. The format of the URL is Protocol://host:port. If Service Manager is listening on a Sonic broker configured to listen on SSL, the URL would be in the format of ssl://host:port. This value is configured in the Sonic broker.ini file. The Sonic listener supports failover if unable to reach a particular broker. You must provide a comma-separated list of broker URLs. The client attempts to connect to brokers in this list, for example, ssl://host:port.
Pending Queue	Requests that cannot be completed because system resources are down or temporarily unavailable can be placed into the pending system for later execution. When listening on a TOPIC, the adapter uses a Sonic queue for pending.

Property Name	Property Description
Form of Acknowledgment	Acknowledgment mode support: The session acknowledgment mode is either Transactional (to send and receive a series of messages and then explicitly commit or rollback the group) or one of the available acknowledgment modes. **Client Provides:** An explicit acknowledge on a message acknowledges the receipt of all messages that are produced and consumed by the session that gives the acknowledgment. When a session is forced to recover, it restarts with its first unacknowledged message. **Duplicates Permitted:** The session lazily acknowledges the delivery of messages to consumers, possibly allowing some duplicate messages after a system outage. **Auto Acknowledge:** The session automatically acknowledges the client receipt of a message by successfully returning from a call to receive it (synchronous mode) or when the session message listener successfully returns the receipt (asynchronous mode). The last message can be delivered again.
Send Persistently	Support for persistent and non-persistent messages. In the event of a network or system failure, the persistent option prevents messages from being lost. In the event of a broker or iWay Service Manager failure, non-persistent messages are volatile. Persistent messages are saved to disk.

Property Name	Property Description
Durable Topic Subscriber	Support for durable topic subscribers. In Publish and Subscribe messaging, messages are not stored for later delivery unless the client establishes a subscription name that is associated with its user identity on the message broker. **False (Non-durable subscribers).** A client uses a topic subscriber to receive messages that have been published to a topic. A regular topic subscription is not durable. Messages are delivered when active but never retained. **True (Durable subscribers).** If a client must receive all the messages published on a topic including those published while the subscriber is inactive, it uses a durable TopicSubscriber. The message broker retains a record of this durable subscription and ensures that all messages from the topic publishers are retained until they are either acknowledged by this durable subscriber or expire. Sessions with durable subscribers must always provide the same client identifier. In addition, each client must specify a name that uniquely identifies (within the client identifier) each durable subscription it creates. Within a session, durable topic subscribers must be uniquely named.
Message Priority	The Sonic broker attempts to deliver messages with a higher priority ahead of messages with lower values. Priority is suggested to the JMSQ provider, and only orders delivery of undelivered messages. JMSQ defines ten levels of message priority with values 0 through 9, where 0 is the lowest and 9 is the highest. Zero through four are considered normal settings and five through nine, expedited. The Message Priority field is the default priority value set in the JMSQ header.
Set Reply Correlation Id	If set to a value, that value used as the correlation ID of the response.
Load Balance	Select *true* to enable the load balancing option on the Service Manager listener. With load balancing enabled, a connect request can be redirected to another broker within a Sonic cluster, if load balancing was not disabled on the broker side.

Property Name	Property Description
Duplicates Detect	You can configure the broker to commit transactions such that they index a universally unique, 64-character identifier (UUID) supplied by the sender. The sender then uses a commit method that commits the transacted messages, unless a previously sent transaction identifier is still unexpired. Otherwise, the method forces a rollback of the transaction. For more information, see the INDEXED_TXN_ server properties in the installation section of the *Sonic MQ V5 Configuration and Administration Guide*.
User	A valid user ID defined to Sonic. You can use the Sonic Explorer Users option (found under message broker) to display users defined to Sonic. The user ID is required when listening on a Sonic queue with client authentication.
Password	A valid user ID and password combination defined to Sonic. You can use the Sonic Explorer Users option (found under message broker) to display users defined to Sonic. The password is required when listening on a Sonic queue with client authentication.
Client ID	The broker retains messages for durable subscriber, using the userName and clientID of the connection, plus the subscriptionName to index the subscription. The Client ID is a unique identifier that can prevent conflicts for durable subscriptions when many clients are using the same user name and the same subscription name.
Connection ID	The identifies the connection. When combined with Client ID, must be unique at the broker.
Subscription Name	A subscription name always includes the name of the topic. To distinguish different message selectors used in subscriptions, you can include a string that helps identify the message selector. For example, you can use a subscription named Atlas_priority4 for a subscription to the Atlas topic with selector JMSPriority=4. This construct enables you to create many durable subscriptions that are easily understood and non-conflicting.

Property Name	Property Description
Message Selector	In PTP domains, all message selection takes place on the server. However, in Pub/Sub domains, all messages for a subscribed topic are by default delivered to the subscriber and then, the filter is applied to decide what is consumed. Message selectors can consist of: ❏ Literals and indefinites ❏ Operators and expressions ❏ Comparison tests ❏ Parentheses ❏ White space
Output Message Type	Select *Bytes*, *Text*, or *Dynamic*.
Prefetch Count	Only applies to queue messages; does not apply to topic messages. Prefetch count is the number of messages buffered locally. A worker (thread) prefetches messages to this limit. Injudicious use of this configuration can result in unbalanced thread use and appear to cause worker starvation. For example, if a prefetch count is set to 3 and two workers are defined and three messages are in the queue, the first worker prefetches all three messages. The second worker finds the queue empty and awaits the arrival of yet another message. Thus, it appears that the first worker processes three messages while the second does no work. To achieve balance, set the prefetch count to 1 and the prefetch threshold to 0. Use of prefetch count and threshold can result in improved performance through overall reduction in network traffic to the broker.
Prefetch Threshold	Fewer messages cause prefetch to be initiated. For additional information, see the description of the Prefetch Count property.
Duration	The maximum time that a document can remain in the retry pending queue.
Retry	The interval between retrying pending requests.

Property Name	Property Description
Preserve Undelivered	The preserve messages that are not retrieved to a dead letter queue.
Notify Undelivered	Notifies the broker administrator if undelivered are preserved.
Sequential	Determines the order in which the connection to the brokers is made when multiple brokers are listed. If this property is enabled, connection occurs sequentially. If disabled, connection occurs in a random manner.
Fault Tolerant	Determines whether the listener is connecting to the Sonic Broker as a fault tolerant client or not. The default value is false, that is, the listener is not connecting as a fault tolerant client. This feature is supported only when the Sonic Broker is installed with a Sonic Fault Tolerance license code.
Reconnect Fault Timeout	Sets the interval, in seconds, of how long to wait before attempting to reconnect to the broker if the session is lost.
Initial Connect Timeout	Sets the interval, in seconds, of how long to wait before attempting to connect to the broker, or to another broker on the list, if connection fails when the listener is first started.
Whitespace Normalization	Specifies how the parser treats whitespace in Element content. Choose *preserve* to turn off all normalization as prescribed by the XML Specification. Choose *condense* to remove extra whitespace in pretty printed documents and for compatibility with earlier versions.
Accepts non-XML (flat) only	If set to *true*, non-XML input is expected. If enabled, XML input still can be passed to the listener. Preparsers do not run.
Optimize Favoring	Use this option to customize listener performance. For smaller transactions, select *performance*. For large payloads that could monopolize the amount of memory used by Service Manager, select *memory*.

Property Name	Property Description
Multithreading	The number of worker threads. (Equivalent to the number of requests that Service Manager can handle in parallel.) Setting this to a value of greater than 1 enables the listener to handle a second request while an earlier request is being processed. The total throughput of a system is affected by the number of threads operating. Default: 1 Max value: 99
Execution Time Limit	The maximum time a request can take to complete. A request that takes longer to complete terminates. Prevents runaway requests. **Note:** The kill interval property checks for runaway requests that exceed their maximum life. Default is 60 (seconds). Duration is xxhxxmxxs, for example, 1h2m3s = one hour, two minutes, and three seconds.
Polling Interval	Indicates frequency in seconds that the listener returns control to Service Manager to determine if a stop listener was requested. The listener is constantly connected to the queue to retrieve incoming messages. The default value is 2.0.
Default Java File Encoding	The default encoding if incoming message is not self-declaring (that is, XML).
Agent Precedence	Sets the order by which Service Manager selects agents. Service Manager usually looks for a document entry in the configuration dictionary and when a match is found, the agent specified in that document entry is selected. If a matching document entry is not found or no agent is specified, the engine looks in the input protocol configuration (listener). To have the processing agent taken directly from the listener (thus ignoring the document entry), use <listener> overrides <document>. <document> overrides <listener> is the default value.
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined destinations.
Error Documents treated normally	If set to *true*, error documents are processed by configured preemitters.

Property Name	Property Description
Listener is Transaction Manager	If set to *true*, agents run in a local transaction. Agents can roll back uncompleted transactions.
Record in Activity Log(s)	If set to *true*, activity on this channel will be recorded in the activity logs, else the activity will not be recorded.

Note: The Sonic listener supports streaming. Streaming is used for large documents or documents for which the application needs to split the input into sections under the same transaction. For more information on streaming and configuring streaming preparsers, see the *iWay Service Manager Component and Functional Language Reference Guide*.

Reference: Sonic Listener Special Registers

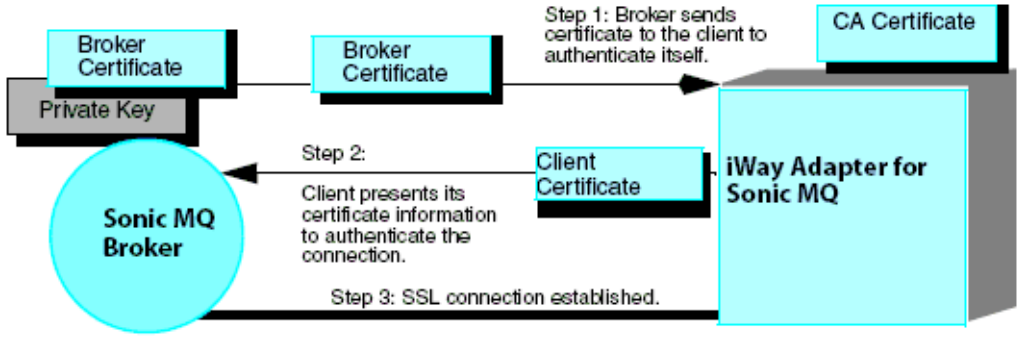
The following table lists and describes the special registers (SREGs) available on the Sonic listener.

Name	Level	Type	Description
correlid	Document	String	The correlation ID.
destination	Document	String	The default destination for reply (from message).
iwayconfig	System	String	The current active configuration name.
msgid	Document	String	The message ID.
msgsize	Document	Integer	The physical length of the message payload.
name	System	String	The assigned name of the master (listener).
priority	Document	String	The priority of this message.
protocol	System	String	The protocol on which this message was received.
source	Document	String	The queue or topic on which message was received.
tid	Document	String	Unique transaction ID.

Configuring a Sonic Listener Using SSL Client Certificate

When you configure the Sonic broker with the SSL property SSL_CLIENT_AUTHENTICATION=TRUE, you can achieve authentication by exchanging digital certificates between Service Manager and the Sonic broker. In addition to this mutual authentication, Service Manager can present a user name and password to the broker at run time (optional).

The following diagram shows Step 1, in which the broker sends the certificate to the client to authenticate itself; Step 2, in which the client presents its certificate information to authenticate the connection; and Step 3, in which an SSL connection is established.



Set Broker Properties

```
SSL_CLIENT_AUTHENTICATION=TRUE
SSL_CERTIFICATE_CHAIN=certs/server.p7c
SSL_PRIVATE_KEY=certs/serverKey.pkcs8
SSL_PRIVATE_KEY_PASSWORD=password
SSL_CA_CERTIFICATE_DIR=certs/CA
```

JVM Options

```
SSL_CA_CERTIFICATES_DIR=install_dir/certs/CA
SSL_CERTIFICATE_CHAIN=certs/client.p7c
SSL_PRIVATE_KEY=certs/clientKey.pkcs8
SSL_PRIVATE_KEY_PASSWORD=password
```

JAR files supplied by Sonic are required to emit messages to a Sonic message queue using the SSL protocol. For more information, see *Registering Sonic Client JAR Files* on page 264.

Reference: JVM Options for SSL With Certificates

The following table lists and describes the properties required to configure the appropriate environment for communication to Sonic using SSL with certificates.

For instructions on setting Java system properties, see *Registering Library Files and Setting JVM Options* on page 40. Complete the Property and Value fields as described here.

Property	Description
SSL_CA_CERTIFICATES_DIR SSL_CA_CERTIFICATES_DIR={certs/ca|path} SSL_CA_CERTIFICATES_DIR_n={certs/ca|path}	Specifies the location of the Certificate Authority (CA) certificate. The path must be fully qualified or relative to the Service Manager installation directory. The default directory is certs/ca.
SSL_CERTIFICATE_CHAIN SSL_CERTIFICATE_CHAIN={certs/server.p7c|path} SSL_CERTIFICATE_CHAIN_n={certs/server.p7c|path}	Specifies the location of the file containing the client keystore certificate chain for SSL. The path must be fully qualified or relative to the Service Manager installation directory. The default directory is certs/server.p7c.
SSL_CERTIFICATE_CHAIN_FORM SSL_CERTIFICATE_CHAIN_FORM={LIST|PKCS12|PKCS7} SSL_CERTIFICATE_CHAIN_FORM_n={LIST|PKCS12|PKCS7}	Specifies the format of the file containing the certificate chain. PKCS7 is the default value, a comma-delimited list of path names that point to files containing each individual certificate in the chain.
SSL_PRIVATE_KEY SSL_PRIVATE_KEY={serverKey.pkcs8|path} SSL_PRIVATE_KEY_n={serverKey.pkcs8|path}	Provides the location of the file containing the client encrypted private key for SSL. This path must be fully qualified or relative to the Service Manager installation directory. The default value is serverKey.pkcs8.

Property	Description

`SSL_PRIVATE_KEY_PASSWORD`
`SSL_PRIVATE_KEY_PASSWORD={password|password}`
`SSL_PRIVATE_KEY_PASSWORD_n={password|password}`
	Provides the password that encrypts the private key for SSL.
	The default value is password.

Sonic Listener Properties for SSL With Client Certificate

The following table lists and describes the Sonic listener properties for SSL with client certificate. For instructions on creating a listener, see *Configuring Listeners* on page 18.

Property Name	Property Description

Form of Input	`TOPIC`
	Is for a TopicSession in the Publish and Subscribe domain.
	`QUEUE`
	Is for a QueueSession in the PTP domain.

Receiver Name (required)	The name of the queue on which input documents will be received for processing by Service Manager.
Default Output Form	The topic or queue target.
Standard Reply To	The queue or topic to which the system writes response messages. The property may be overwritten by the configuration properties.

Property Name	Property Description
Broker URL (required)	The URL (address) used by the listener to connect to the Sonic broker. The format of the URL is Protocol://host:port. If Service Manager is listening on a Sonic broker configured to listen on TCP, the format of the URL is tcp://host:port. The default TCP port on which Sonic listens is 2506. This value is configured in the Sonic broker.ini file. When iWay Service Manager listens to Sonic, the URL is in the form of http://host:port where the HTTP port is defined in the Sonic broker.ini file. The Sonic listener supports failover if unable to reach a particular broker. You must provide a comma-separated list of broker URLs. The client attempts to connect to brokers in this list, for example, tcp://host:port, tcp://host:port.
Pending Queue	If the system resources are down or temporarily unavailable, requests that cannot be completed can be placed into the pending system for later execution. When listening on a TOPIC, the iWay adapter uses a Sonic queue for pending.
Form of Acknowledgment	Acknowledgment mode support: The session acknowledgment mode is either Transacted (to send and receive a series of messages and then explicitly commit or roll back the group) or in one of the available acknowledgment modes. **Client Acknowledge:** An explicit acknowledge on a message acknowledges the receipt of all messages that are produced and consumed by the session that gives the acknowledgment. When a session is forced to recover, it restarts with its first unacknowledged message. **Duplicates OK Acknowledge:** The session lazily acknowledges the delivery of messages to consumers, possibly allowing some duplicate messages after a system outage. **Auto Acknowledge:** The session automatically acknowledges the client receipt of a message by successfully returning from a call to receive (synchronous mode) or when the session message listener successfully returns (asynchronous mode). The last message can be delivered again.

Property Name	Property Description
Send Persistently	The support for persistent and non-persistent messages. The persistent option prevents messages from being lost in the event of a network or system failure. In the event of a broker or Service Manager failure, non-persistent messages are volatile. Persistent messages are saved to disk.
Durable Topic Subscriber	Support for durable topic subscribers. In Publish and Subscribe messaging, messages are not stored for later delivery unless the client establishes a subscription name that is associated with its user identity on the message broker. **False (Non-Durable Subscribers).** A client uses a topic subscriber to receive messages that are published to a topic. A regular topic subscription is not durable. Messages are delivered when active but never retained. **True (Durable Subscribers).** If a client must receive all the messages published on a topic including those published while the subscriber is inactive, it uses a durable TopicSubscriber. The message broker retains a record of this durable subscription and ensures that all messages from the topic publishers are retained until they are either acknowledged by this durable subscriber or expire. Sessions with durable subscribers must always provide the same client identifier. In addition, each client must specify a name that uniquely identifies (within the client identifier) each durable subscription it creates. Within a session, durable topic subscribers must be uniquely named.
Message Priority	The Sonic broker attempts to deliver messages with a higher priority ahead of messages with lower values. Priority is suggested to the JMSQ provider and can only order the delivery of undelivered messages. JMSQ defines ten levels of message priority with values 0 through 9, where 0 is the lowest and 9 is the highest. Zero through four are considered normal settings and five through nine, expedited. The Message Priority field is the default priority value set in the JMSQ header.

Property Name	Property Description
Set Reply Correlation Id	If set to a value, that value is used as the correlation ID of the response.
Load Balance	Select the load balancing option on the Service Manager listener for load balancing. With load balancing enabled, a connect request can be redirected to another broker within a Sonic cluster, if load balancing was not disabled on the broker side.
Duplicates Detect	You can configure the broker to commit transactions to index a universally unique, 64-character identifier (UUID) supplied by the sender. The sender then uses a commit method that commits the transacted messages, unless a previously sent transaction identifier is still unexpired. Otherwise, the method forces a rollback of the transaction. For more information, see the INDEXED_TXN_ server properties in the installation section of the *Sonic MQ V5 Configuration and Administration Guide*.
User	A valid user ID defined to Sonic. You can use the Sonic Explorer Users option (under message broker) to display users defined to Sonic. User ID only required if Sonic security is enabled.
Password	A valid user ID and password combination defined to Sonic. Use the Sonic Explorer Users option (under message broker) to display users defined to Sonic.
Client ID	The broker retains messages for durable subscriber, using the userName and clientID of the connection plus the subscriptionName to index the subscription. The Client ID is a unique identifier that can prevent conflicts for durable subscriptions when many clients are using the same user name and the same subscription name.
Connection ID	Identifies the connection. When combined with Client ID, must be unique at the broker.

Property Name	Property Description
Subscription Name	A subscription name always includes the name of the topic. To distinguish between different message selectors used in subscriptions, you can include a string that helps identify the message selector. For example, you can use a subscription named Atlas_priority4 for a subscription to the Atlas topic with selector JMSPriority=4. This construct lets you create many durable subscriptions that are easily understood and nonconflicting.
Message Selector	In PTP domains, all message selection takes place on the server. However, in Pub/Sub domains, all messages for a subscribed topic are delivered by default to the subscriber, and then, the filter is applied to decide what is consumed. Message selectors can consist of: ❑ Literals and indefinites ❑ Operators and expressions ❑ Comparison tests ❑ Parentheses ❑ White space
Output Message Type	Select *Bytes*, *Text Message*, or *Dynamic*.

Property Name	Property Description
Prefetch Count	Only applies to queue messages; does not apply to topic messages. Prefetch count is the number of messages buffered locally. Any worker (thread) prefetches messages to this limit. Injudicious use of this configuration can result in unbalanced thread use and appear to cause worker starvation. For example, if a prefetch count is set to 3 and two workers are defined and three messages are in the queue, the first worker prefetches all three messages. The second worker finds the queue empty and awaits the arrival of yet another message. Thus, it appears that the first worker processes three messages while the second does no work. To achieve balance, set the prefetch count to 1 and the prefetch threshold to 0. Use of prefetch count and threshold can result in improved performance through overall reduction in network traffic to the broker.
Prefetch Threshold	Fewer messages cause prefetch to be initiated. For additional information, see the Prefetch Count property.
Duration	The maximum time that a document can remain in the retry pending queue.
Retry	The interval between retrying pending requests.
Preserve Undelivered	Preserves messages that are not retrieved to a dead letter queue.
Notify Undelivered	Notifies the broker administrator if undelivered are preserved.
Sequential	Determines the order in which the connection to the brokers is made when multiple brokers are listed. If this property is enabled, connection occurs sequentially. If disabled, connection occurs in a random manner.
Fault Tolerant	Determines whether the listener is connecting to the Sonic Broker as a fault tolerant client or not. The default value is false, that is, the listener is not connecting as a fault tolerant client. This feature is supported only when the Sonic Broker is installed with a Sonic Fault Tolerance license code.

Property Name	Property Description
Reconnect Fault Timeout	Sets the interval, in seconds, of how long to wait before attempting to reconnect to the broker if the session is lost.
Initial Connect Timeout	Sets the interval, in seconds, of how long to wait before attempting to connect to the broker, or to another broker on the list, if connection fails when the listener is first started.
Accepts non-XML (flat) only	The select if non-XML input is expected. If enabled, XML input still can be passed to the listener. Preparsers do not run.
Optimize Favoring	Use this option to customize listener performance. For smaller transactions, select *performance*. For large payloads that could monopolize the amount of memory used by Service Manager, select *memory*.
Multithreading	The number of worker threads. (Equivalent to the number of requests that Service Manager can handle in parallel.) Setting this to a value of greater than 1 enables the listener to handle a second request while an earlier request is being processed. The total throughput of a system can depend on the number of threads operating. Default: 1 Max value: 99
Execution Time Limit	The maximum time a request can take to complete. A request that takes longer to complete terminates. Prevents runaway requests. **Note:** The kill interval property checks for runaway requests that exceed their maximum life. Default is 60 (seconds). Duration is xxhxxmxxs, for example, 1h2m3s = one hour, two minutes, and three seconds.
Polling Interval	Indicates frequency in seconds that the listener returns control to Service Manager to determine if a stop listener was requested. The listener is constantly connected to the queue to retrieve incoming messages. The default value is 2.0.
Default Java File Encoding	The default encoding if incoming message is not self-declaring (that is, XML).

Property Name	Property Description
Agent Precedence	Sets the order by which Service Manager selects agents. Service Manager selects the agent or agents to process the document by searching through the configuration dictionary. Usually, it looks for a document entry in the configuration and when a match is found, the agent specified in that document entry is selected. If a matching document entry is not found or no agent is specified, the engine looks in the input protocol configuration (listener). To have the processing agent taken directly from the listener (thus ignoring the document entry), use <listener> overrides <document>. <document> overrides <listener> is the default value.
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined destinations.
Error Documents treated normally	If set to *true*, error documents are processed by configured preemitters.
Listener is Transaction Manager	If set to *true*, agents run in a local transaction. Agents can roll back uncompleted transactions.
Initialization Agent	The name (parameters) of the processing module called at listener start up.

