IBM MessageSight

IBM MessageSight Protocol Plug-in SDK (Technical Preview)

Version 1 Release 2

Note

Before using this information and the product it supports, read the information in “Notices” on page 27.

This edition applies to version 1 release 2 of IBM MessageSight and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Developing a protocol plug-in

You can use protocol plug-ins to add industry-specific messaging protocols to IBM® MessageSight in a controlled way that does not compromise the security of the appliance environment. The protocol plug-in can also support legacy sensors that are not able to change to MQTT. You can write your own protocol plug-ins to use with IBM MessageSight by using the IBM MessageSight protocol plug-in SDK. The protocol plug-in facility is not available in high availability environments.

Before you begin

Download and extract the contents