Note: The Sonic listener supports streaming. Streaming is used for large documents or documents for which the application needs to split the input into sections under the same transaction. For more information on streaming and configuring streaming preparsers, see the *iWay Service Manager Component and Functional Language Reference Guide*.

Configuring a Sonic Listener Using HTTPS

To configure the Sonic listener for SSL (the user ID or certificate depending on how the broker is configured), modify the destination address to specify an HTTPS URL.

The format of the destination property is queue@url. When emitting to Sonic queues over HTTPS, the format is queue@https://host:port. The HTTP port is configured in the Sonic broker.ini file. To send a message to iWay.Reply hosted on a Sonic broker on your localhost, the URL is iWay.Reply@https://localhost:2507.

For instructions on creating a listener, see *Configuring Listeners* on page 18.

iWay Adapter for Sonic Emitter Functionality

To route an output document or error message to a protocol other than that of the listener, you must configure an emitter. An emitter sends documents outbound either on the same protocol that the document arrived on or across protocols. You can return a processed document to one or more alternate destinations. By default, an output document is returned using the same protocol on which it was received. For example, an application may send input over TCP but want to route the output to a Sonic queue. In this case, you would configure an emitter.

Note: Configuring a Sonic emitter is not required if the emitter protocol used in the outlet of the channel is the same as the listener protocol used in the inlet of the channel. For more information on inlets and outlets, see *Defining an Inlet* on page 15 and *Defining an Outlet* on page 25.

Sonic supports standard Internet protocols including Secure Socket Layer (SSL), HTTP, encryption, TCP, and security. Service Manager can send responses to Sonic queues through TCP, SSL, HTTP, and HTTPS protocols. Each protocol requires specific set up. This topic describes configuration information for each protocol.

Configuring a Sonic Emitter Using TCP or HTTP

The Transport Control Protocol (TCP) is a connection-oriented protocol that establishes a connection with another host before it sends its data. Before a connection is started between two hosts, control messages called a handshake are sent to initiate the connection. TCP is the default protocol of a Sonic broker installation. Client applications that are Internet-enabled generally use TCP/IP protocols.

Hypertext Transfer Protocol (HTTP), the underlying protocol used by the World Wide Web, defines how messages are formatted and transmitted, and the actions web servers and browsers must take in response to various commands.

HTTP operates over TCP connections. After a successful connection, the client transmits a request message to the server, which sends a reply message back.

By server design or by company security policy, proxy servers and firewalls frequently allow only HTTP-based traffic to pass through. You can establish a direct connection to a Sonic broker using HTTP tunneling as the protocol. However, because the HTTP tunneling protocol is significantly slower than TCP or SSL, this option is only recommended when TCP and SSL protocols are not available.

To emit messages to Sonic, JAR files supplied by Sonic are required. For more information, see *Registering Sonic Client JAR Files* on page 264.

Sonic Emitter Properties for TCP or HTTP

The following table lists and describes the Sonic emitter properties. For instructions on creating an emitter, see *How to Create an Emitter* on page 29.

A Sonic message consists of a JMSQ header, body, and user-defined properties.

Property Name	Property Description
Destination (required)	The destination to which the message is delivered. The format of the destination property is queue@url. When emitting to a Sonic broker configured to listen over TCP, the format is queue@tcp://host:port. The default TCP port on which Sonic listens is 2506. This value is configured in the Sonic broker.ini file. To send a message to iWay.Reply hosted on a Sonic broker on your localhost, the URL is iWay.Reply@tcp://localhost:2506. For HTTP, the URL is in the format queue@http://host:port. The default HTTP port on which Sonic listens is 2580. To send a message to iWay.Reply hosted on a Sonic broker on your localhost, the URL is iWay.Reply@http://localhost:2580

Form of Output	`TOPIC`
	Is for a TopicSession in the Publish and Subscribe domain.
	`QUEUE`
	Is for a QueueSession in the PTP domain.

User	A valid user ID defined to Sonic. You can use the Sonic Explorer Users option (under the message broker) to display users defined to Sonic. User ID is only required if Sonic security is enabled.
Password	A valid user ID and password combination defined to Sonic. You can use the Sonic Explorer Users option (under message broker) to display users defined to Sonic.
Send Persistently	Support for persistent and non-persistent messages. In the event of a network or system failure, the persistent option prevents messages from being lost. In the event of a broker or Service Manager failure, non-persistent messages are volatile. Persistent messages are saved to disk.

Property Name	Property Description
Load Balance	Select *true* to enable the load balancing option on the Service Manager listener. With load balancing enabled, a connect request can be redirected to another broker within a Sonic cluster, if load balancing was not disabled on the broker side.
Duplicates Detect	You can configure the broker to commit transactions that index a universally unique, 64-character identifier (UUID) supplied by the sender. The sender then uses a commit method that commits the transacted messages, unless a previously sent transaction identifier is still unexpired. Otherwise, the method forces a rollback of the transaction. For more information, see the INDEXED_TXN_ server properties in the *Installation* section of the *Sonic MQ V5 Configuration and Administration Guide*.
Set Reply Correlation Id	If set to a value, that value is used as the correlation ID of the response.
Duration	The maximum time that a document can remain in the retry pending queue.
Message Priority	The Sonic broker attempts to deliver messages with a higher priority ahead of messages with lower values. Priority is suggested to the JMSQ provider and can only order delivery of undelivered messages. JMSQ defines ten levels of message priority with values 0 through 9, where 0 is the lowest and 9 is the highest. Zero through four are considered normal settings and five through nine, expedited. The Message Priority field is the default priority value set in the JMSQ header.
Output Message Type	Select *Bytes*, *Text*, or *Dynamic*.
Preserve Undelivered	Preserves messages that are not retrieved to a dead letter queue.
Notify Undelivered	Notifies the broker administrator if undelivered are preserved.

Property Name	Property Description
Sequential	Determines the order in which the connection to the brokers is made when multiple brokers are listed. If this property is enabled, connection occurs sequentially. If disabled, connection occurs in a random manner.
Fault Tolerant	Determines whether the listener is connecting to the Sonic Broker as a fault tolerant client or not. The default value is false, that is, the listener is not connecting as a fault tolerant client. This feature is supported only when the Sonic Broker is installed with a Sonic Fault Tolerance license code.
Reconnect Fault Timeout	Sets the interval, in seconds, of how long to wait before attempting to reconnect to the broker if the session is lost.
Initial Connect Timeout	Sets the interval, in seconds, of how long to wait before attempting to connect to the broker, or to another broker on the list, if connection fails when the listener is first started.

Sonic Emitter Configuration Example

You would like to read a file from its local file system and place it as a message on iWay.Reply queue defined at the Sonic broker, which listens on default Sonic MQ port on a machine named iwxfoc. The broker does not have security enabled.

To configure the components needed to support this scenario, you would need to do the following in Service Manager:

Note: For information on the steps required to accomplish these tasks, see the *iWay Service Manager User's Guide*.

- Copy the default file1 channel that is provided with the iSM installation and rename it to sonic_emit_channel.
- Create an emitter, named sonic_emit, that will place a message obtained from a file listener on iWay.Reply queue defined at the Sonic broker on iwxfoc.

  The sonic_emit_channel will use the default file1 inlet, which is actually a file listener also named file1. In addition, it will use an outlet that contains the sonic_emit emitter.
- Assign the emitter as an outlet of the channel.

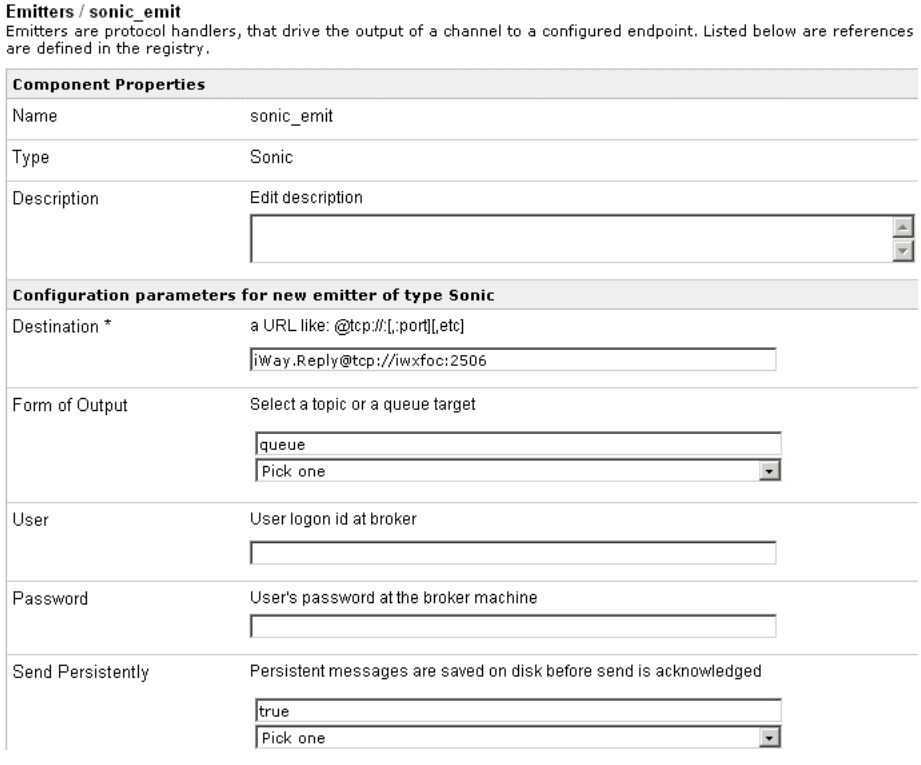
After building and deploying channel you can see the file dropped into file1 pickup folder being placed on iWay.Reply queue on iwxfoc Sonic MQ broker.

Specify the emitter properties as follows:

- **Destination:** iWay.Reply@tcp://iwxfoc:2506
- **Form of Output:** Queue

  Leave default values for the remaining properties.

The following image shows the Sonic emitter configuration pane.



Configuring Sonic Emitter Using SSL

SSL provides authentication and encryption techniques that require that the broker configure appropriate properties and certificates on a security-enabled database. Also, the client must have the appropriate libraries and properties to call its side of an SSL connection. A complete implementation sample is provided in the protocol section of the *Sonic MQ V5 Configuration and Administration Guide*. You must have SSL libraries installed from RSA, with the appropriate Sonic license or from supported third-parties, such as the Institute for Applied Information Processing and Communications (IAIK) or Java Secure Socket Extension (JSSE).

The manner in which you configure SSL for your application depends on whether you implement client authentication with a user name and password or with a client certificate. When the Sonic broker is configured with the SSL property SSL_CLIENT_AUTHENTICATION=FALSE in the broker.ini configuration file, no client certificate is required, and the client is authenticated with a user name and password. The client reads the user name and password, then passes them to the broker.

When the Sonic broker is configured with the SSL property SSL_CLIENT_AUTHENTICATION=TRUE, authentication is achieved through the exchange of Digital Certificates between Service Manager and the Sonic broker. In addition to this mutual authentication, Service Manager can present a user name and password to the broker at run time (optional).

JAR files supplied by Sonic are required to emit messages to a Sonic message queue using the SSL protocol. For more information, see *Registering Sonic Client JAR Files* on page 264.

Reference: JVM Options for SSL With Certificates

The following table lists and describes the properties required to configure the appropriate environment for communication to Sonic using SSL with certificates. For instructions on configuring JVM options, see *Setting Java System Properties for Sonic SSL* on page 276.

Property	Description

`SSL_CA_CERTIFICATES_DIR`
`SSL_CA_CERTIFICATES_DIR={certs/ca|path}`
`SSL_CA_CERTIFICATES_DIR_n={certs/ca|path}`
	Specifies the location of the Certificate Authority (CA) certificate. Path must be fully qualified or relative to the Service Manager installation directory.
	
	The default directory is certs/ca.

Property	Description
SSL_CERTIFICATE_CHAIN SSL_CERTIFICATE_CHAIN={certs/server.p7c|path} SSL_CERTIFICATE_CHAIN_n={certs/server.p7c|path}	Specifies the location of the file containing the client keystore certificate chain for SSL. Path must be fully qualified or relative to the Service Manager installation directory. The default directory is certs/server.p7c.
SSL_CERTIFICATE_CHAIN_FORM SSL_CERTIFICATE_CHAIN_FORM={LIST|PKCS12|PKCS7} SSL_CERTIFICATE_CHAIN_FORM_n={LIST|PKCS12|PKCS7}	Specifies the format of the file containing the certificate chain. PKCS7 is the default value, a comma-delimited list of path names that point to files containing each individual certificate in the chain.
SSL_PRIVATE_KEY SSL_PRIVATE_KEY={serverKey.pkcs8|path} SSL_PRIVATE_KEY_n={serverKey.pkcs8|path}	Provides the location of the file containing the client encrypted private key for SSL. Path must be fully qualified or relative to the Service Manager installation directory. The default value is serverKey.pkcs8.
SSL_PRIVATE_KEY_PASSWORD SSL_PRIVATE_KEY_PASSWORD={password|password} SSL_PRIVATE_KEY_PASSWORD_n={password|password}	Provides the password that encrypts the private key for SSL. The default value is password.

Sonic Emitter Properties for SSL With Certificate

The following table lists and describes the Sonic emitter properties for SSL with certificate. For instructions on creating an emitter, see *How to Create an Emitter* on page 29.

A Sonic message consists of a JMSQ header, body, and user-defined properties.

HTTPS is similar to HTTP except that data is transmitted over a Secure Socket Layer (SSL) instead of a normal socket connection. Web servers listen for HTTP requests on one port while another listens for HTTPS requests.

Property Name	Property Description
Destination (required)	The destination to which the message is delivered. The format of the destination property is queue@url. When emitting to Sonic queues over SSL, the format is queue@ssl://host:port. The SSL port is configured in the Sonic broker.ini file. To send a message to iWay.Reply hosted on a Sonic broker on your localhost, the URL is iWay.Reply@ssl://localhost:2507.

Form of Output	`TOPIC`
	Is for a TopicSession in the Publish and Subscribe domain.
	`QUEUE`
	Is for a QueueSession in the PTP domain.

User	You must specify the common name AUTHENTICATED from the certificate as the user name. The password is not required because you authenticate the client using the client certificate in this example. When Service Manager emits to a Sonic queue it can specify a user name and password to the broker (optional). The broker authenticates this user name and password if they are specified, but otherwise uses the information in the client certificate to identify the user.
Password	This property is optional and is only required if a user ID is specified.
Send Persistently	The support for persistent and non-persistent messages. In the event of a network or system failure, the persistent option prevents messages from being lost. In the event of a broker or Service Manager failure, non-persistent messages are volatile. Persistent messages are saved to disk.
Load Balance	If set to *true*, then this enables the load balancing option on the Service Manager listener. With load balancing enabled, a connect request can be redirected to another broker within a Sonic cluster, if load balancing was not disabled on the broker side.

Property Name	Property Description
Duplicates Detect	You can configure the broker to commit transactions that index a universally unique, 64-character identifier (UUID) supplied by the sender. The sender then uses a commit method that commits the transacted messages, unless a previously sent transaction identifier is still unexpired. Otherwise, the method forces a rollback of the transaction. For more information, see the INDEXED_TXN_ server properties in the Installation section of the *Sonic MQ V5 Configuration and Administration Guide*.
Set Reply Correlation Id	If set to a value, that value is used as the correlation ID of the response.
Duration	The maximum time that a document can remain in the retry pending queue.
Message Priority	The Sonic broker attempts to deliver messages with a higher priority ahead of messages with lower values. Priority is suggested to the JMSQ provider and can only order delivery of undelivered messages. JMSQ defines ten levels of message priority with values 0 through 9, where 0 is the lowest and 9 is the highest. Zero through four are considered normal settings and five through nine, expedited. The Message Priority field is the default priority value set in the JMSQ header.
Output Message Type	Select *Bytes*, *Text*, or *Dynamic*.
Preserve Undelivered	Preserves messages that are not retrieved to a dead letter queue.
Notify Undelivered	Notifies the broker administrator if undelivered are preserved.
Sequential	Determines the order in which the connection to the brokers is made when multiple brokers are listed. If this property is enabled, connection occurs sequentially. If disabled, connection occurs in a random manner.

Property Name	Property Description
Fault Tolerant	Determines whether the listener is connecting to the Sonic Broker as a fault tolerant client or not. The default value is false, that is, the listener is not connecting as a fault tolerant client. This feature is supported only when the Sonic Broker is installed with a Sonic Fault Tolerance license code.
Reconnect Fault Timeout	Sets the interval, in seconds, of how long to wait before attempting to reconnect to the broker if the session is lost.
Initial Connect Timeout	Sets the interval, in seconds, of how long to wait before attempting to connect to the broker, or to another broker on the list, if connection fails when the listener is first started.

Procedure: How to Configure Sonic Emitter Properties Using HTTPS

To configure Sonic emitter properties using HTTPS:

1. Refer to the values defined in *Sonic Emitter Properties for SSL With Certificate* on page 301 (user ID or certificate, depending on how the broker is configured) when creating the emitter.
2. Modify the destination address to specify an HTTPS URL.

   The format of the destination property is queue@url. When emitting to Sonic queues over HTTPS, the format is queue@https://host:port. The HTTP port is configured in the Sonic broker.ini file. To send a message to iWay.Reply hosted on a Sonic broker on your localhost, the URL is iWay.Reply@https://localhost:2507.

Sonic Message Queuing Troubleshooting

When using a Sonic listener, if the system cannot find the queue specified, you receive the following message:

```
Cannot emit reply to .XDSonicEmit<no dead letter path>: XD[FAIL] in emit
() error create queue.
```

Using Sonic Explorer, verify that the queue exists, that the name is spelled correctly, and that the name is in the correct case, as the Sonic listener is case-sensitive.

TIBCO Rendezvous

TIBCO Rendezvous is the messaging system that is the foundation of TIBCO ActiveEnterprise. TIBCO Rendezvous delivers true real-time publish/subscribe and request/reply messaging. It also supports qualities of service ranging from lightweight informational message handling to certified transactional delivery.

TIBCO Rendezvous uses a distributed architecture to eliminate bottlenecks and single points of failure. Applications can select from several qualities of service including reliable, certified, and transactional, as appropriate for each interaction.

Messaging can be publish/subscribe or request/reply, synchronous, or asynchronous, locally delivered, or sent using WAN or the Internet. TIBCO Rendezvous messages are self-describing and platform independent, with a user-extensible type system that provides support for data formats such as XML.

Queuing Messages With TIBCO Rendezvous

TIBCO Rendezvous software uses subject-based addressing technology to direct messages to their destinations, so program processes can communicate without knowing the details of network addresses or connections. Subject-based addressing conventions define a uniform name space for messages and their destinations.

The locations of component processes become entirely transparent. Any application component can run on any network host without modifying, recompiling, or reconfiguring. Application programs migrate easily among host computers. You can dynamically add, remove, and modify components of a distributed system without affecting other components.

Subject names consist of one or more elements separated by dot characters (periods). The elements can implement a subject name hierarchy that reflects the structure of information in an application system.

The following strings are examples of valid subject names:

- RUN.HOME
- RUN.for.Elected_office.President

For more information on subject-based addressing, see the documentation for TIBCO Rendezvous.

The server provides a TIBCO Rendezvous listener, supporting XML messages to and from the server system on TIBCO Rendezvous queues. Service Manager can accept messages arriving on a named queue and route these messages to either another queue or to a non-TIBCO Rendezvous destination.

Similarly, messages received through any of the other Service Manager listeners can be directed to a TIBCO Rendezvous queue. During the message flow, Service Manager can apply the standard document analysis, validation, transformation, and business logic capabilities to the document.

TIBCO Rendezvous support accepts and processes messages arriving at Service Manager on a TIBCO Rendezvous queue, regardless of the originator of the message. As messages arrive, they are accepted by Service Manager and processed.

Service Manager can also emit a message (in either XML or non-XML formats, for example, HIPAA, SWIFT, or HL7) to a TIBCO Rendezvous queue. You can do this regardless of the inbound protocol of the message.

The following topics describe how to configure the TIBCO Rendezvous network listener and emitter.

Registering TIBCO JAR Files

The TIBCO listener requires that you register the following jar file:

For the steps required to register the file, see *Registering Library Files and Setting JVM Options* on page 40.

Type	JAR File
TIBCO Listener	Tibrvj.jar is an archive file containing the Rendezvous classes JAR file located at [Tibrv home]/lib/tibrvj.jar. Specifies the Rendezvous shared library files, for example, C:\libraries (the directory in which the shared libraries are installed) in the Additions to the System Path section. You can find it at [tibco rendezvous home]\bin.

The Additions to the System Path facility amends the path that is used at run time. Adding to the path enables a Java class to load a DLL (for example, tibrvj.dll) that is not already on the system path.

However, if the DLL loaded from Java requests another DLL, that search appears to be on the original path and fails, making it look as if the original load failed. The directory that contains the secondary DLL must also be on the path.

For instructions on registering JAR files, see *Registering Library Files and Setting JVM Options* on page 40.

TIBCO Listener Properties

The following table lists and describes the TIBCO listener properties. For instructions on creating a listener, see *How to Create a Listener* on page 18.

Property Name	Property Description
Event Queue Name	If this property is not specified, the listener listens on the TIBCO Rendezvous default queue.
Service Name	UDP service on which to listen for TIBCO Rendezvous messages. This property accepts a service name or a port number. The default TIBCO Rendezvous port during installation is 7500. If service, network, or daemon are not present, the listener attempts to connect to a local instance of running TIBCO Rendezvous service. Otherwise, specify the TIBCO Rendezvous instance in the form host:<port>.
Network	The network property instructing the Rendezvous daemon to use a particular network for all communications involving this transport. This property may be a host name, IP address, or network name. For more information about this property, see the TIBCO Rendezvous administration manual. If the daemon is running on the local host, this property may be omitted.
Daemon	RV Daemon information to find the TIBCO Rendezvous daemon and establish a connection.
Send Subject	Each Rendezvous message bears a subject name. The subject name is used by data consumers to receive all messages labeled with a given name. If the property is left blank, the TIBCO Rendezvous listener listens on subject *. For more information on the send subject property, see the Subject Names section in the *TIBCO Rendezvous Concepts* manual.
Reply Subject	The return subject of the message.
Field Name	The custom field name on which the listener filters.

Property Name	Property Description
Data Type	You can select the type of data for emission. The choices are opaque, string, TibrvMsg, or TibrvXml.
Normalize To Decimal	Normalizes data from scientific to decimal notation.
Add Data Type Info	Adds an attribute of data type, the value of which is the field type name.
Accepts non-XML (flat) only	If set to *true*, non-XML input is expected. If enabled, XML input still can be passed to the listener. Preparsers do not run.
Optimize Favoring	Use this option to customize listener performance. For smaller transactions, select *performance*. For large payloads that could monopolize the amount of memory used by Service Manager, select *memory*.
Multithreading	Indicates the number of worker threads. (Equivalent to the number of requests that Service Manager can handle in parallel.) Setting this to a value of greater than 1 enables the listener to handle a second request while an earlier request is being processed. The total throughput of a system is affected by the number of threads operating. Default: 1 Max value: 99
Execution Time Limit	The maximum time a request can take to complete. A request that takes longer to complete terminates. Prevents runaway requests. See also the description of the system property, Kill interval.
Whitespace Normalization	Specifies how the parser treats whitespace in Element content. Choose *preserve* to turn off all normalization as prescribed by the XML Specification. Choose *condense* to remove extra whitespace in pretty printed documents and for compatibility with earlier versions.
Polling Interval	Indicates the frequency (in seconds) that the listener returns control to Service Manager to determine if a stop listener was requested. The listener is constantly connected to the queue to retrieve incoming messages. The default value is 2.0.

Property Name	Property Description
Default Java File Encoding	The default encoding if incoming message is not self-declaratory that is, XML.
Agent Precedence	Sets the order by which Service Manager selects the agent or agents to process the document. It does this by looking for a document entry in the configuration dictionary and when a match is found, the agent specified in that document entry is selected. If no matching document entry is found or no agent is specified, the engine looks in the input protocol configuration (listener). To have the processing agent taken directly from the listener (thus ignoring the document entry), use <listener> overrides <document>. Possible values are <document> overrides <listener> and <listener> overrides <document>. <document> overrides <listener> is the default value.
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined destinations.
Error Documents treated normally	If set to *true*, error documents are processed by configured preemitters.
Listener is Transaction Manager	If set to *true*, agents run in a local transaction. Agents can roll back uncompleted transactions.
Record in Activity Log(s)	If set to *true*, activity on this channel will be recorded in the activity logs, else the activity will not be recorded.

Reference: TIBCO Listener Special Registers

The following table lists and describes the special registers (SREGs) available on the TIBCO listener.

Name	Source	Level	Type	Description
iwayconfig	Listener	System	String	The current active configuration name.

Name	Source	Level	Type	Description
msgsize	Listener	Document	Integer	The physical length of the message payload.
name	Listener	System	String	The assigned name of the master (listener).
protocol	Listener	System	String	The protocol on which this message was received.
tid	Listener	Document	String	Unique transaction ID.

Configuring a TIBCO Rendezvous Emitter

Messages are sent to the destinations at the completion of a workflow. The state of the document determines which destination is used. The order in which the destinations are used cannot be predicted. Destinations cannot accept responses. These configurations also are used to send a message over a different protocol.

Note: Configuring a TIBCO Rendezvous emitter is not required if the emitter protocol used in the outlet of the channel is the same as the listener protocol used in the inlet of the channel. For more information on inlets and outlets, see *Defining an Inlet* on page 15 and *Defining an Outlet* on page 25.

To route an output document or error message to a protocol other than that of the listener, you must configure an emitter. For example, an application may send input over TCP but want to route the output to an TIBCO Rendezvous queue.

TIBCO Rendezvous Emitter Properties

The following table lists and describes the TIBCO Rendezvous emitter properties. For instructions on creating an emitter, see *How to Create an Emitter* on page 29.

Property Name	Property Description
Destination (required)	Host:port on which the RVD daemon is listening.

Property Name	Property Description
Send Subject	Each Rendezvous messages has a subject name. The subject name is used by data consumers to receive all messages labeled with a given name. If the property is left blank, the iWay TIB/Rendezvous emitter defaults the send subject for the TIB/Rendezvous message to IWAYDEFAULTSUBJECT.
Reply Subject	The return address, to which recipients can send reply messages. If this property is omitted, no default value is supplied.
Field Name	Character string. Each field can have a maximum of one name. Several fields can have the same name. If the field name property is not specified, the TIBCO Rendezvous message with which the payload is associated is emitted with the default field name of IWAYDEFAULTOBJECT.
Service Name	The UDP service to use whenever TIBCO Rendezvous conveys messages (service name or port number).
Network	The network for network transport objects to communicate with other transport objects.
Data Type	Type of data for emission. Select one of the following: ❑ Opaque ❑ String ❑ TibrvMsg ❑ TibrvXml

TIBCO Rendezvous Troubleshooting

The following table describes an error that you may encounter when using a TIBCO Rendezvous listener.

Error	Reason	Solution
`Failed to open Tibrv in native implementation: TibrvException [error=901, message=Library not found: tibrvj]]: [line 3]`	The system path was not updated to include the shared library directory.	Use the Path Settings pane to register %TibcoRV home\bin in the Path field. For this to take effect, the server must be stopped and restarted. Another reason for this error is that although the libraries were included using this pane, the server was not stopped and started again to include the libraries in the configuration. Restarting the server is not sufficient.

RabbitMQ

RabbitMQ is open source message broker software that implements version 0-9-1 of the Advanced Message Queuing Protocol (AMQP). The listener supports AMQP version 0-9-1 exclusively since that is the version supported natively by RabbitMQ.

Note: AMQP version 1.0 is a newer version, but it is not compatible and should be considered a different protocol, despite the similar name.

iWay Service Manager (iSM) provides native bi-directional support for communication with RabbitMQ to read/write messages. This does not require any additional layer for communication such as Java Message Service (JMS), which can also be used as an alternative, but does have its limitations.

Configuring the RabbitMQ Listener (com.ibi.edaqm.XDRabbitMQMaster)

Syntax:

```
com.ibi.edaqm.XDRabbitMQMaster
```

Description:

The RabbitMQ listener uses the Java client library to communicate with the RabbitMQ server. The required third-party library is automatically installed with the listener. There are no extra installation steps that are required.

Each worker runs a consumer thread to accept messages from the designated queue. After a message is processed, a response can optionally be sent. The listener always reads directly from a queue. If applicable, the response is sent to an exchange. The exchange will route the message to the appropriate queues.

The listener expects the queue and exchange to be pre-configured in the RabbitMQ server. The listener does not declare those objects automatically.

Parameters:

The following table describes the parameters of the RabbitMQ Listener (com.ibi.edaqm.XDRabbitMQMaster).

Parameter	Description
Broker URI *	The Broker URI, in the following format: *scheme*://host:*port*/virtualhost where: *scheme* Is AMQP or AMQPS. *port* Is optional defaulting to 5672 for AMQP and 5671 for AMQPS. *virtualhost* Is also optional.
SSL Context Provider	The iWay Security Provider for SSL Context. If you are using AMQPS and SSL Context Provider is left blank, then the default provider will be used.
User *	User ID for the Broker.
Password *	Password that is associated with the user ID for the Broker.
Queue Name *	Messages will be consumed from this queue.

Parameter	Description
Auto Acknowledgement	Determines whether the acknowledgement is sent automatically or is delayed until the message is processed. By default, this parameter is set to *false*.
Request Header Namespace	The special register namespace into which protocol headers from the incoming request will be saved. By default, this parameter is set to *Default Namespace*.
Default Reply	
Outgoing Exchange Name	The exchange where the outgoing message will be published.
Routing Key	The routing key of the outgoing message.
Mandatory	Determines what happens when a message is routed to zero queues. When set to *true*, an error is returned. When set to *false*, the message is discarded. By default, this parameter is set to *false*.
App ID	Creating application ID.
Content Encoding	MIME Content Encoding of the payload.
Content Type	MIME Content Type of the payload.
Correlation ID	Application correlation identifier.
Delivery Mode	Determines whether the message is persistent (2) or non-persistent (1). By default, this parameter is set to *1* (persistent).
Expiration	The message time to live in the queue, non-negative integral number of milliseconds.
Message ID	Application message identifier.
Priority	Message priority (0 to 255).
Response Reply To	Destination to reply to this response.

Parameter	Description
Timestamp	The value of the timestamp property in the outgoing message, in milliseconds since the epoch.
Type	Message type name.
User ID	Creating user ID.
Response Header Namespace	The special register namespace from which protocol headers for the outbound response will be taken. By default, this parameter is set to *Default Namespace*.
Tuning	
Multithreading	Indicates the number of worker threads (documents or requests) that iWay Service Manager can handle in parallel. Setting this to a value of greater than 1 enables the listener to handle a second request while an earlier request is still being processed. The total throughput of a system can be affected by the number of threads operating. Increasing the number of parallel operations may not necessarily improve throughput. The default is 1. The max value is 99.
Optimize Favoring	Use this option to customize how the listener performs. For smaller transactions, select *performance*. For large input documents that could monopolize the amount of memory used by iWay Service Manager, select *memory*.
Events	
Failed ReplyTo Flow	Name of a published process flow to run if a message cannot be emitted on an address in its reply address list.
Dead Letter Flow	Name of a published process flow to run if an error cannot be emitted on an address in its error address list.

Parameter	Description
Channel Failure Flow	Name of a published process flow to run if this channel cannot start or fails during message handling. iWay Service Manager will attempt to call this process flow during channel shut down due to the error.
Parse Failure Flow	Name of published process flow to run if XML parsing fails for incoming message.
Channel Startup Flow	Name of published process flow to run prior to starting the channel.
Channel Shutdown Flow	Name of published process flow to run when the channel is shut down.
Other	
Whitespace Normalization	Specifies how the parser treats whitespace in Element content. Choose *preserve* to turn off all normalization as prescribed by the XML Specification. Choose *condense* to remove extra whitespace in pretty printed documents and for compatibility with earlier versions.
Accepts non-XML (flat) only	If set to *true*, the input data is sent directly to the business logic step. The data is not preparsed, parsed, or validated. This flag is used primarily to send non-XML to the business logic or replyTo without processing it.
Execution Time Limit	The maximum time that a request may take to complete. Used to prevent runaway requests. Any request that takes longer to complete than this value will be attempted to be terminated.
Default Java File Encoding	The default encoding if incoming message is not self-declaring (that is, XML).

Parameter	Description
Agent Precedence	Sets the order by which iWay Service Manager selects agents. iWay Service Manager selects the agent or agents to process the document by searching through the configuration dictionary. Usually, it looks for a document entry in the configuration and when a match is found, the agent specified in that document entry is selected. If a matching document entry is not found, or no agent is specified, the engine looks in the input protocol configuration (listener). To have the processing agent taken directly from the listener (thus ignoring the document entry), use <listener> overrides <document>. Possible values are <document> overrides <listener> and <listener> overrides <document>. The default value is <document> overrides <listener>.
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined reply-to and error-to destinations.
Error Documents treated normally	If set to *true*, error documents are processed by any configured preemitters.
Listener is Transaction Manager	If set to *true*, agents run within a local transaction.
Record in Activity Log(s)	If set to *true*, activity on this channel will be recorded in the activity logs, otherwise the activity will not be recorded.
AES Key	If the channel will receive encrypted AFTI messages, set the AES key (maximum 16 characters) to be used for decrypting.
Startup Dependencies	A comma-separated list of channel names that must be started before this one is called.

The Broker URI has the following format:

```
scheme://host:port/virtualhost
```

The scheme is AMQP for communication in clear with the port defaulting to 5672. For secure communication over TLS, the scheme is AMQPS with the port defaulting to 5671. The virtual host is a RabbitMQ concept that allows multiple configurations for the same RabbitMQ server. When /virtualhost is not specified, the virtual host defaults to /. The syntax `scheme://host:port` is the only way to specify / as the virtual host, since ending the URI with / is not allowed.

When using AMQPS, the TLS configuration is specified with an SSL Context provider. This provider is created in the iSM Administration Console within the Security Provider page under the Server tab.

The user name and password are the credentials to access the RabbitMQ server. The user is created in the RabbitMQ console within the Users page under the Admin tab. Make sure the user is given permission to access the chosen virtual host. The RabbitMQ console can usually be accessed at:

```
http://host:15672
```

The *Auto Acknowledgement* parameter controls whether the acknowledgement is sent automatically or is delayed until the message is processed.

If there are no other replies configured, then the default reply will be sent. The destination of the default reply is controlled by the *Outgoing Exchange Name* parameter with the value specified by the Routing Key. If the *Outgoing Exchange Name* parameter has no value specified, then it defaults to the *amqp.replyTo* special register (SREG). The *Mandatory* parameter determines what happens when the message is routed to zero queues. An error is returned if the *Mandatory* parameter is set to *true*, otherwise the message is silently discarded. User-configured headers can be added by specifying a value for the *Response Header Namespace* parameter. Each register of type HDR in that namespace will create a header of the same name.

The properties in the Default Reply group support deferred evaluation (except for Response Header Namespace). This means prepending the expression with a backtick character (`) causes the expression to be evaluated just before the reply is sent. This makes it possible to get different values for each reply, which might depend on the input message. For example, the default value of the *Outgoing Exchange Name* parameter can be emulated with the following expression:

```
`_SREG(amqp.replyTo)
```

If the backtick character is absent, then the parameter is evaluated once when the listener is initialized, just like the other parameters.

Special Registers (SREGs)

There is a fixed list of message properties as documented in the AMQP specification. The service stores these properties in Special Registers (SREGs) in the AMQP namespace.

The following table describes the SREGs assigned by the RabbitMQ Read service based on the properties in the message envelope.

SREG Name	Description
amqp.deliveryTag	The server-assigned and channel-specific delivery tag.
amqp.exchange	The name of the exchange that routed this message.
amqp.routingKey	The key used to route this message.
amqp.isRedeliver	The boolean redelivery flag.

The following table describes the special registers assigned by the RabbitMQ Read service based on the properties in the message.

SREG Name	Description
amqp.appId	Creating application ID.
amqp.contentEncoding	MIME content encoding.
amqp.contentType	MIME content type.
amqp.correlationId	Application correlation identifier.
amqp.deliveryMode	A value of 1 indicates non-persistent. A value of 2 indicates persistent.
amqp.exiration	Message time to live in the queue, integer >= 0 in milliseconds.
amqp.messageId	Application message identifier.
amqp.priority	Message priority, 0 to 255.
amqp.replyTo	Destination to reply to.
amqp.timestamp	Message timestamp, number of seconds since the epoch.

SREG Name	Description
amqp.type	Message type name.
amqp.userId	Creating user ID.

Unlike message properties, message headers are not standardized. They are under the sending application control. When present, message headers are stored as special registers in the specified Request Header Namespace. For example, if a header is named *hdr1* and the Request Header Namespace is *ns*, then the special register will be called *ns.hdr1*.

SSL Configuration

The initial configuration of RabbitMQ declares a regular TCP listener listening on port 5672. The configuration file must be edited if an SSL listener is required. On Windows, the location of the configuration file is:

```
%APPDATA%\RabbitMQ\rabbitmq.confi
```

Here is a sample configuration with an SSL listener listening on port 5671 and no TCP listeners.

```
[
  {rabbit, [
     {tcp_listeners, []},
     {ssl_listeners, [5671]},
     {ssl_options, [{cacertfile,"C:\\rabbitmq\\ssl\\cacert.pem"},
                    {certfile,"C:\\rabbitmq\\ssl\\cert.pem"},
                    {keyfile,"C:\\rabbitmq\\ssl\\key.pem"},
                    {verify,verify_none},
                    {fail_if_no_peer_cert,false}]}
   ]}
].
```

The Broker URI to access this server has the following format:

```
amqps://hostname:5671/vhost
```

The Java keytool command can be used to extract a certificate from a keystore. The following command should be entered on one line (shown on multiple lines for display purposes only):

```
keytool -exportcert -rfc -file cert.pem -alias myalias -keystore
mystore.p12 -storepass secret -storetype PKCS12
```

Unfortunately, the keytool command does not have an option to export a private key. This can be done with openssl:

```
openssl pkcs12 -in mystore.p12 -out keys.pem -clcerts -nocerts -nodes
```

Edit keys.pem with a text editor to keep only the key you want. Then run the following command:

```
openssl rsa -in keys.pem -out key.pem
```

RabbitMQ Emit Service (com.ibi.agents.XDRabbitMQEmitAgent)

Syntax:

```
com.ibi.agents.XDRabbitMQEmitAgent
```

Description:

This service emits a document to a RabbitMQ server using version 0-9-1 of the Advanced Message Queuing Protocol (AMQP). The Broker URI is the server address. The destination is a combination of the Outgoing Exchange Name and the Routing Key.

The Broker URI has the following format:

```
scheme://host:port/virtualhost
```

The scheme is AMQP for communication in clear with the port defaulting to 5672. For secure communication over TLS, the scheme is AMQPS with the port defaulting to 5671. The virtual host is a RabbitMQ concept that allows multiple configurations for the same RabbitMQ server. When /virtualhost is not specified, the virtual host defaults to /. The syntax `scheme://host:port` is the only way to specify / as the virtual host, since ending the URI with / is not allowed.

When using AMQPS, the TLS configuration is specified with an SSL Context provider. This provider is created in the iSM Administration Console within the Security Provider page under the Server tab.

The user name and password are the credentials to access the RabbitMQ server. The user is created in the RabbitMQ console within the Users page under the Admin tab. Make sure the user is given permission to access the chosen virtual host. The RabbitMQ console can usually be accessed at:

```
http://host:15672
```

In version 0-9-1 of AMQP, the message is not sent directly to a queue. Instead, the sender sends the message to an exchange and associates a Routing Key. It is the responsibility of the exchange to route the message to all the queues with a binding that matches the message and the routing key. The queue bindings are part of the exchange configuration on the server. The matching rule depends on the exchange type as follows.

Exchange Type	Matching Rule
direct	The routing key must equal the binding key.
fanout	The routing key is ignored and the binding always matches.
headers	The routing key is ignored. The binding specifies a header name and the value it must have to match. It is possible to specify multiple headers at the same time, in which case the x match attribute determines if 'any' or 'all' headers must match.
topic	The routing key is matched with wildcards.

The Mandatory property determines what happens when the message does not match any queue bindings and therefore was routed to zero queues. When this happens, an error is returned if mandatory is set, otherwise the message is silently discarded.

The RabbitMQ Emit Service expects the exchange, the queue bindings, and the destination queues to be pre-configured on the server. The service does not declare those objects automatically.

AMQP defines a set of properties that are always part of the message. Most properties are open to interpretation by the receiving application. These are: App ID, Content Encoding, Content Type, Correlation ID, Message ID, Response Reply To, Timestamp, Type and User ID. For example, the RabbitMQ server considers the payload to be a byte string. The receiving application may use the Content Encoding and Content Type to reconstruct the message, but that is just by convention.

The properties that affect the broker are: Delivery Mode, Expiration and Priority. The delivery mode is Persistent or Non-Persistent. Persistent messages held in a durable queue will survive a broker restart.

The Expiration is the time the message can remain in the queue before it expires. Expired messages are either discarded or dead-lettered, but only when they reach the head of the queue. Until then, they occupy storage and are counted in the queue statistics. The Priority affects the order messages are delivered by a priority queue. The AMQP specification says the priority is from 0 to 9 but RabbitMQ supports priorities from 0 to 255.

Unlike message properties, message headers are not standardized. They are under the sending application control. When the Request Header Namespace is specified, every special register of type HDR it contains will create a message header of the same name. For example, if the Request Header Namespace is *ns* and there is special register of type *HDR* called *ns.hdr1* with value *v1*, then a message header called *hdr1* will be created with value *v1*.

The Avoid Preemitter parameter determines whether preemitters will be executed. The default is to skip the preemitters.

The Return Document parameter determines the contents of the output document. Choose status to return a status document, or input to return the input document of the service.

Parameters:

The following table describes the parameters of the RabbitMQ Emit service (com.ibi.agents.XDRabbitMQEmitAgent).

Parameter	Description
Outgoing Exchange Name	The exchange where the outgoing message will be published.
Broker URI *	The Broker URI, in the following format: *scheme*://host:*port*/virtualhost where: *scheme* Is AMQP or AMQPS. *port* Is optional defaulting to 5672 for AMQP and 5671 for AMQPS. *virtualhost* Is also optional.
SSL Context Provider	The iWay Security Provider for SSL Context. If you are using AMQPS and SSL Context Provider is left blank, then the default provider will be used.
User *	User ID for the Broker.

Parameter	Description
Password *	Password that is associated with the user ID for the Broker.
Routing Key	The routing key of the outgoing message.
Mandatory	Determines what happens when a message is routed to zero queues. When set to *true*, an error is returned. When set to *false*, the message is discarded. By default, this parameter is set to *false*.
App ID	Creating application ID.
Content Encoding	MIME Content Encoding of the payload.
Content Type	MIME Content Type of the payload.
Correlation ID	Application correlation identifier.
Delivery Mode	Determines whether the message is persistent (2) or non-persistent (1). By default, this parameter is set to *1* (persistent).
Expiration	The message time to live in the queue, non-negative integral number of milliseconds.
Message ID	Application message identifier.
Priority	Message priority (0 to 255).
Response Reply To	Destination to reply to this response.
Timestamp	The value of the timestamp property in the outgoing message, in milliseconds since the epoch.
Type	Message type name.
User ID	Creating user ID.
Request Header Namespace	The special register namespace from which protocol headers for the outbound request will be taken. By default, this parameter is set to *none*.

Parameter	Description
Avoid Preemitter	Determines whether any preemitter should be avoided. By default, this parameter is set to *true*.
Return Document	Determines whether the output document will be a status document or the input document. By default, this parameter is set to *status*.

Edges:

The following table lists and describes the line edges that are returned by the RabbitMQ Emit service (com.ibi.agents.XDRabbitMQEmitAgent).

Line Edge	Description
success	The message was successfully sent.
fail_parse	An iFL expression could not be evaluated.
fail_connect	The service could not connect to the broker.
fail_operation	The operation could not be completed successfully.

RabbitMQ Read Service (com.ibi.agents.XDRabbitMQReadAgent)

Syntax:

```
com.ibi.agents.XDRabbitMQReadAgent
```

Description:

This service reads a single message from a RabbitMQ queue using version 0-9-1 of the Advanced Message Queuing Protocol (AMQP).

The Broker URI has the following format:

```
scheme://host:port/virtualhost
```

The scheme is AMQP for communication in clear with the port defaulting to 5672. For secure communication over TLS, the scheme is AMQPS with the port defaulting to 5671. The virtual host is a RabbitMQ concept that allows multiple configurations for the same RabbitMQ server. When /virtualhost is not specified, the virtual host defaults to /. The syntax `scheme://host:port` is the only way to specify / as the virtual host, since ending the URI with / is not allowed.

When using AMQPS, the TLS configuration is specified with an SSL Context provider. This provider is created in the iSM Administration Console within the Security Provider page under the Server tab.

The user name and password are the credentials to access the RabbitMQ server. The user is created in the RabbitMQ console within the Users page under the Admin tab. Make sure the user is given permission to access the chosen virtual host. The RabbitMQ console can usually be accessed at:

`http://host:15672`

The Queue Name identifies which queue will be read.

The Timeout is the period to wait in milliseconds for the message to become available. When the timeout expires, the service returns the input document with the edge noMessage. A value of zero (0) indicates to return immediately.

Parameters:

The following table describes the parameters of the RabbitMQ Read service (com.ibi.agents.XDRabbitMQReadAgent).

Parameter	Description
Broker URI *	The Broker URI, in the following format: *scheme*://host:*port*/virtualhost where: *scheme* Is AMQP or AMQPS. *port* Is optional defaulting to 5672 for AMQP and 5671 for AMQPS. *virtualhost* Is also optional.
SSL Context Provider	The iWay Security Provider for SSL Context. If you are using AMQPS and SSL Context Provider is left blank, then the default provider will be used.
User *	User ID for the Broker.
Password *	Password that is associated with the user ID for the Broker.
Queue Name *	The name of the queue from which messages will be consumed.
Request Header Namespace	The special register namespace into which protocol headers from the incoming request will be saved. By default, this parameter is set to *Default Namespace*.
Timeout	Period to wait in milliseconds for the message to become available. A value of zero (0) indicates to return immediately. By default, this parameter is set to *5000*.

Edges:

The following table lists and describes the line edges that are returned by the RabbitMQ Read service (com.ibi.agents.XDRabbitMQReadAgent).

Line Edge	Description
success	The message was successfully sent.
fail_parse	An iFL expression could not be evaluated.
fail_connect	The service could not connect to the broker.
fail_operation	The operation could not be completed successfully.
fail_partner	The operation failed because of an error from the peer.
noMessage	The timeout expired before a message became available.

WebSphere MQ and MQJMS

The IBM WebSphere (formerly MQSeries) messaging products enable application integration by helping business applications to exchange information across different platforms by sending and receiving data as messages.

WebSphere consists of Native MQ and MQSeries JMS (MQJMS). These products handle network interfaces, assure once-only delivery of messages, deal with communication protocols, dynamically distribute workloads across available resources, handle recovery after system problems occur, and help make programs portable.

These actions are performed so that programmers can use their skills to handle key business requirements, instead of wrestling with underlying network complexities.

Note: Throughout this topic, the Native MQ listener is also referred to as WebSphere, MQSeries, or MQ.

Queuing Messages With WebSphere MQ

Using the iWay Adapter for WebSphere, Service Manager can read messages arriving on a named queue and route the responses to either another queue or to a non-MQ destination. Similarly, messages received through any of the other Service Manager listeners can be directed to an MQ queue. During the message flow, Service Manager can apply the standard document analysis, validation, transformation, and business logic capabilities to the message. An optional pending facility ensures that messages are retried if appropriate resources are not available at the time of execution.

Service Manager also supports queue operations using the WebSphere client, so that each user need not install the full IBM WebSphere Server product.

MQSeries Classes for Java Message Service (JMS)

MQSeries classes for Java Message Service is a set of Java classes that implement the Sun Microsystem Java Message Service specification. A JMS application can use the classes to send MQSeries messages to either existing MQSeries or new JMS applications. An application can be configured to connect as an MQSeries client using TCP/IP or directly using the Java Native Interface (JNI). If the client-style connection is used, no additional MQSeries code is required on the client machine.

Use of the MQSeries classes for Java Message Service offers benefits associated with using an open standard to write MQSeries applications, such as the protection of investment both in skills and application code. In addition, the JMS classes provide some additional features not present in the MQSeries classes for Java. These extra features include:

- Explicit support for publish and subscribe.
- Asynchronous message delivery.
- Message selectors.
- Structured message classes.
- Support for XA transactions through the XAResource interface (not available for OS/390 or z/OS).

Registering MQ JAR and DLL Files

The WebSphere listener requires that you register the following files that are in [MQSeries]/java/lib/:

Platform/Notes	JAR Files
JAR (Client-side)	
MQ Version 6	❑ com.ibm.mq.jar ❑ connector.jar

Platform/Notes	JAR Files
MQ Version 7 **Note:** When using MQ Version 7, version 7.0.1.3 and higher is required.	❏ com.ibm.mq.jmqi.jar ❏ com.ibm.mq.jar ❏ com.ibm.mq.headers.jar ❏ com.ibm.mq.commonservices.jar ❏ connector.jar

DLL (Server-side)

Register the DLL directory (this is the directory where the name interface resides), for example,

```
[mqseries home]/java/lib -classpath libpath
```

For some platforms, you must explicitly add HOME/java/lib to the path. For example, for OS/390 you must issue the following command:

```
export LIBPATH=/mqm/java/lib:$LIBPATH
```

For instructions on registering JAR files, see *Registering Library Files and Setting JVM Options* on page 40.

WebSphere MQ Listener Properties

The following table lists and describes the WebSphere MQ listener properties. For instructions on creating a listener, see *How to Create a Listener* on page 18.

Property Name	Property Description
Manager (required)	The case-sensitive name of the queue manager.
Request Queue (required)	The case-sensitive name of the queue where request documents are received. (Queues are named message repositories upon which messages accumulate until they are retrieved by programs that service those queues. Queues are managed by a queue manager.)

Property Name	Property Description
Default Reply Queue	The default queue where responses are directed unless specified otherwise (for example, in the message header or in the request document or listener setting). This property can be ignored if an outlet is already configured for the channel that is being used.
Pending Queue	The name of the queue where Service Manager keeps requests that failed (due to the back-end data server not being available).
Duration	The maximum time that a document can remain in the retry pending queue.
Retry	The interval between retrying pending requests contained in a message.
Filter By Msg Id	Service Manager accepts messages with this message header value only.
Filter By Correlation Id	If set to a value, then incoming messages on the queue are filtered to reject messages without this correlation ID (the set value).
Filter By Group Id	Service Manager accepts messages with this message header value only.
Message Priority	JMS defines ten levels of message priority with values 0 through 9, where 0 is the lowest and 9 is the highest. Zero through four are considered normal settings and five through nine, expedited. The Message Priority field is the default priority value set in the JMS header.
Port (MQ Client only)	The number to connect to an MQ Server Queue Manager. The default port number is 1414.
Host (MQ Client only)	The host on which the MQ Server is located.
Channel (MQ Client only)	Case-sensitive name of the *SRV_CONN* channel that connects with the remote MQ Server queue manager. The default value is *SYSTEM.DEF.SVRCONN*.

Property Name	Property Description
SSL CipherSpec	Defines which Secure Sockets Layer (SSL) specification should be used by the listener. SSL can only be used in client mode.
CCSID	Specifies the coded character set number to use, overriding the machine configured CCSID. If omitted, the configured CCSID is used.
Ignore MQ Conversions	If set to *false* (default), the normal MQ conversions will be performed. If set to *true*, the MQ conversions will be bypassed.
COD	Service Manager accepts Confirmation of Delivery (COD) messages. A global document type definition, which enables processing of these messages when received from MQSeries, is also created. By default, this property is set to *false*.
COA	Service Manager accepts Confirmation of Arrival (COA) messages. A global document type definition, which enables processing of these messages when received from MQSeries, is also created. By default, this property is set to *false*.
Request COD	If set to *true*, iSM sets the report property of an outgoing MQSeries message to request that a Confirmation of Delivery (COD) is sent to the report queue. By default, this property is set to *false*.
Request COA	If set to *true*, iSM sets the report property of an outgoing MQSeries message to request that a Confirmation of Arrival (COA) is sent to the report queue. By default, this property is set to *false*.
Received Report Handling	Determines how to handle received reports on the input queue. Selecting *tree* shows a tree of the data with elements describing the report. Selecting data shows the data as the message contents. **Note:** If the channel is XML, then the incoming report data must be XML.
Report Queue	The destination to which MQSeries must send requested CODs and COAs.
User Headers	If set to *true* (default), user header values are included in the RFH2 header.

Property Name	Property Description
User Registers	If set to *true*, user-level registers are included in the generated RFH2 header. By default, this property is set to *false*.
Accept Zero Length Messages?	If set to *true*, messages with a zero length payload will be processed, treating the value of the Empty Message Signal as the content of the message.
Empty Message Signal	A string to treat as the content of a zero length message, if such messages are accepted. If not supplied, the default value is "".
Whitespace Normalization	Specifies how the parser treats whitespace in element objects. Select *preserve* (default) to turn off all normalization as prescribed by the XML Specification. Select *condense* to remove extra whitespaces in pretty printed documents and for compatibility with earlier versions.
Accepts non-XML (flat) only	Select *true* if non-XML input is expected. If enabled, XML input still can be passed to the listener. Preparsers do not run. By default, this property is set to *false*.
Optimize Favoring	Use this option to customize listener performance. For smaller transactions, select *performance* (default). For large payloads that could monopolize the amount of memory used by iWay Service Manager, select *memory*.
Multithreading	Indicates the number of worker threads. (Equivalent to the number of requests that Service Manager can handle in parallel.) Setting this to a value of greater than 1 enables the listener to handle a second request while an earlier request is being processed. The total throughput of a system is affected by the number of threads operating. The default value is 1. The maximum value is 99.
Execution Time Limit	The maximum time a request can take to complete. A request that takes longer to complete terminates. Prevents runaway requests.
Polling Interval	Indicates the frequency (in seconds) that the listener returns control to Service Manager to determine if a stop listener was requested. The listener is constantly connected to the queue to retrieve incoming messages. The default value is 2.0.

Property Name	Property Description
Default Java File Encoding	The default encoding if incoming message is not self-declaring (that is, XML). The default value is Cp1252.
Agent Precedence	Sets the order by which Service Manager selects agents. Service Manager selects the agent or agents to process the document by searching through the configuration dictionary. Usually, it looks for a document entry in the configuration and when a match is found, the agent specified in that document entry is selected. If a matching document entry is not found or no agent is specified, the engine looks in the input protocol configuration (listener). For the processing agent to be taken directly from the listener (thus ignoring the document entry), use <listener> overrides <document>. Possible values are <*document*> *overrides* <*listener*> (default) and <*listener*> *overrides* <*document*>.
Always Reply to Listener Default	If set to *true*, the default reply definition is used in addition to defined replies. By default, this property is set to *false*.
Error Documents Treated Normally	If set to *true*, error documents will get processed by any configured preemitters. By default, this property is set to *false*.
Listener is Transaction Manager	If set to *true*, agents run in a local transaction. Agents can roll back uncompleted transactions. By default, this property is set to *false*.
Record in Activity Log(s)	If set to *true* (default), activity on this channel will be recorded in the activity logs. If set to *false*, the activity will not be recorded.

Note: The WebSphere MQ listener supports streaming. Streaming is used for large documents or documents for which the application needs to split the input into sections under the same transaction. For more information on streaming and configuring streaming preparsers, see the *iWay Service Manager Component and Functional Language Reference Guide*.

Reference: WebSphere MQ Listener Special Registers

The following table lists and describes the special registers (SREGs) available on the WebSphere MQ listener.

Name	Level	Type	Description
backoutcount	Document	String	The current backout count of the message
correlid	Document	String	The correlation ID, which can be base64(id)
destination	Document	String	The default destination for reply (from message)
expiry	Document	String	The expiry field of the MQMD.
format	Document	String	The format field of the MQMD.
iwayconfig	System	String	The current active configuration name.
iwayhome	System	String	The base at which the server is loaded.
iwayworkdir	System	String	The path to base of the current configuration.
mqcharacterset	Document	String	The character set field of the MQMD.
msgid	Document	String	The message ID, which can be base64(id).
msgsize	Document	Integer	The physical length of the message payload.
name	System	String	The assigned name of the master (listener).
outmsgid	Document	String	The message ID to apply when emitting, which can be base64(id).
persistence	Document	String	The persistence field of the MQMD.
priority	Document	String	The priority of this message.
protocol	System	String	The protocol on which message was received.
putapplicationname	Document	String	The Put application name field of the MQMD.

Name	Level	Type	Description
queuedepth	Document	String	The current depth of the queue at time of lookup to this register. If it cannot be read, -1 is returned.
replytoq	Document	String	The replyto queue field of the MQMD.
replytoqmgr	Document	String	The replyto queue manager field of the MQMD.
report	Document	String	The report field of the MQMD.
reporttype	Document	String	The report type. This can be coa, cod or other.
type	Document	String	The type of the message. It can be either report, reply, request, or datagram.
userid	Document	String	The user ID field of the MQMD.
tid	Document	String	Unique transaction ID.

Websphere MQ Listener Configuration Example

You are receiving documents on a WebSphere MQ queue, named default running under a Queue Manager named QM_iwxfoc using a channel named SYSTEM.DEF.SVRCONN. Specify the listener values as follows:

- **Manager:** QM_iwxfoc
- **Request Queue:** default
- **Default Reply Queue:** out
- **Port:** 1414

- **Channel:** SYSTEM.DEF.SVRCONN

Configuration parameters for new listener of type MQ	
Manager *	Name of the local MQ Series queue manager to be used QM_iwxfoc
Request Queue *	Queue on which request documents are received default
Default Reply Queue	If document and message does not contain a reply queue, use this queue out
Pending Queue	Queue on which requests to be retried are stored
Duration	Maximum time that a document can remain in the retry pending queue 86400
Retry	Interval between retrying pending requests 600
Filter By Msg Id	If set to a value, incoming messages on the queue are filtered to reject those without this message id
Filter By Correlation Id	If set to a value, incoming messages on the queue are filtered to reject those without this correlation id
Filter By Group Id	If set to a value, incoming messages on the queue are filtered to reject those without this group id
Message Priority	Outgoing message priority (if omitted uses incoming priority)
Port	**For MQ Client only.** Port number to connect to an MQ Server 1414
Host	**For MQ Client only.** Host on which MQ Server is located
Channel	**For MQ Client only.** Case sensitive name of the *SRV_CONN* channel on the remote queue mgr. SYSTEM.DEF.SVRCONN

Configuring a WebSphere MQ Emitter

Messages are sent to the destinations at the completion of a workflow. The state of the document determines which destination is used. The order in which the destinations are used cannot be predicted. Destinations cannot accept responses. These configurations also are used to send a message over a different protocol.

Note: Configuring a WebSphere MQ emitter is not required if the emitter protocol used in the outlet of the channel is the same as the listener protocol used in the inlet of the channel. For more information on inlets and outlets, see *Defining an Inlet* on page 15 and *Defining an Outlet* on page 25.

To route an output document to a protocol other than that of the listener, configure an emitter. For example, an application may send input over a TCP listener but want to route the output to a WebSphere MQ queue.

Reference: WebSphere MQ Emitter Properties

The following table lists and describes the WebSphere MQ emitter properties. For instructions on creating an emitter, see *How to Create an Emitter* on page 29.

Property Name	Property Description
Destination (required)	The location to where your reply is delivered. The format of the destination is Queue@QueueManager.
Manager (required)	The name of the queue manager to which the server must connect.
Set Reply Correlation Id	The correlation ID on which to be filtered. The correlation ID is set in the MQSeries header. Unless a base64() function is used in the correlation ID, the search correlation ID is assumed to be a Unicode string correlation ID. This follows the current rules for correlation ID in the rest of the system.
Message Priority	The message priority. The value must be greater than or equal to zero; zero is the lowest priority.
Request COD	The Request Confirmation of Delivery report message. Requires a Report Queue.
Request COA	The Request Confirmation of Arrival report message. Requires a Report Queue.
Report Queue	The reply information for the outgoing document. Intended as a report destination.
Host (MQ Client only)	The host on which the MQ Server is located.

Property Name	Property Description
Port (MQ Client only)	The number to connect to an MQ Server queue manager.
Channel (MQ Client only)	Case-sensitive name of the channel that connects with the remote MQ Server queue manager. SYSTEM.DEF.SVRCONN is the default channel name for MQSeries.
SSL CipherSpec	Defines that Secure Sockets Layer (SSL) is to be used. This is configured for the channel by the IBM Websphere MQ Administrator. SSL can be used only in client mode.
CCSID	Specifies the coded character set number to use, overriding the machine configured CCSID. If omitted, the configured CCSID is used.
Message Format	The format name of the message data. This is the name that the sender of the message can use to indicate the nature of the data in the message to the receiver. Any characters that are in the queue manager character set can be specified for the name, but iWay Software recommends that the name be restricted to uppercase A through Z or numeric digits 0 through 9. String is the default value.
Message Type	Type of MQ message to be written. Select one of the following values from the drop-down list: ❏ datagram (default) ❏ request ❏ response
Persistence	The values allowed are queue, persistent, and non-persistent. Queue is the default value, which means that it is persistent as the queue is defined.
Expiry	The time that the message remains available in the system awaiting pick up. It is an integer measured in 1/10 seconds. No expiration is the default value.

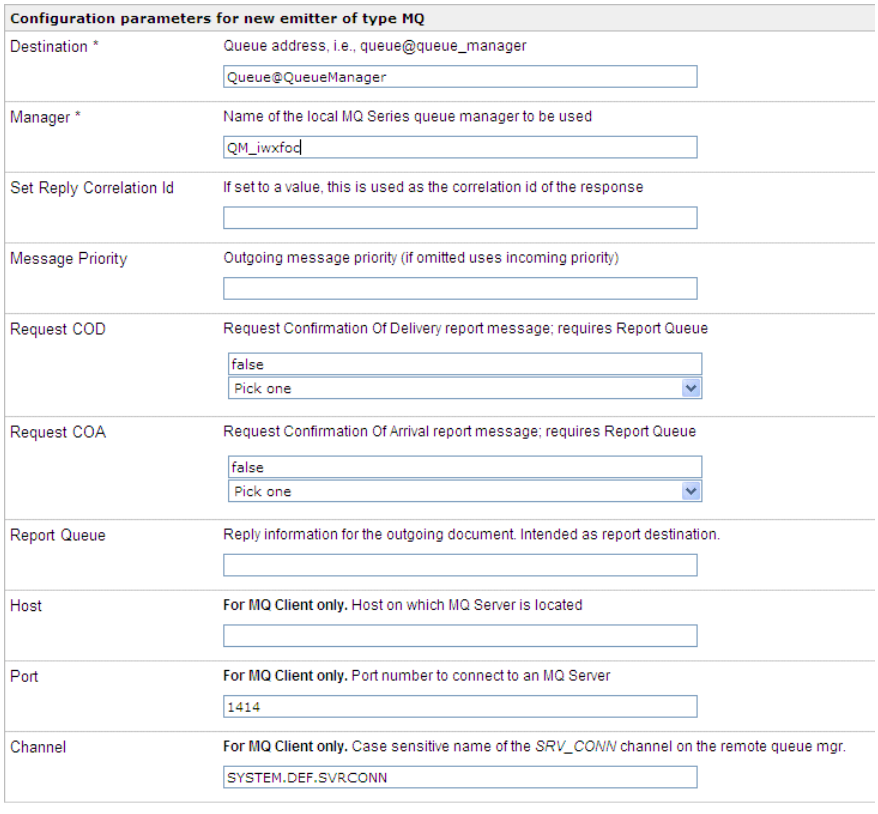
Property Name	Property Description
User Headers	If set to *true*, user header values are included in the RFH2 header. If set to *false* user header values are not included in the RFH2 header (this is the default setting).
User registers	If set to *true*, user-level registers are included in the generated RFH2 header. By default, this property is set to *false*.

WebSphere MQ Emitter Configuration Example

You want to emit documents to a WebSphere MQ queue, named default, running under a Queue Manager named QM_iwxfoc using a channel named SYSTEM.DEF.SVRCONN. Specify the emitter values as follows:

- **Destination:**
- **Manager:** QM_iwxfoc
- **Request Queue:** default –
- **Default Reply Queue:** out –
- **Port:** 1414

- **Channel:** SYSTEM.DEF.SVRCONN



Configuring WebSphere MQ and iWay Service Manager to Communicate With Transport Layer Security / Secure Sockets Layer

This section describes how to configure WebSphere MQ and iWay Service Manager (iSM) to communicate with the Transport Layer Security (TLS) / Secure Sockets Layer (SSL) cryptographic protocols.

CipherSpec Mappings

In MQ, a particular choice of key exchange algorithm, block cipher and MAC algorithm for TLS/SSL communication is called a CipherSpec. In Java, the same concept is called a CipherSuite. In general, the name of a CipherSpec is different than its equivalent CipherSuite.

In a Java program, the CipherSpec must be mapped to a CipherSuite to instruct the JSSE provider which combination of algorithms is desired. By default, the IBM MQ libraries for Java assume the IBMJSSE2 provider will be used. This works well on IBM JVMs where IBMJSSE2 is available. Unfortunately, the default mapping produces CipherSuite names that do not exist in an Oracle JVM.

When running in an Oracle JVM, MQ needs a different mapping of CipherSpec to CipherSuite to be compatible with the SunJSSE provider. This mapping is enabled by defining the system property com.ibm.mq.cfg.useIBMCipherMappings to false.

Default Mapping (when com.ibm.mq.cfg.useIBMCipherMappings is absent or true).

CipherSpec	CipherSuite	Protocol	FIPS?	Note
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	SSL_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	TLSv1.2	true	deprecated
ECDHE_ECDSA_AES_128_CBC_SHA256	SSL_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLSv1.2	true	
ECDHE_ECDSA_AES_128_GCM_SHA256	SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLSv1.2	true	
ECDHE_ECDSA_AES_256_CBC_SHA384	SSL_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLSv1.2	true	
ECDHE_ECDSA_AES_256_GCM_SHA384	SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLSv1.2	true	
ECDHE_ECDSA_NULL_SHA256	SSL_ECDHE_ECDSA_WITH_NULL_SHA	TLSv1.2	false	deprecated

CipherSpec	CipherSuite	Protocol	FIPS?	Note
ECDHE_ECDSA_RC4_128_SHA256	SSL_ECDHE_ECDSA_WITH_RC4_128_SHA	TLSv1.2	false	deprecated
ECDHE_RSA_3DES_EDE_CBC_SHA256	SSL_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1.2	true	deprecated
ECDHE_RSA_AES_128_CBC_SHA256	SSL_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLSv1.2	true	
ECDHE_RSA_AES_128_GCM_SHA256	SSL_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLSv1.2	true	
ECDHE_RSA_AES_256_CBC_SHA384	SSL_ECDHE_RSA_WITH_AES_256_CBC_SHA384	TLSv1.2	true	
ECDHE_RSA_AES_256_GCM_SHA384	SSL_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLSv1.2	true	
ECDHE_RSA_NULL_SHA256	SSL_ECDHE_RSA_WITH_NULL_SHA	TLSv1.2	false	deprecated
ECDHE_RSA_RC4_128_SHA256	SSL_ECDHE_RSA_WITH_RC4_128_SHA	TLSv1.2	false	deprecated

CipherSpec	CipherSuite	Protocol	FIPS?	Note
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA	TLSv1	true	
TLS_RSA_WITH_AES_128_CBC_SHA256	SSL_RSA_WITH_AES_128_CBC_SHA256	TLSv1.2	true	
TLS_RSA_WITH_AES_128_GCM_SHA256	SSL_RSA_WITH_AES_128_GCM_SHA256	TLSv1.2	true	
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA	TLSv1	true	
TLS_RSA_WITH_AES_256_CBC_SHA256	SSL_RSA_WITH_AES_256_CBC_SHA256	TLSv1.2	true	
TLS_RSA_WITH_AES_256_GCM_SHA384	SSL_RSA_WITH_AES_256_GCM_SHA384	TLSv1.2	true	
TLS_RSA_WITH_NULL_SHA256	SSL_RSA_WITH_NULL_SHA256	TLSv1.2	false	deprecated
TLS_RSA_WITH_RC4_128_SHA256	SSL_RSA_WITH_RC4_128_SHA	TLSv1.2	false	deprecated
TLS_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA	TLSv1	false	deprecated
TLS_RSA_WITH_3DES_EDE_CBC_SHA	SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1	true	deprecated

CipherSpec	CipherSuite	Protocol	FIPS?	Note
DES_SHA_EXPORT	SSL_RSA_WITH_DES_CBC_SHA	SSLv3	false	deprecated
DES_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA	SSLv3	false	deprecated
FIPS_WITH_DES_CBC_SHA	SSL_RSA_FIPS_WITH_DES_CBC_SHA	SSLv3	false	deprecated
FIPS_WITH_3DES_EDE_CBC_SHA	SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA	SSLv3	false	deprecated
NULL_MD5	SSL_RSA_WITH_NULL_MD5	SSLv3	false	deprecated
NULL_SHA	SSL_RSA_WITH_NULL_SHA	SSLv3	false	deprecated
RC2_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	SSLv3	false	deprecated
RC4_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC4_40_MD5	SSLv3	false	deprecated
RC4_MD5_US	SSL_RSA_WITH_RC4_128_MD5	SSLv3	false	deprecated
RC4_SHA_US	SSL_RSA_WITH_RC4_128_SHA	SSLv3	false	deprecated

CipherSpec	CipherSuite	Protocol	FIPS?	Note
RC4_56_SHA_EXPORT1024	SSL_RSA_EXPORT1024_WITH_RC4_56_SHA	SSLv3	false	deprecated
TRIPLE_DES_SHA_US	SSL_RSA_WITH_3DES_EDE_CBC_SHA	SSLv3	false	deprecated

Oracle Mapping (when com.ibm.mq.cfg.useIBMCipherMappings is false).

CipherSpec	CipherSuite	Protocol	FIPS?	Note
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	TLSv1.2	false	deprecated
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLSv1.2	false	
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLSv1.2	false	
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLSv1.2	false	
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLSv1.2	false	

CipherSpec	CipherSuite	Protocol	FIPS?	Note
ECDHE_ECDSA_NULL_SHA256	TLS_ECDHE_ECDSA_WITH_NULL_SHA	TLSv1.2	false	deprecated
ECDHE_ECDSA_RC4_128_SHA256	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	TLSv1.2	false	deprecated
ECDHE_RSA_3DES_EDE_CBC_SHA256	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1.2	false	deprecated
ECDHE_RSA_AES_128_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLSv1.2	false	
ECDHE_RSA_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLSv1.2	false	
ECDHE_RSA_AES_256_CBC_SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	TLSv1.2	false	
ECDHE_RSA_AES_256_GCM_SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLSv1.2	false	
ECDHE_RSA_NULL_SHA256	TLS_ECDHE_RSA_WITH_NULL_SHA	TLSv1.2	false	deprecated
ECDHE_RSA_RC4_128_SHA256	TLS_ECDHE_RSA_WITH_RC4_128_SHA	TLSv1.2	false	deprecated

CipherSpec	CipherSuite	Protocol	FIPS?	Note
TLS_RSA_WITH_3DES_EDE_CBC_SHA	SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1	false	deprecated
TLS_RSA_WITH_AES_128_CBC_SHA	TLS_RSA_WITH_AES_128_CBC_SHA	TLSv1	false	
TLS_RSA_WITH_AES_128_CBC_SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256	TLSv1.2	false	
TLS_RSA_WITH_AES_128_GCM_SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256	TLSv1.2	false	
TLS_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_AES_256_CBC_SHA	TLSv1	false	
TLS_RSA_WITH_AES_256_CBC_SHA256	TLS_RSA_WITH_AES_256_CBC_SHA256	TLSv1.2	false	
TLS_RSA_WITH_AES_256_GCM_SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384	TLSv1.2	false	
TLS_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA	TLSv1	false	deprecated
TLS_RSA_WITH_NULL_SHA256	TLS_RSA_WITH_NULL_SHA256	TLSv1.2	false	deprecated
TLS_RSA_WITH_RC4_128_SHA256	SSL_RSA_WITH_RC4_128_SHA	TLSv1.2	false	deprecated

CipherSpec	CipherSuite	Protocol	FIPS?	Note
NULL_MD5	SSL_RSA_WITH_NULL_MD5	SSLv3	false	deprecated
NULL_SHA	SSL_RSA_WITH_NULL_SHA	SSLv3	false	deprecated
RC4_MD5_EXPORT	SSL_RSA_EXPORT_WITH_RC4_40_MD5	SSLv3	false	deprecated
RC4_MD5_US	SSL_RSA_WITH_RC4_128_MD5	SSLv3	false	deprecated

Deprecated CipherSpecs

The tables above show which CipherSpecs are deprecated as of the latest MQ version. Deprecated CipherSpecs can still be used, but they must be enabled by configuring the MQ Server.

Set the environment variable AMQ_SSLWEAK_CIPHER_ENABLE to a single CipherSpec name, a comma-separated list of CipherSpec names, or the value *ALL*. For example:

```
AMQ_SSL_WEAK_CIPHER_ENABLE=ECDHE_RA_RC4_128_SHA256
```

To enable an SSLv3 CipherSpec, set the environment variable AMQ_SSL_V3_ENABLE to *1*. For example:

```
AMQ_SSL_V3_ENABLE=1
AMQ_SSL_WEAK_CIPHER_ENABLE=RC4_MD5_US
```

Alternatively, you can add an entry in the SSL stanza in the .ini file of the queue manager in the following location:

```
C:\ProgramData\IBM\MQ\qmgrs\<qmgrname>\qm.ini
```

where:

`<qmgrname>`
: Is the queue manager name.

For example:

```
SSL:
      AllowWeakCipherSpec=ECDHE_RA_RC4_128_SHA256
```

To enable an SSLv3 CipherSpec, set AllowSSLV3 to *Y*. For example:

```
  SSL:
      AllowSSLV3=Y
      AllowWeakCipherSpec=RC4_MD5_US
```

The CipherSpec(s) listed must be deprecated for your MQ version, otherwise the value is rejected and the channel will not start.

The Queue Manager must be restarted for the changes to take effect.

Disabled CipherSuite

In Java, the strong CipherSuites are enabled by default and the weak CipherSuites are disabled. However, before you can use a disabled CipherSuite, it must be re-enabled.

Procedure: How to Re-enable CipherSuite

To re-enable CipherSuite:

1. Locate the java.security file within the JRE used to run iWay Service Manager. For example:

   - If the JRE is standalone, the location could be:

     ```
     JAVA_HOME\lib\security\java.security
     ```

   - If the JRE is part of a JDK, the location could be:

     ```
     JAVA_HOME\jre\lib\security\java.security
     ```

2. Edit the java.security file and modify the following property if necessary:

   ```
   jdk.tls.disabledAlgorithms
   ```

   The instructions are in the file comments.

   You can also edit the following property:

   ```
   jdk.certpath.disabledAlgorithms
   ```

3. Restart iWay Service Manager for the changes to take effect.

TLS/SSL Protocol Version

The CipherSpec also determines the TLS/SSL Protocol version. You must select the same TLS/SSL version in the SSL Context Provider in iWay Service Manager. Selecting an earlier version will not work because it represents the maximum version supported by the client. Selecting a current protocol version usually works because it can be downgraded.

FIPS

iWay Service Manager will automatically require FIPS in the connection parameters if the CIpherSpec mapping supports it.

CipherSuite Conflicts

A complication occurs when multiple CipherSpecs map to the same CipherSuite but with different SSL/TLS versions. By default, the SSLv3 mapping is assumed and the TLSv1 mappings will not work. To enable the TLSv1 mappings, the following system property must be set to *true*:

```
com.ibm.mq.cfg.preferTLS
```

This property affects only the CipherSpecs listed in the following table. When absent, the default value is false for the above property.

Prefer TLS	CipherSpec	CipherSuite	Protocol
false	DES_SHA_EXPORT	SSL_RSA_WITH_DES_CBC_SHA	SSLv3
false	TRIPLE_DES_SHA_US	SSL_RSA_WITH_3DES_EDE_CBC_SHA	SSLv3
false	TLS_RSA_WITH_DES_CBC_SHA	unavailable	
false	TLS_RSA_WITH_3DES_EDE_CBC_SHA	unavailable	
true	DES_SHA_EXPORT	unavailable	
true	TRIPLE_DES_SHA_US	unavailable	

Prefer TLS	CipherSpec	CipherSuite	Protocol
true	TLS_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA	TLSv1
true	TLS_RSA_WITH_3DES_EDE_CBC_SHA	SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1

Private Key

When creating the private key for the queue manager, you must select a key algorithm compatible with the CipherSpec.

- If your CipherSpec starts with TLS_RSA_ or ECDHE_RSA_, then you must select a signature algorithm ending with RSA , for example, SHA1WithRSA.
- If the CipherSpec starts with ECDHE_ECDSA, then you must select a signature algorithm ending with ECDSA, for example SHA2WithECDSA.

The security strength of the key size must be strong enough.

- For RSA, the key size must be 1024 or higher.
- For ECDSA, the key size must be 256 or higher.

Smaller key sizes will not work causing errors to appear at runtime.

Setting System Properties

System properties for iWay Service Manager can be set directly on the command line of the Java field, or in the Java Settings of the iWay Service Manager console.

For example:

```
java -Dcom.ibm.mq.cfg.useIBMCipherMappings=false -
Dcom.ibm.mq.cfg.preferTLS=true ...
```

Configuring MQ for TLS/SSL

This section is a tutorial that explains how to configure a Queue Manager for TLS/SSL communication. The instructions apply to environments where MQ Explorer is available (Windows and Linux x86-64).

First, login to the computer where MQ is installed.

Installing the Unrestricted Policy Files

This section describes how to install the unrestricted policy files.

Determine the Version of the IBM JRE

Change the directory to:

```
<WebSphere MQ install dir>\java\jre\bin
```

Then, run the following file:

```
java -version
```

For example, MQ 7.5.0.2 runs JRE 1.6 SR13, and MQ 8.0.0.2 runs JRE 1.7.

Download the Unrestricted Policy Files

The policy files from Oracle do not work with an IBM JVM.

You can download the unrestricted policy files directly from IBM. For more information, see *https://www-01.ibm.com/marketing/iwm/iwm/web/preLogin.do?source=jcesdk*.

If that does not work, you can try searching for *Unrestricted SDK JCE policy files* within that site, and then download the policy files appropriate for the version of the IBM JRE.

Copy the Policy Files

Unzip the archive and copy local_policy.jar and US_export_policy.jar to:

```
<WebSphere MQ install dir>\java\jre\lib\security
```

Restart MQ

Stop all of IBM WebSphere MQ and restart it.

On Windows, you can achieve this by restarting the WebSphere MQ service. You can manage the services on that platform by running services.msc.

Starting MQ Explorer

From the start menu, expand *IBM Websphere MQ* and select *Websphere MQ Explorer*. If you have it installed, double-click strmqcfg.exe from your installation directory.

You can also run strmqcfg.cmd from the command prompt. For example:

```
C:\Program Files\IBM\WebSphere MQ\bin\strmqcfg.cmd
```

Create a New Queue Manager

To create a new queue manager:

1. Right-click *Queue Managers*, select *New*, and then click *Queue Manager*.
2. Enter the queue manager name, QM_SSL and then click *Next*.
3. Keep clicking *Next* to accept the default settings until you reach the dialog box for selecting ports.
4. Select a free port, for example, *1413*.
5. Click *Finish*.

Repair the Port

Repairing the port is not always taken into consideration.

1. Right-click *Queue Managers*, click *QM_SSL*, and then select *Properties*.
2. In the left pane, click *TCP*.
3. Update the port (if needed) to the previously selected value, for example, *1413*.
4. Right-click *Queue Managers*, select *QM_SSL*, *Stop*, and then click *Immediate*.
5. Right-click *Queue Managers*, select *QM_SSL*, *Start*, and then select *AS Create*, and click *OK*.

Create a Queue

To create a queue:

1. Right-click *Queue Managers*, select *QM_SSL*, *Queues*, *New*, and then click *Local Queue*.
2. In the Name field, enter *q1*.
3. Click *Next*.
4. Click *Finish*.

Create a Channel

To create a channel, ensure that QM_SSL is running. If it is not, start it.

1. Expand *Queue Managers* and click *QM_SSL*.
2. Right-click *Queue Managers*, select *QM_SSL*, click *Channels*, select *New*, and then click *Server Connection Channel*.
3. Enter the following name, being careful not to mistake SVR with SRV:

   ```
   SSL.SVRCONN
   ```

4. Create it with attributes. For example:

   ```
   SYSTEM.DEF.SVRCONN
   ```

5. Click *Next*
6. Click *SSL* in the left pane.
7. Select a CipherSpec. For example:

   ```
   TLS_RSA_WITH_AES_128_CBC_SHA
   ```

8. Change the SSL Authentication to *Optional* (this makes the SSL Client Authentication optional).
9. Click *Finish*.

Create an Auth Record to Allow Access

By default, MQ 7.1 and higher are configured to block all MQ administrators from connecting as a client to the queue manager.

1. Click *Queue Managers*, select *QM_SSL*, click *Channels*, and then select *Channel Authentication Records*.

   Note: If you are also an MQ administrator, delete the default authentication record for channels.

2. Right-click *Channel Authentication Records*, select *New*, and then click *Channel Authentication Record*.
3. Select *Allow Access* and then click *Next*.
4. Select *Client Application User ID*, then click *Next*.
5. Type the following syntax in the Channel Profile field:

   ```
   SSL.SVRCONN
   ```

6. Click *Next*.
7. Type the user name (for example, mquser) in the Remote Client User ID field and then click *Next*.
8. Select the User ID of the channel and then click *Next*.
9. Select the Queue Manager and then click *Next*.
10. Enter a description (optional), click *Next*, then click *Finish*.

You can now access this channel with the new user name and its password. Ensure that the username exists in the operating system.

Update SSL Properties

To update SSL Properties:

1. Right-click *Queue Managers*, select *QM_SSL*, and then click *Properties*.
2. Select *SSL* in the left pane.
   - ❑ For MQ 8, the SSL key repository already ends in:

     ```
     ...\QM_SSL\ssl\key
     ```

   - ❑ For MQ7, append the \key to obtain:

     ```
     ...\QM_SSL\ssl\key
     ```

     This value is a file path without the implied .kdb extension. Note that the directory (without the \key) for the next step in the example is:

     ```
     C:\ProgramData\IBM\MQ\qmgrs\QM_SSL\ssl
     ```

     The Certificate label for the following step in the example, is:

     ```
     ibmwebspheremqqm_ssl
     ```

3. Leave the SSL FIPS Required parameter to *No*, unless you need FIPS and your CipherSpec supports FIPS.

Starting the IBM Key Management Utility

From the Start menu, expand *IBM Websphere MQ* and click *IBM Key Management*, or double-click ikeyman.exe from where you have it installed on your computer. For example:

```
C:\Program Files\IBM\WebSphere MQ\java\jre\bin\ikeyman.exe
```

Create a New Key Database

To create a new Key Database:

1. Select *Key Database File*, and click *New*.
2. Enter the Key Database Type. For example, *CMS*.
3. Enter the file name. For example, *key.kdb*.
4. In the location field, enter the key repository directory.
   - ❑ For MQ 8, the SSL key repository already ends in:

     ```
     ...\QM_SSL\ssl\key
     ```

- For MQ7, append the \key to obtain:

  ```
  ...\QM_SSL\ssl\key
  ```

  This value is a file path without the implied .kdb extension. Note that the directory (without the \key) for the next step in the example is:

  ```
  C:\ProgramData\IBM\MQ\qmgrs\QM_SSL\ssl
  ```

5. Enter and confirm the following password:

   ```
   secret
   ```

6. Select *Stash password to a file*.

   Note: You must select this option even though it is not available with all key database types. The default type is set to CMS, but you cannot have MQ use your JKS keystore.

Create a New Personal Certificate (That Creates a New Private Key)

To create a new personal certificate:

1. In the section above the contents pane, select *Personal Certificates*.
2. On the right pane, click *New Self-Signed*.
3. In the Key Label field, enter the Certificate label.

   For example:

   ```
   ibmwebspheremqqm_ssl
   ```

4. Select a signature algorithm compatible with your CipherSpec.

   - If your CipherSpec starts with ECDHE_ECDSA_..., then select a signature algorithm ending with ECDSA, for example:

     ```
     SHA2WithECDSA
     ```

   - If your CipherSpec starts with TLS_RSA_... or ECDHE_RSA_..., then select a signature algorithm ending with RSA, for example:

     ```
     SHA1WithRSA
     ```

5. Select your Key size.

   - For RSA, the key size must be 1024 or higher.
   - For ECDSA, the key size must be 256 or higher.

   Note: Shorter key sizes will not work and can return errors at runtime.

6. Change the organizational unit to the queue manager name, *QM_SSL*.
7. Select the validity period. The default setting is 365 days.

   Test environments can use longer validity periods. If necessary, the MQ will clip the value to its maximum validity period.
8. Click *OK*.

Export the Certificate to Build the Trust Store for iWay Service Manager

To export the certificate to build the trust store for iWay Service Manager:

1. Select *ibmwebspheremqqm_ssl* in the list of personal certificates.
2. In the right pane, click *Extract Certificate*.
3. Select the *Binary DER* data type.
4. Change the certificate file name to *qm_ssl.der*.
5. Change the location to:

   ```
   c:\temp
   ```
6. Copy the following file to the host where iWay Service Manager is running.

   ```
   qm_ssl.der
   ```

Enable the Chosen CipherSpec

The CipherSpec file, TLS_RSA_WITH_AES_128_CBC_SHA, is considered strong and it is enabled by default.

If the chosen CipherSpec is deprecated, it needs to be enabled explicitly. For more information, see *Deprecated CipherSpecs* on page 349.

Restart the Queue Manager in MQ Explorer

To restart the queue manager in MQ explorer:

1. Right-click *Queue Managers*, click *QM_SSL*, *Stop*, *Immediate*, and then click *OK*.
2. Right-click *Queue Managers*, click *QM_SSL*, select *Start*, and then click *OK*.

If you are only changing the SSL configuration, you can refresh the queue manager by right-clicking *Queue Managers*, selecting *QM_SSL*, clicking *Security*, and then selecting *Refresh SSL*. Note that changing the qm.ini file requires a complete stop and restart.

Configuring iWay Service Manager

This section describes the steps involved to configure iWay Service Manager.

Creating the Trust Store

To create the trust store:

1. Create a Keystore by importing the self-signed certificate qm_ssl.jks. For example:

   ```
   keytool -import -keystore qm_ssl.jks -file qm_ssl.der -alias
   ibmwebspheremqqm_ssl -storepass secret
   ```

2. When you are asked whether to Trust this certificate, enter *Yes*.

Creating a Keystore Provider Pointing to the qm_ssl.jks File

To create a Keystore provider pointing to the qm_ssl.jks file:

1. From the iWay Service Manager console, click *Server* from the toolbar.
2. On the left pane, click *Security Provider*, and then under Keystore Provider, click *New*.
3. In the Provider Name field, enter *qm_ssl*.
4. In the location field, enter the keystore location.
5. In the Keystore password field, enter the following password:

   ```
   secret
   ```

6. In the Keystore Type section, select *JKS* and then click *Add*.

Creating an SSL Context Provider

To create an SSL context provider:

1. On the left pane, click *Security Provider*.
2. Under SSL Context Provider, click *New*.
3. In the provider name field, enter the following name:

   ```
   qm_ssl_ctx
   ```

4. Set both the trust store and keystore to *qm_ssl*, even though the keystore is not used.
5. Select the same version of the Security Protocol as your CipherSpec. This information is given in the table of CipherSpecs. Selecting an earlier version will not work, so you must select the latest version.
6. Click *Add*.

Configure an MQ Emit Agent or MQ Listener

The following table shows the configuration of an MQ component.

Parameter	Value
Queue Name	q1
Queue Manager	SM_SSL
Host	<hostname>
Port	<port>
Channel	SSL.SVRCONN
Authentication	true
User ID	mquser
User Password	<password>
SSL CipherSpec	TLS_RSA_WITH_AES_128_CBC_SHA
SSL Context Provider	qm_ssl_ctx

Configuring System Properties

If you are running an Oracle JVM, the CipherSpec to CipherSuite mapping for Oracle JVMs by setting the system property to false.

The system property can be set directly on the Java command line or in the Java settings of the iWay Service Manager console. For example:

```
java -Dcom.ibm.mq.cfg.useIBMCipherMappings=false -
Dcom.ibm.mq.cfg.preferTLS=true ...
```

Copying the MQ Client Jar Files

This step is not specific to SSL, but it is necessary to complete the installation.

1. Find the JRE within the MQ installation. For example:

   ```
   c:\Program Files\IBM\WebSphere MQ\java\jre
   ```

2. Depending on your version of MQ, locate the following jar files under lib or lib\ext:

   For MQ 7.5.0.2:

   - com.ibm.mq.commonservices.jar
   - com.ibm.mq.jar
   - com.ibm.mq.jmqi.jar
   - com.ibm.mq.pcf.jar
   - com.ibm.mq.headers.jar
   - connector.jar

   For MQ 8.0 and higher, you only need com.ibm.mq.allclient.jar.

3. Copy the IBM jars to the lib directory of the iWay Service Manager installation. This only needs to be done once.

Troubleshooting

Many things can go wrong when configuring MQ and iWay Service Manager to communicate with TLS/SSL. This section provides help and support when problems occur.

The following IBM article explains how to troubleshoot Java/JMS SSL Configurations: *http://www-01.ibm.com/support/docview.wss?uid=swg21614686*.

Always inspect the queue manager logs in the following location, and replace QM_SSL with your queue manager name in the path.

```
C:\ProgramData\IBM\MQ\Qmgrs\QM_SSL\errors
```

It is recommended to turn on JSSE debugging in iWay Service Manager. Add the following system properties on the command line of the Java field:

```
ava -Djavax.net.debug=all -Djava.security.debug=certpath ...
```

For example, edit a copy of iway8.cmd or iway8.sh, and then start the modified script in a test window to view the traces.

WebSphere MQSeries Troubleshooting

The following table lists and describes errors that you may encounter when using an MQ listener.

Error	Reason	Solution
`setupQM: error received java.lang.UnsatisfiedLink Error: no mqjbnd02 in java.library.path: [line 9]`	The system path was not updated to include the directory containing mqjdbnd02.dll.	Use the Path Settings pane to register [MQJAVA]\lib in the Path field. The server must be stopped and restarted for this to take effect.
`Failed MQ operation Completion Code 2, Reason 2058:`	Queue manager name not valid or not known.	Verify that the case-sensitive queue manager defined actually exists. Use the MQSeries explorer (or the DISPLAY QL command) to verify that it is running.
`Failed MQ operation Completion Code 2, Reason 2085`	The system was unable to find the queue(s) specified.	Either the queue was not defined or was incorrectly spelled. The MQSeries listener is case-sensitive. Verify that the queue name exists and is entered in the Configure MQ Listener section in the correct case.
`Cannot load MQ Series. Probable classpath error. Java.lang.NoClassdeffound Error: javax/resource/ ResourceException`	The Java path of MQSeries is not in your classpath.	Use the Path Settings pane to add connector.jar to the class path.

Error	Reason	Solution

MQSeries Listener fails to connect Completion Code 2, Reason 2059 on AIX	Queue manager not available.	If you are using MQSeries V5.2, you can add an extra stanza to mqs.ini which affects the part of shared memory MQSeries uses when connecting. The property is IPCCBaseAddress and is set on a per queue manager basis. By default, this property is set to the value 8, but iWay Software recommends setting this to the value 11. The following is an example of an altered QueueManager stanza in an mqs.ini file:
		`QueueManager:`
		`Name=MQJavaTest`
		`Prefix=/var/ mqm`
		`IPCCBaseAddress=12`
		Values permitted for this property are 4, 5, 8, 9, 10, 11, or 12. For more information, see the MQSeries V5.2 documentation. Also, when using MQSeries 5.2 in conjunction with Service Manager on the AIX platform with WebSphere, you must export the LDR_CNTRL variable as follows.
		`export LDR_CNTRL=MAXDATA=0x300 00000`
		Similar issues can be found in IBM Defect 114907.

Registering MQJMS JAR and DLL Files

You must register the following files, which are in the *installation_drive:**install_folder*\ibm \WebSphere MQ\Java\Lib\ directory:

Platform/Notes	JAR Files
JAR (Client-side)	
All	❑ com.ibm.mq.jar ❑ com.ibm.mqjms.jar ❑ fscontext.jar ❑ connector.jar ❑ jms.jar ❑ j2ee.jar
Depending on service pack and platform, also register	com.ibm.mq.iiop.jar
OS/390 and z/OS, also register	jta.jar
Specify the location of the property files directory you receive when you install the MQSeries java classes, such as mqji_en_US.properties.	
DLL (Server-side)	

Register the DLL directory (the directory where the name interface resides), for example,
`[mqseries home]/java/lib`
For some platforms, you must explicitly add MQSERIES HOME/java/lib to the path. For example, for OS/390 you must issue the following command:
`export LIBPATH=/mqm/java/lib:$LIBPATH`	

For instructions on registering JAR files, see *Registering Library Files and Setting JVM Options* on page 40.

MQJMS Listener Properties

The following table lists and describes the MQJMS listener properties. For instructions on creating a listener, see *How to Create a Listener* on page 18.

Property Name	Property Description

Form of Input	`TOPIC`
	Is for a TopicSession in the Publish and Subscribe domain.
	`QUEUE`
	Is for a QueueSession in the PTP domain.

Queue Manager	The WebSphere MQ queue manager name.
Channel	The server connection channel name, only required for MQ client connection.
Host	The host name or IP address, required for MQ client connection only.
Port	The port number, required for MQ client connection only.
Form of Acknowledgment	The way the listener acknowledges received messages: ❑ auto acknowledge ❑ client provides ❑ duplicates permitted ❑ transactional
Receiver Name (required)	The name of the queue on which input documents will be received for processing by Service Manager.
Default Reply	The default queue or topic to which response documents are written.
Pending Queue	The queue to hold documents pending retry.
User	A valid user ID defined to the JMS Server.
Password	A valid password defined to the JMS Server.

Property Name	Property Description
Duration	The maximum time that a document can remain in the retry pending queue.
Retry	The interval between retrying pending requests.
Message Selector	If used, input is filtered by this selector.
Message Priority	Outgoing message priority (if omitted, uses incoming priority).
Output Message Type	Select *Bytes*, *Text*, or *Dynamic*.
Accepts non-XML (flat) only	Select *true* if non-XML input is expected. If enabled, XML input still can be passed to the listener. Preparsers do not run.
Optimize Favoring	Use this option to customize listener performance. For smaller transactions, select *performance*. For large payloads that could monopolize the amount of memory used by Service Manager, select *memory*.
Multithreading	Indicates number of worker threads (documents or requests) that Service Manager can handle in parallel. Setting it greater than 1 enables the listener to handle a second request while an earlier request is being processed. Default: 1 Max value: 99
Execution Time Limit	The maximum time a request can take to complete. A request that takes longer to complete terminates. Prevents runaway requests.
Polling Interval	The interval (in seconds) at which to check for new input or stop listener requests. The default value is 2.0.
Default Java File Encoding	The default encoding if incoming message is not self-declaring (that is, XML).

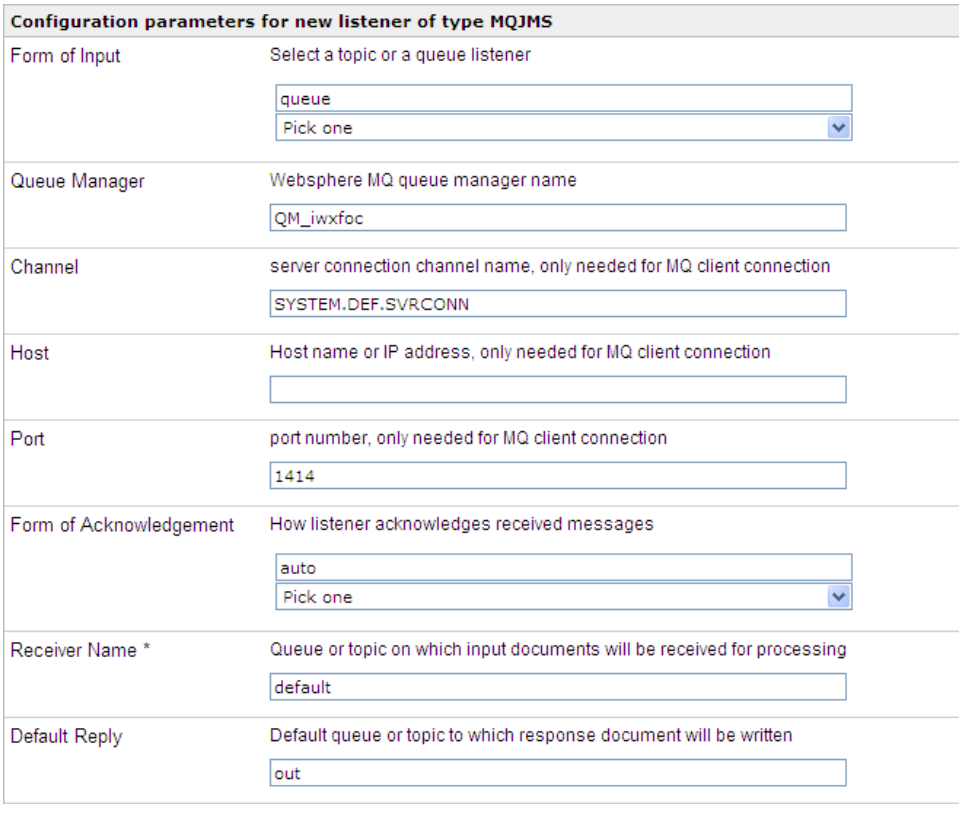
Property Name	Property Description
Agent Precedence	Sets the order by which Service Manager selects agents. Service Manager usually looks for a document entry in the configuration library and when a match is found, the agent specified in that document entry is selected. If no matching document entry is found or agent specified, the engine looks in the input protocol configuration (listener). To have the processing agent taken directly from the listener (thus ignoring the document entry), use <listener> overrides <document>. <document> overrides <listener> is the default value.
Always reply to listener default	If set to *true*, the default reply definition is used in addition to defined destinations.
Error documents treated normally	If set to *true*, error documents are processed by configured preemitters.
Listener is Transaction Manager	If set to *true*, agents run in a local transaction. Agents can roll back uncompleted transactions.
Record in Activity Log(s)	If set to *true*, activity on this channel will be recorded in the activity logs, else the activity will not be recorded.

MQJMS Listener Configuration Example

You are receiving documents on an MQJMS queue, named default, running under a Queue Manager named QM_iwxfoc using a channel named SYSTEM.DEF.SVRCONN. Specify the listener values as follows:

- **Form of Input:** queue
- **Queue Manager:** QM_iwxfoc
- **Channel:** SYSTEM.DEF.SVRCONN
- **Port:** 1414
- **Receiver Name:** default

- **Default Reply:** out



Configuring an MQJMS Emitter

Messages are sent to the destinations at the completion of a workflow. The state of the document determines which destination is used. The order in which the destinations are used cannot be predicted. Destinations cannot accept responses. These configurations also are used to send a message over a different protocol.

Note: Configuring an MQJMS emitter is not required if the emitter protocol used in the outlet of the channel is the same as the listener protocol used in the inlet of the channel. For more information on inlets and outlets, see *Defining an Inlet* on page 15 and *Defining an Outlet* on page 25.

To route an output document to a protocol other than that of the listener, configure an emitter. For example, an application may send input over a TCP listener but want to route the output to an MQJMS queue.

MQJMS Emitter Properties

The following table lists and describes the MQJMS emitter properties. For instructions on creating an emitter, see *How to Create a Listener* on page 18.

Property Name	Property Description
Destination (required)	The queue address queue@queue_manager.

Form of Output	`TOPIC`
	Is for a TopicSession in the Publish and Subscribe domain.
	`QUEUE`
	Is for a QueueSession in the PTP domain.

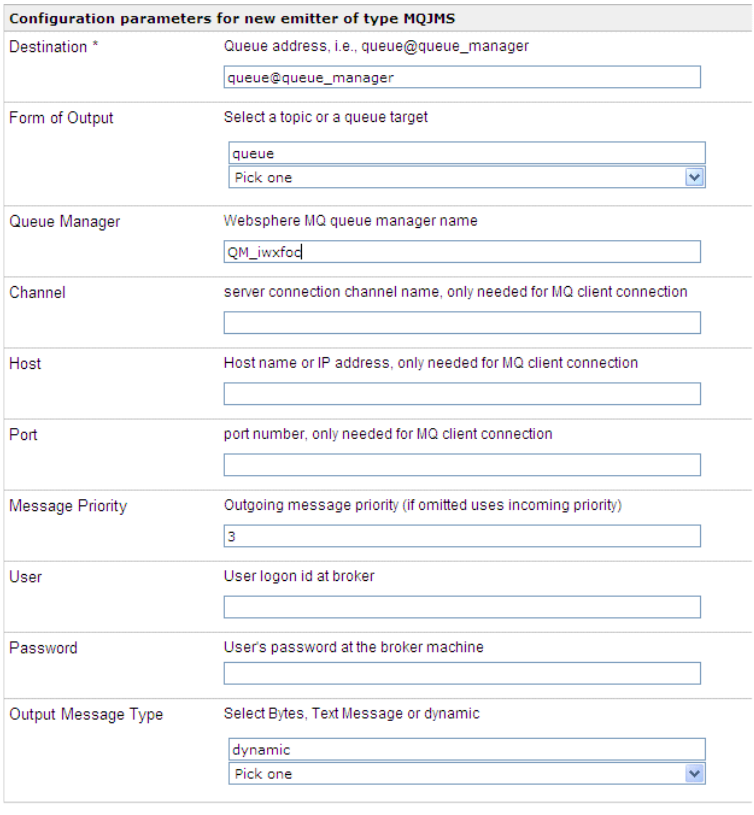
Queue Manager	The Websphere MQ queue manager name.
Channel	The server connection channel name. Required for MQ client connection only.
Host	The host name or IP address. Required for MQ client connection only.
Port	The port number. Required for MQ client connection only.
Message Priority	The outgoing message priority. If omitted, uses incoming message priority.
User	A valid user ID at broker.
Password	A valid password at the broker machine.
Output Message Type	Select *Bytes*, *Text*, or *Dynamic*.

Example: **Local MQJMS Emitter Configuration Example**

You want to emit documents to an MQJMS queue running under a local Queue Manager named QM_iwxfoc. Specify the emitter values as follows:

- **Form of Input:** queue

- **Queue Manager:** QM_iwxfoc



Internal and Ordered Queue Processing

iWay Service Manager (iSM) provides channels that link processes within iSM to other processes in the same or another instance of iSM. The following channel types are available for Internal and Ordered queue processing:

- **Internal.** Passes messages between channels for asynchronous or synchronous execution.
- **Ordered.** Passes messages between channels for asynchronous execution, maintaining execution order and batch control.

For more information on how to configure channels for Internal and Ordered queue processing, see the *iWay Cross-Channel Services Guide*.

Appendix A

Configuring Basic Properties

This section describes how to configure properties for iWay Service Manager (iSM).

In this appendix:

Configuring Properties

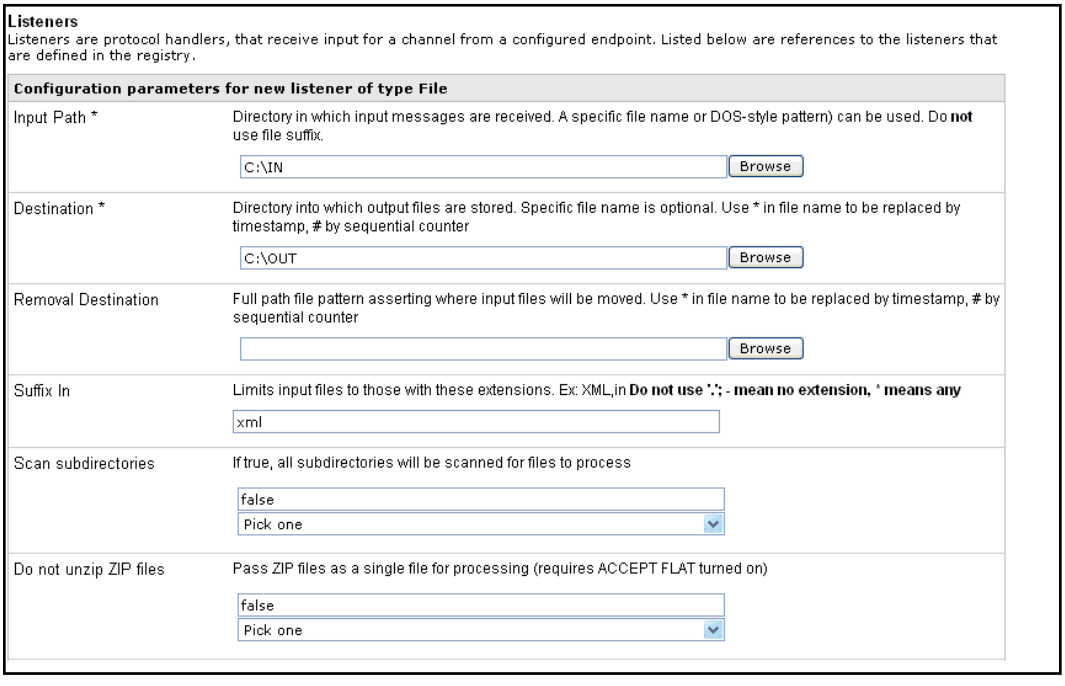
You can provide the properties required to configure iSM in the following ways:

- From a constant value.
- From values stored in a file.
- Using Lightweight Directory Access Protocol (LDAP).
- Using a document-centric query (XPath) or (JsonPath, JsonPointer).
- Using a Special Register (SREG).

Configuring Properties as Constant Values

You can provide configuration properties and store them as constant values.

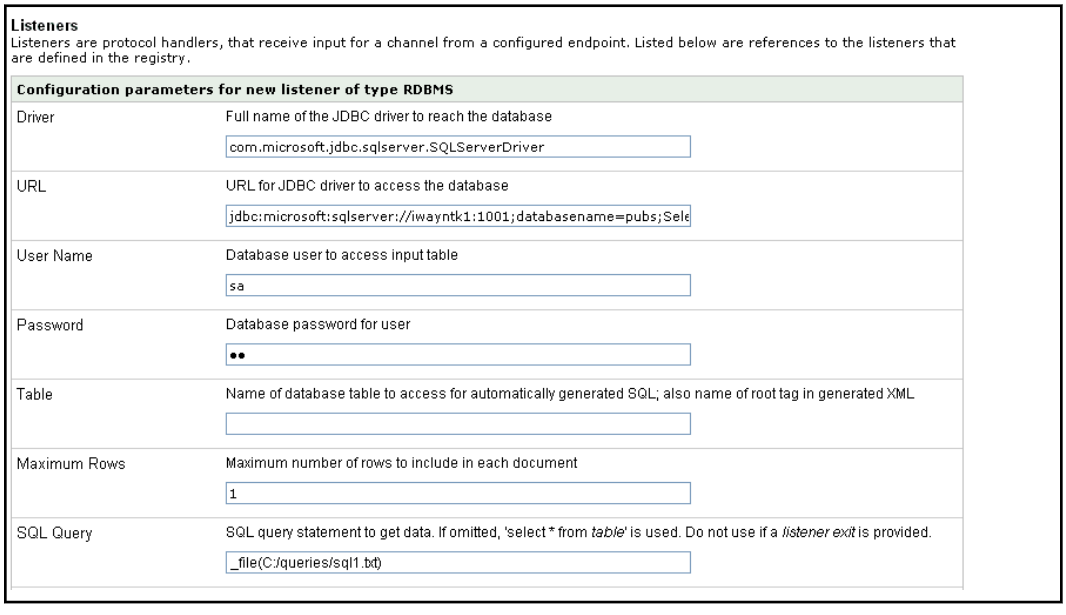
For example, in the Listeners pane, values for the Input Path and Destination for a File listener are displayed.



Obtaining Configuration Properties From the File System

A configuration property can be stored in a file on the file system. This method is useful for large, complex queries used with an SQL listener.

The format is `_file(drive:/directory/filename)`. In the RDBMS listener configuration pane, the query resides in the *queries* directory in a file named *sql1.txt*, as shown in the following image.



Obtaining Configuration Properties Using LDAP

LDAP (Lightweight Directory Access Protocol) is a well-established emerging standard for access to corporate directories, such as Microsoft Active Directory and Novell Directory. You can use LDAP to store security information, for example, user IDs and passwords, and configuration properties.

iWay Service Manager (iSM) supports LDAP for looking up parameters to be used in processing exits. In iSM, processing exits include preparsers, preemitters, and services. The LDAP information is resolved at iSM start-up time.

In LDAP, a directory is subdivided into contexts. Within each context, a filter describes a section of the directory from which an attribute is to be obtained. For example, in the iWay Software context, under the filter of <surname='Smith', system='SmithSystem'>, the attribute password would be Smith's password in SmithSystem.

Using LDAP to store configuration properties offers the following benefits:

- There is no duplication of information in the iSM configuration files.
- Any configuration property can be accessed directly from any LDAP-enabled directory.

- As information in the registry changes, the change is automatically propagated to iSM during the next start up, without reconfiguring iSM.

Using LDAP

You can use an LDAP look-up request for most properties in the iSM configuration. To use LDAP, you must define the LDAP directory to iSM. You enter the LDAP provider URL that identifies the path to the LDAP directory and optionally, a root context, for example:

```
ldap://iwaldap:1234/dc=people, do=etc
```

After it is provided, the initial context is used unless it is overridden in an LDAP access request function elsewhere in the configuration.

Some configurations require that you also enter a valid user ID and password on the LDAP directory server. If you request LDAP access and it is not authorized, you cannot start iSM. LDAP servers that are configured to provide anonymous access do not require a user ID or password.

After you receive authorization to use an identified LDAP context, you can specify the value of any property as:

```
_ldap(filter;attribute_to_get[;context])
```

The context is optional, defaulting to the context set in the initial LDAP access specification. Failure to locate the attribute within the context under the filter results in an empty property value.

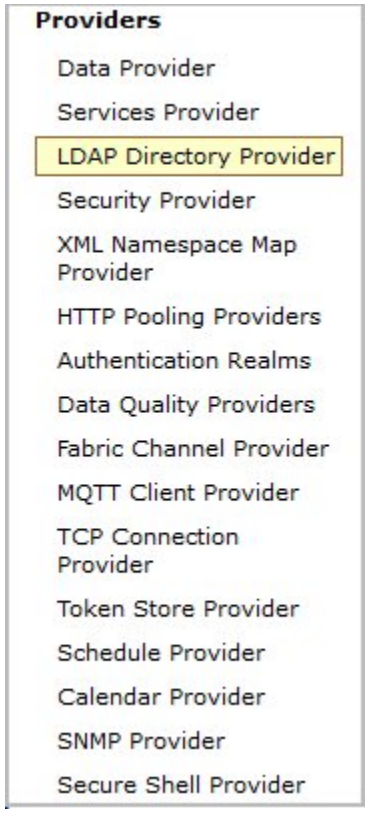
You must configure access to an LDAP server before using LDAP as a means of storing parameters for use by iSM.

Procedure: How to Configure Access to an LDAP Server

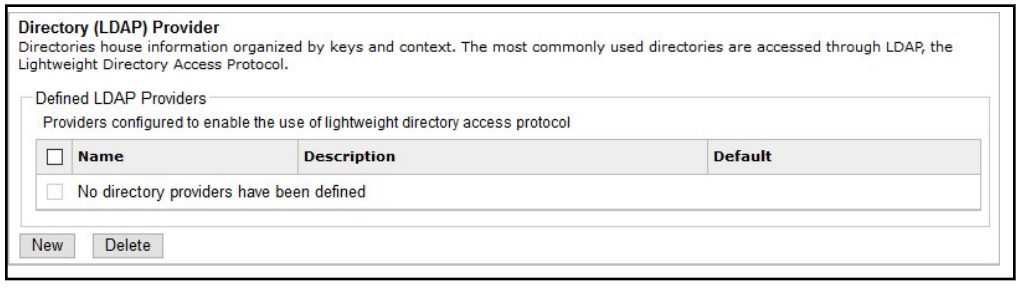
To configure LDAP:

1. In the top pane, click *Server*.

2. From the Providers list in the left pane, click *Directory Provider*.

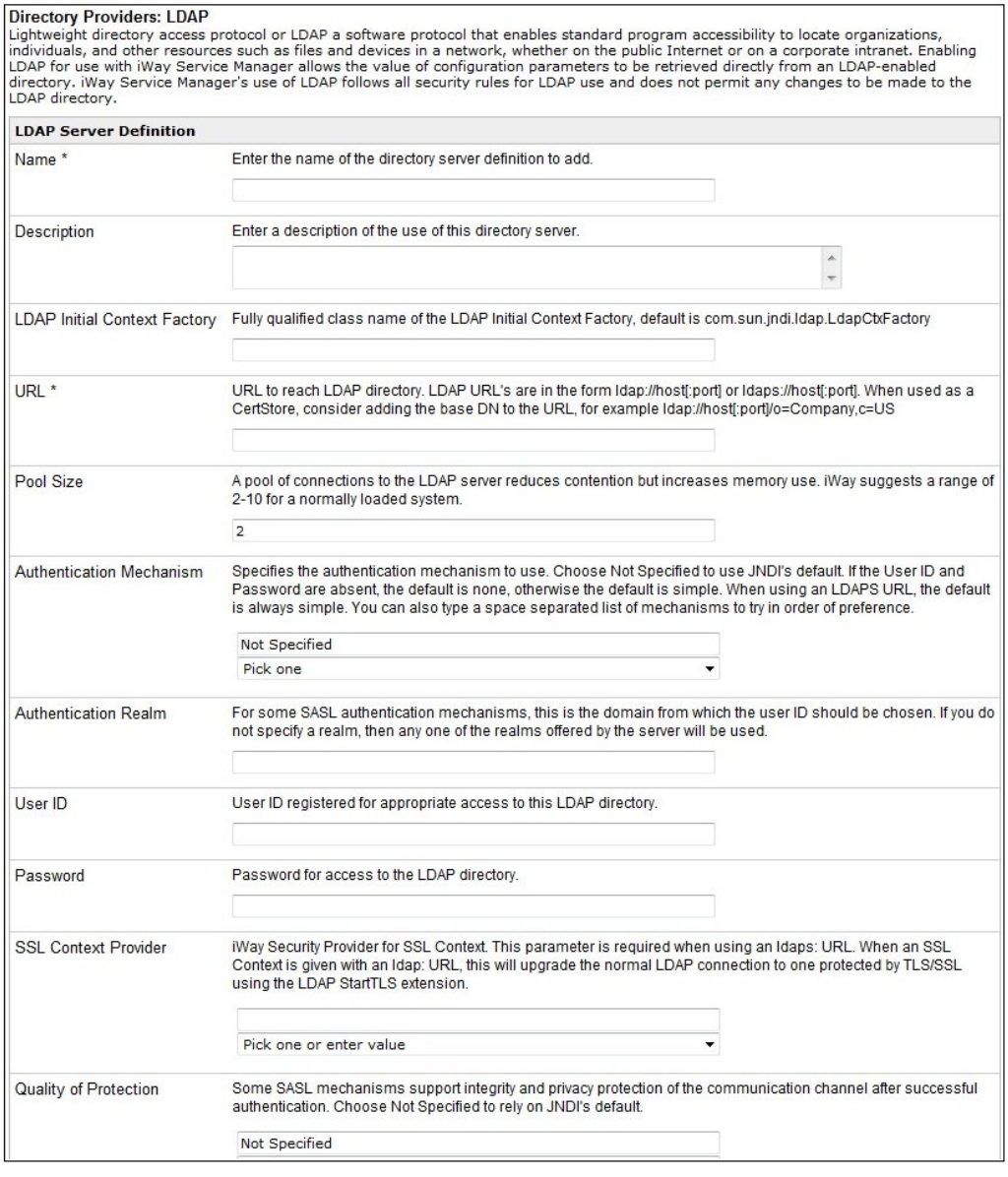


The Directory Provider pane opens.



3. Click *New*.

The Directory Providers: LDAP pane opens, as shown in the following image.



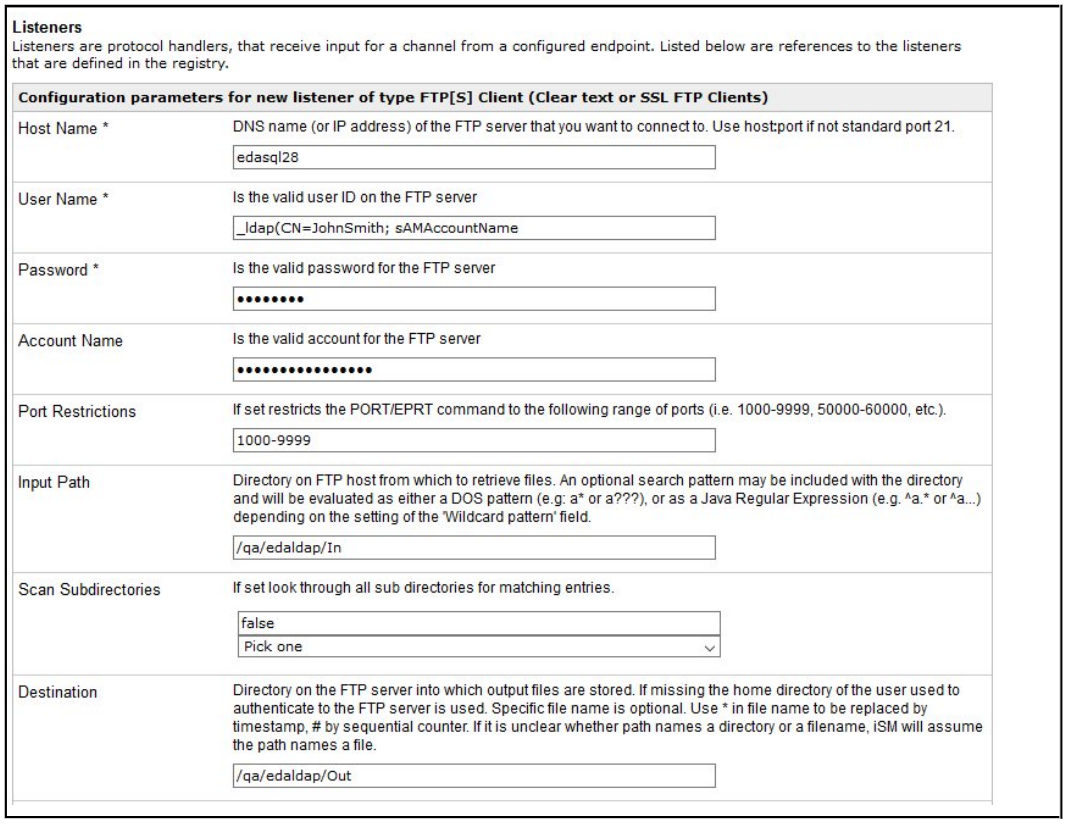
4. Type the property values that are specific to your LDAP server.
5. Click *Test* to verify the provided values, and correct any mistakes if there is a Test Result Failure shown at the top of the pane.

6. When the test is successful, scroll to the bottom of the pane and click *Add*.

Example: **Configuring an FTP Listener Using LDAP**

Any property can be retrieved using LDAP. The following image shows a sample configuration of an FTP listener where the user name and account name are retrieved from LDAP.

```
_ldap(CN=John Smith; sAMAccountName)
```



Obtaining Configuration Properties Using a Document-Centric Query

Using iWay Functional Language (iFL), the internal server provides functions that can address the parsed form of a payload. These functions require a document to be present and in process. This differs from other iFL functions such as _sreg() or _property(), which can access information without a document being present or in process. For XML, the standard Xpath language is provided, while for JSON, JsonPath and JsonPointer are offered.

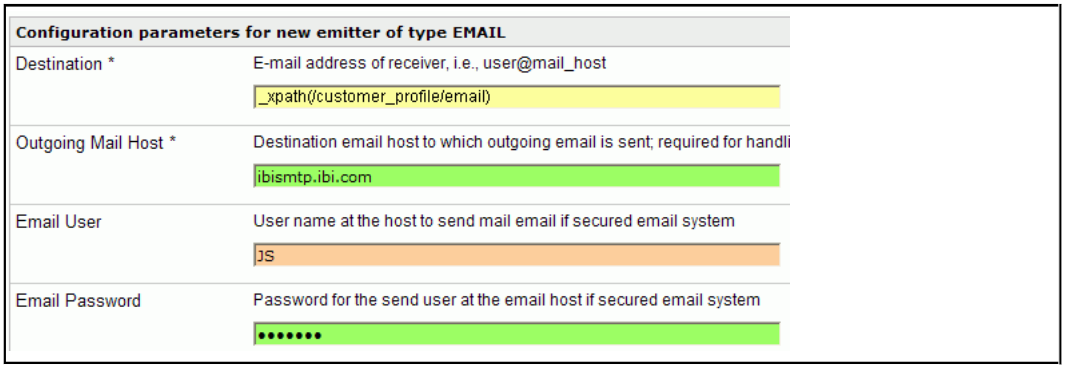
This section describes the use of XPath to route the XML document. However for JSON documents, the _jsonpath() or _jsonptr() functions can be used. For more information on these functions, see the *iWay Functional Language Reference Guide*.

Example: Using XPath to Dynamically Route Output

The <email> node in the following sample code is referenced by iSM to route a reply-to message:

```
<customer_profile>
<name>Joseph Bloggs</name>
<email>Joe_Smith@ibi.com</email>
<vehicles>
   <car>JAGUAR</car>
 </vehicles>
</customer_profile>
```

The XPath notation _xpath(/customer_profile/email) shown in the following image indicates the emitter information derived from the file.



Obtaining Configuration Properties Using Special Registers

Special registers are named tokens that contain information available to services. You can use special registers in:

- Incoming documents
- Parameters
- Configuration values

Special registers are accessed by their token name, _sreg(*sregname*).

Some special registers are completed automatically by iWay Service Manager (iSM) during operation. For example, the correlation ID of a message from a queue (for example, JMS, MQ, or SONIC) can be obtained by _sreg(correlid). An emitter with the property correlid= that is used to set the correlation ID might be set to _sreg(msgid) so that the reply is correlated with the incoming message.

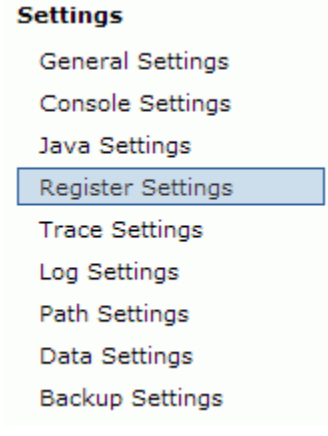
You can define other special registers to the iSM configuration by using the defined special register facility. You can also use a defined special register elsewhere in the configuration and as a value substitution in a document. This feature enables you to define a value, such as a queue manager name, in one place and reference it in another. A system with a dozen MQSeries listeners would then need to change the queue manager name only in one place.

In addition, you can use a special register in the LDAP function as a value for the filter, enabling a search on a complex name without having to spell it out in several places.

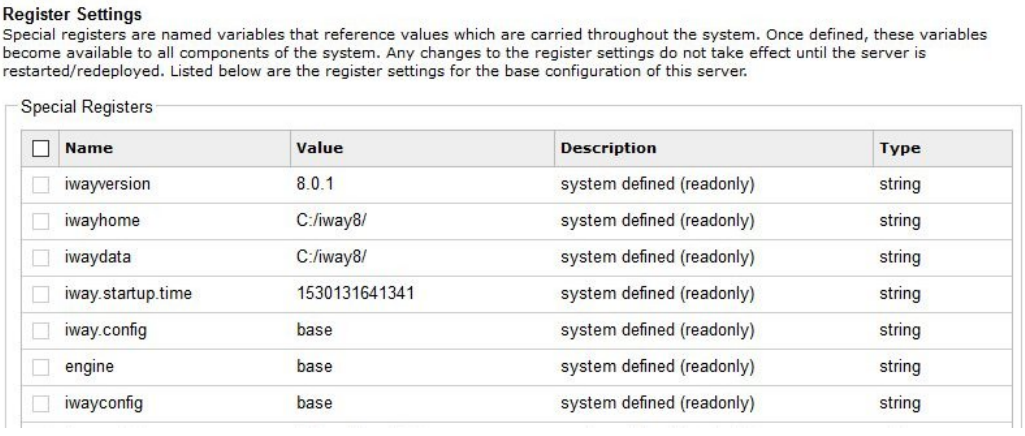
Procedure: How to Display System-Wide Special Registers

To display system-wide special register names and values:

1. In the top pane, click *Server*.
2. From the Settings list in the left pane, click *Register Settings*.



The Register Settings pane opens.



Register Settings

Special registers are named variables that reference values which are carried throughout the system. Once defined, these variables become available to all components of the system. Any changes to the register settings do not take effect until the server is restarted/redeployed. Listed below are the register settings for the base configuration of this server.

Special Registers

Name	Value	Description	Type
iwayversion	8.0.1	system defined (readonly)	string
iwayhome	C:/iway8/	system defined (readonly)	string
iwaydata	C:/iway8/	system defined (readonly)	string
iway.startup.time	1530131641341	system defined (readonly)	string
iway.config	base	system defined (readonly)	string
engine	base	system defined (readonly)	string
iwayconfig	base	system defined (readonly)	string

The name, value, and data type for the special registers in use are displayed.

Procedure: How to Display Special Registers (Register Sets) for a Channel

To display special registers (register sets) for a channel:

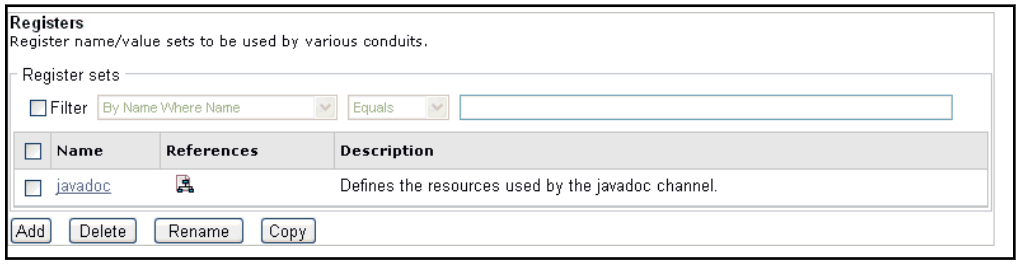
1. In the top pane, click *Registry*.
2. From the Variables list in the left pane, click *Registers*.

The Registers pane opens.

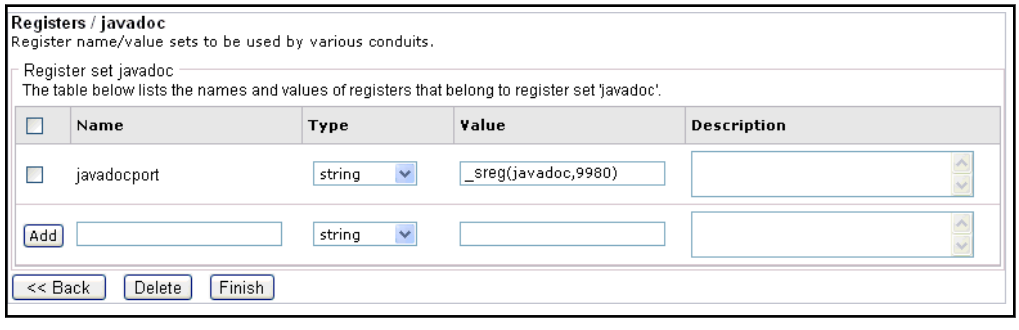
The special register sets used by various conduits are displayed in this pane. In this example, the special register set for a channel called javadoc is shown.

You can use the Registers pane to define groups of registers that can be assigned to channels and process flows (in iIT Designer). These registers may be used to configure components of a channel and process flow, or used by these components at run time in the case when components refer to registers.

3. In the Registers pane, click *javadoc*.



The Register set pane for the javadoc channel opens.

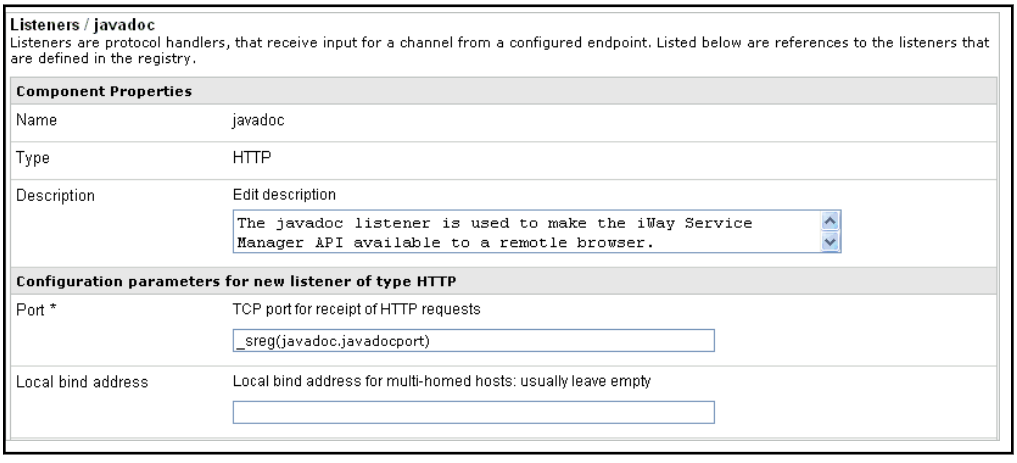


This table lists the names, types, values, and descriptions of registers that belong to a register set. For example, javadocport is a register that has been defined for the javadoc register set.

To see where the javadocport register is actually being used, you can look at the inlet of the javadoc channel, which is an HTTP listener.

To use the register set for a channel, the SREG definition must fully qualify the register set, for example, _sreg(javadoc.javadocport). Notice the period that separates the register set from the actual register name defined within the register set.

The following image shows the Listeners pane for the javadoc listener.



Notice the Port field, which contains the following value:

```
_sreg(javadoc.javadocport)
```

Procedure: How to Use a Special Register

This procedure uses a special register to configure an emitter. For example, to send reply mail to the sender, you can configure an email emitter as follows:

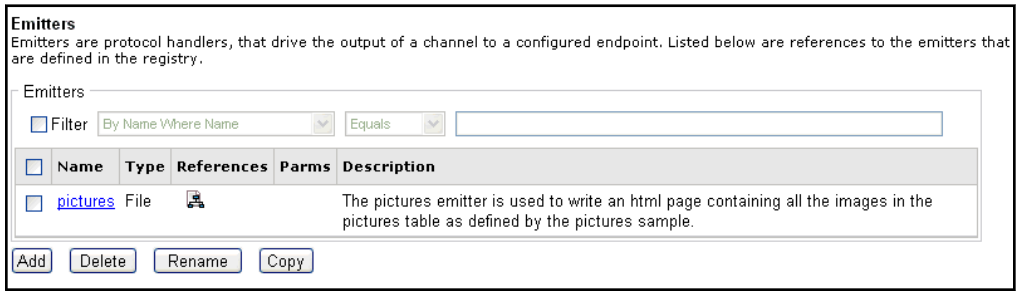
```
Destination=_sreg(from)
```

To use a special register:

1. In the top pane, click *Registry*.
2. From the Components list in the left pane, click *Emitters*.

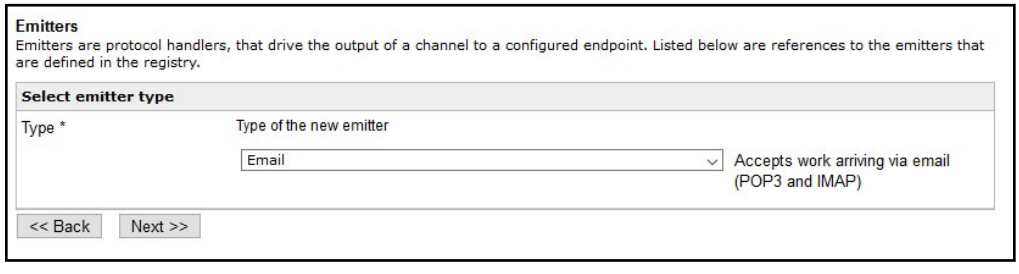
The Emitters pane opens.

3. Click *Add*.

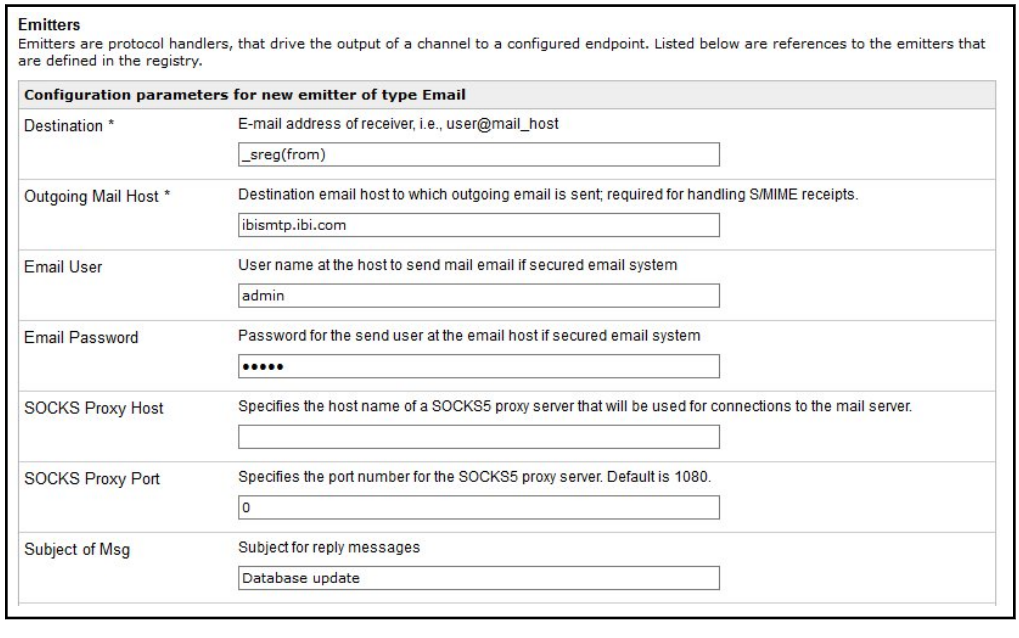


The Select emitter type pane opens.

4. From the Type drop-down list, select *EMAIL* and click *Next*.

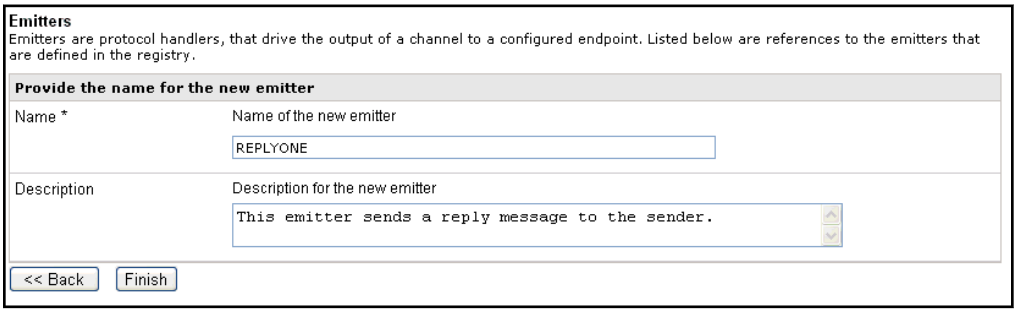


Configuration parameters for the email emitter are displayed.



5. In the Destination field, type *_sreg(from)*.
6. In the Outgoing Mail Host field, specify an email host.
7. Enter appropriate values for the remaining parameters that are applicable to your environment, for example, an email user name and password, and click *Next*.

   The following pane, which contains a Name and Description field for the emitter is displayed.



8. Type a name and description for the emitter, and click *Finish*.

   Once executed, this emitter will send an email message to the recipient using the address defined in the special register _sreg(from).

Enriching a Document With the Content of a Special Register

You can enrich a document by referring to a special register that supplies content for the document. For example, you can update the following document with the name of the sender of the email message.

```
<document>
 <sender></sender>
</document>
```

To place the name of the sender of the email in the document code:

```
<document>
<sender>_sreg(from)</sender>
</document>
```

After execution, the document is enriched as follows:

```
<document>
<sender>John_Smith@iWaysoftware.com</sender>
</document>
```

Processing a document using this method requires that the EvalWalk service be configured and chained. For more information, see the *iWay Service Manager Component and Functional Language Reference Guide*.

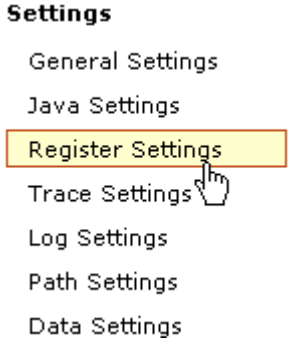
Using Registers, Register Sets, and Parameters

This section explains the differences between registers and register sets in iWay Service Manager and describes how to bind register sets to a channel. In addition, information on how to define parameters is provided.

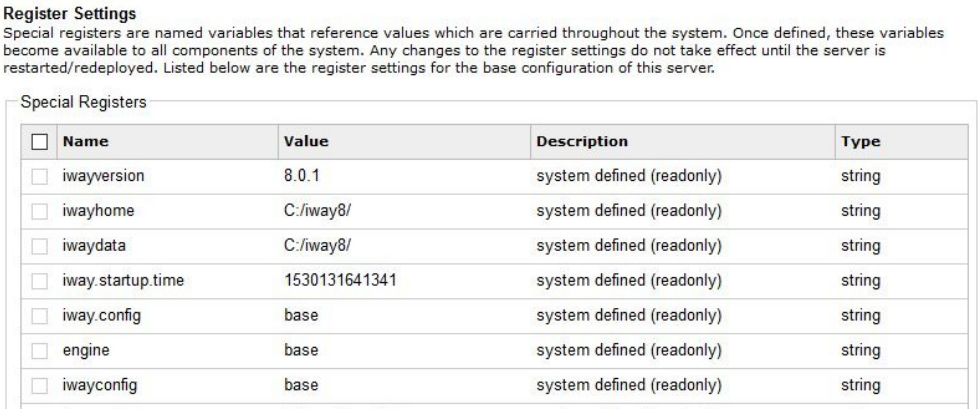
Registers

Registers are global parameter values that are available to any process running within an iWay Service Manager instance. These values are set in the runtime environment and can be used by any iWay Registry component, for example, listener, process flow, adapter, agent, and so on.

The name and value of a register can be configured in the Register Settings pane of the iWay Service Manager Administration Console. To access the Register Settings pane, click *Register Settings* from the Settings list in the left pane of the Server page, as shown in the following image.



The Register Settings pane opens.



Register Settings

Special registers are named variables that reference values which are carried throughout the system. Once defined, these variables become available to all components of the system. Any changes to the register settings do not take effect until the server is restarted/redeployed. Listed below are the register settings for the base configuration of this server.

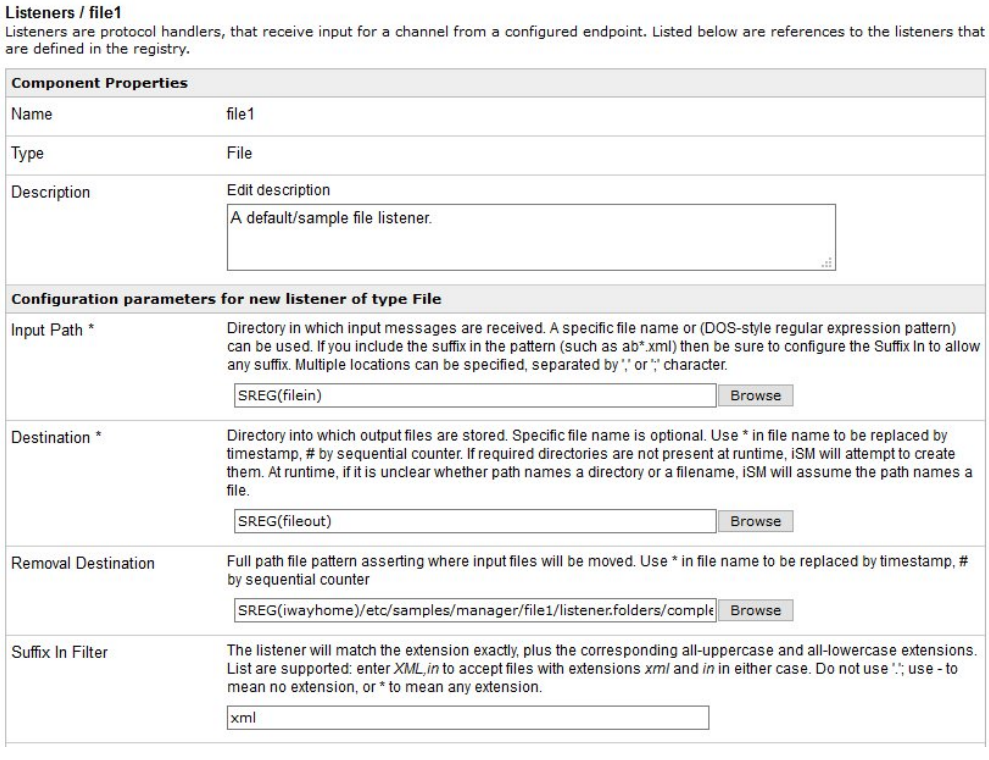
Special Registers

Name	Value	Description	Type
iwayversion	8.0.1	system defined (readonly)	string
iwayhome	C:/iway8/	system defined (readonly)	string
iwaydata	C:/iway8/	system defined (readonly)	string
iway.startup.time	1530131641341	system defined (readonly)	string
iway.config	base	system defined (readonly)	string
engine	base	system defined (readonly)	string
iwayconfig	base	system defined (readonly)	string

The name, value, description, and data type for the available registers are displayed.

Notice the Special Register (SREG) in the list called *iwayconfig*, which is created by default for each iWay Service Manager configuration that is created. For the master configuration called *base*, the _sreg(iwayconfig) would have a value equal to *base*.

The following image shows the configuration parameters for a listener called *file1*. The Input Path field contains an SREG called *filein* to represent the input path and the Destination field contains an SREG called *fileout* to represent the output path.



_sreg(filein) and _sreg(fileout) provide good examples of how registers can be used in iWay Service Manager once they are configured in the Register Settings pane.

Notice that the Removal Destination field contains the following directory path suffix:

```
_sreg(iwayhome)/etc/samples/manager/file1/listener.folders/complete
```

You can also use multiple SREGs to create a path, for example, *_sreg(homedir)* or *_sreg(userdir)*.

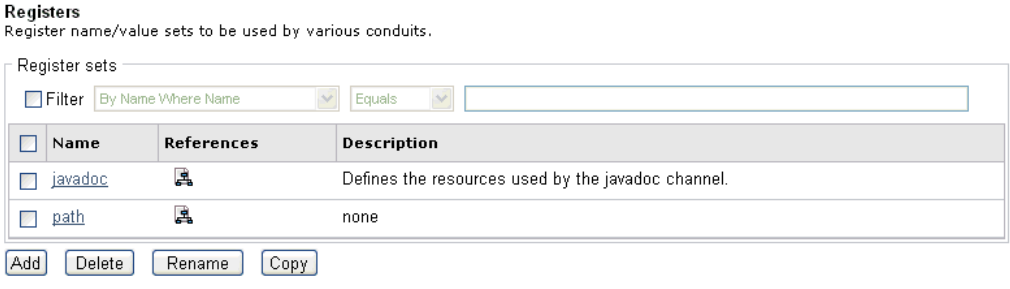
Note: After a new Special Register is created, it is immediately available for use by all iWay Registry components, for example, listener, process flow, adapter, agent, and so on. However, you must perform an iWay Service Manager console restart (warm restart) after editing the value of a Special Register if you want the new value to be used by any associated iWay Registry components.

Register Sets

Register sets are very similar to registers, but they are not global by default. A register set contains a pool of one or more Special Registers (SREGs).

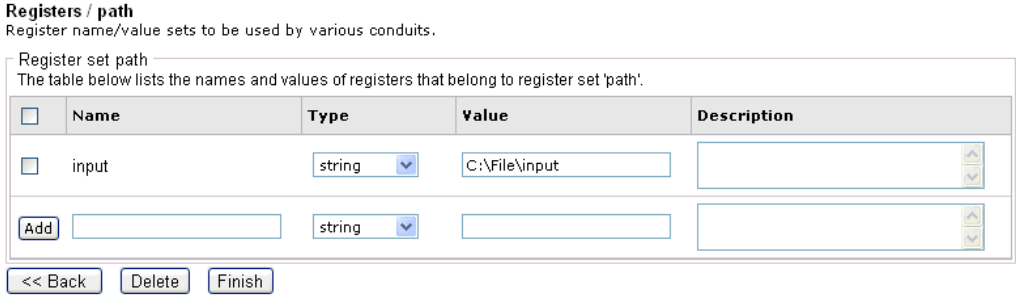
Register sets can be configured in the Registers pane of the iWay Service Manager Administration Console. To access the Registers pane, click *Registers* from the Variables list in the left pane of the Registry page, as shown in the following image.

The Registers pane opens.



The name, reference, and description for the available register sets are displayed. A register set called *javadoc* is created by default. For demonstration purposes, a new register set called *path* has been created. You can click the name of a register set to open a configuration pane for the register set.

For example, the following image shows the Registers / path configuration pane. The name of the register set is *path* and the name of the register that belongs to this register set is *input*. Using this register set configuration pane, you can add multiple registers to a register set as required.



It is important to understand that register sets are used local to a channel and are defined within the component, which is different from an SREG. This is how you must use the register set _sreg(path.input). Also note that in order to use a register set within a channel you must bind the register set to that channel.

Procedure: How to Bind a Register Set to a Channel

To bind a register set to a channel:

1. Click *Registry* in the menu bar, which is located in the top pane.

The Registry - Repository pane opens, showing links to various types of conduits and components you can configure.

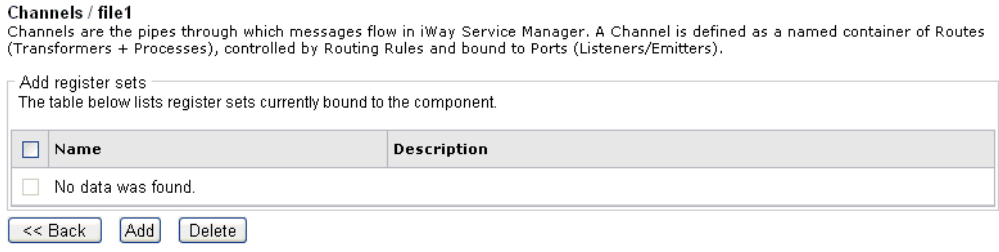
2. Click *Channels*.

The Channels pane opens.

The Regs column (highlighted) shows the number of register sets that are currently bound to each available channel.

3. In the Regs column, click the number for a channel, for example, *file1*.

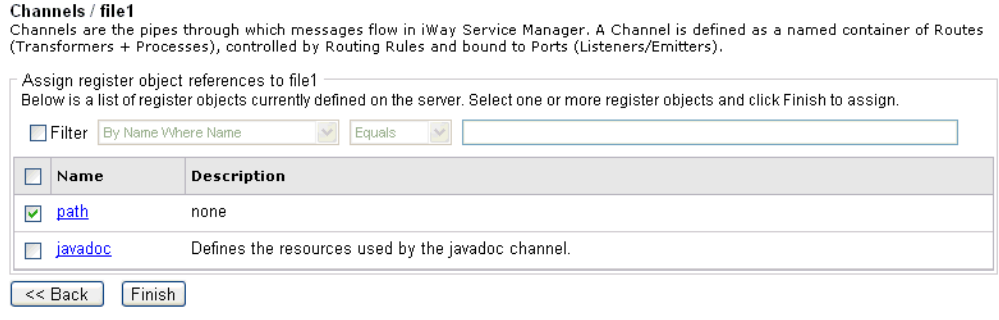
The Add register sets pane opens for the file1 channel, as shown in the following image.



Notice that there are currently no register sets bound to the file1 channel.

4. Click *Add*.

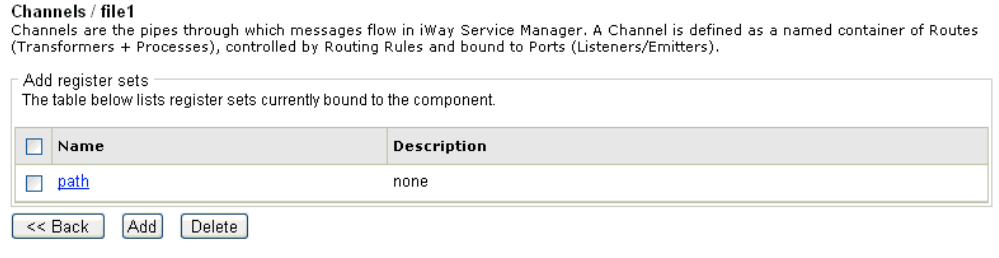
   The Assign register object references to file1 pane opens and provides a list of available register sets.



   Note: Clicking on the name of a register set opens the configuration pane that allows you to modify, add, or remove registers for that register set.

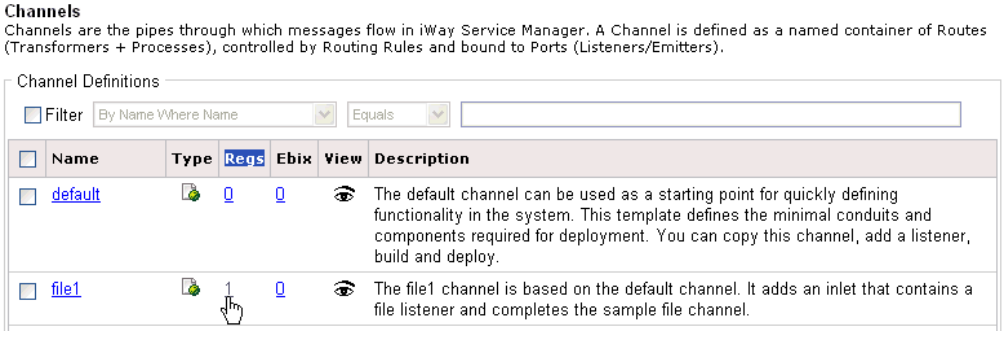
5. Select the check box next to the register set you want to add, for example, *path*.
6. Click *Finish*.

   You are returned to the Add register sets pane for the file1 channel.



Notice that the register set called *path* is now listed for the file1 channel.

When you return to the Channels pane, notice the number 1 that appears in the Regs column for the file1 channel, as shown in the following image.



Note: You must redeploy the channel if changes were made to a register set you are using.

You have successfully bound a register set to a channel in iWay Service Manager.

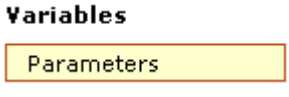
Defining Parameters

The iWay Service Manager Administration Console allows you to define sets of parameters that can be assigned to certain emitters and/or exits. These are dynamic, user-defined parameters that are not described by the metadata of an object. For example, they can be used to provide additional protocol headers.

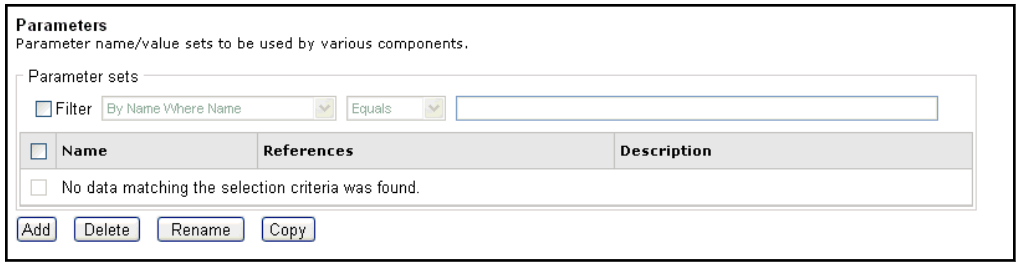
Procedure: How to Define Parameters

To define parameters:

1. In the left console pane of the Registry menu, select *Parameters*.



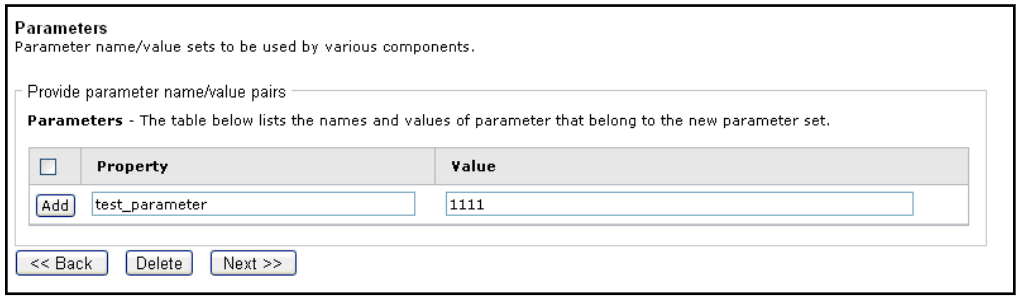
The Parameters pane opens.



The parameter sets used by various conduits are displayed in this pane. Currently, there are no parameter sets defined.

2. Click *Add*.

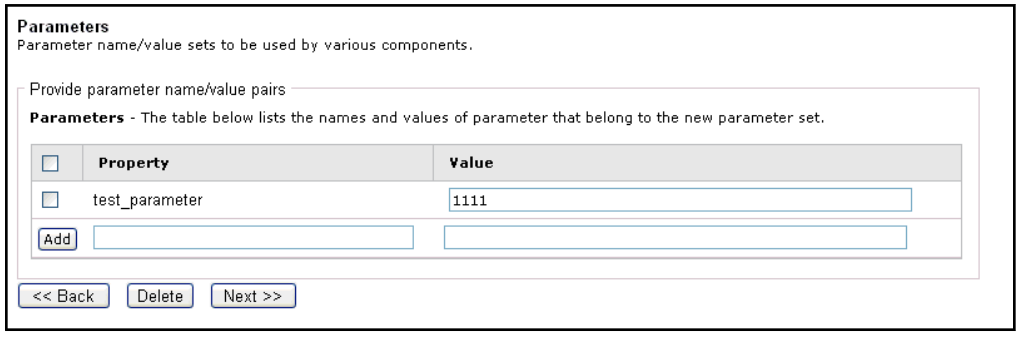
   The Provide parameter name/value pairs pane opens.



   The table that is provided lists the names and values of parameters that belong to the new parameter set.

3. Specify a parameter and a corresponding value.
4. Click *Add*.

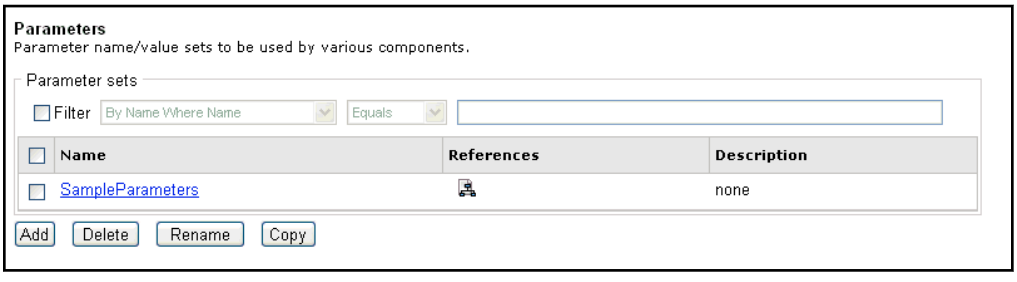
   The parameter is added, as shown in the following image.



   You can add as many parameters as required.

5. Click *Next*.
6. Provide a name and, optionally, a description, for the parameter set, and click *Finish*.

   You are returned to the main Parameters pane, which now includes the new parameter set that was defined.



Appendix B

HTTP Headers and Special Registers

This section provides additional information regarding HTTP headers and special registers.

In this appendix:

History

In earlier versions, all HTTP receive headers were collected as special registers of type HDR. These registers were examined by the process, which decided whether to create new HDR registers or delete existing registers. When an HTTP emit agent was configured, the user had the option to relay headers. If configured to relay, any registers of type HDR found in the context, with a few specific exceptions, were made into outbound headers. If not configured to relay, HDR registers were ignored.

If the emitted HTTP request document received a response with headers, these headers were then converted into special registers of type HDR.

Issue to be Addressed

The existing technique makes it difficult to distinguish information when there are several HTTP emit agents in a flow, or where it is desired to be able to selectively relay header information in a gateway. Currently, the headers sent are filtered only by a pre-set list of registers not to be sent.

For gateway flows, it is important to be able to group header information and to select and operate upon these groups.

Special Registers and HTTP

Special register names can be preceded with a namespace prefix. The namespace is simply the first part of the name before the first decimal. For example, register *xyz* is in the default namespace, while register *abc.xyz* is in the *abc* namespace. This notation is compatible with registers defined in the iWay Service Manager SP1 registry as well as existing notation. IWAF context registers also follow this pattern, as does the iway.home register, which can be considered as the home in the iWay namespace. Only register names that do not contain any decimals are namespace-less, which is to say in the default namespace.

The HTTP/ASx listener is extended to place all incoming header information into a configured namespace. If no namespace name is specified, the default namespace (none) is be used. The HTTP/ASx emitter and the associated agents have new configuration parameters, which are listed and described in the following table.

Parameter	Role
request namespace	The output headers are taken from the configured namespace. This is a combo box, enabling a user to select a pre-defined setting or enter the name of a specific namespace as the source of headers. Selections and their meaning are: ❑ [listener] - Select from the namespace configured on the listener. **Note:** The [listener] option is not applicable to the listener itself. ❑ [default] - Select from the default namespace. The default namespace is one with no namespace prefix. ❑ [none] - Send no headers from special registers, only send those generated by this emitter.

Parameter	Role
response namespace	If there is a response with headers, then the incoming headers are placed in this namespace. The selections and their meaning are: ❏ [listener] - Use the namespace configured as the response namespace on the listener. **Note:** The [listener] option is not applicable to the listener itself. ❏ [default] - Use default namespace. The default namespace is one with no namespace prefix. ❏ [none] - Create no special registers from the input context, only send those generated internally by this emitter.
Excluded Headers	This is a comma delimited list (case-insensitive) of headers that should not be sent with the response, even if they are found in the response header namespace.

A new agent called XDSREGNamespaceAgent, is available to perform the following operations on registers in namespaces:

- Copy - Duplicates registers from a source namespace to a destination namespace. After a copy operation is performed, the registers are available in both namespaces.
- **Move.** Moves registers from one namespace to another.
- **Delete.** Deletes all registers in the namespace.
- **Exist.** If any registers exist in the named namespace, pass the flow down the success edge, else down the notfound edge.

The XDSREGNamespaceAgent offers *from name* and *to name* combo-box parameters. Each of these parameters offers the [listener] and [default] options which are interpreted as described above.

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

TIBCO, the TIBCO logo, the TIBCO O logo, FOCUS, iWay, Omni-Gen, Omni-HealthData, and WebFOCUS are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle Corporation and/or its affiliates.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the readme file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

Index

H

I

J

L

M

N

O

Q

R

S

T

W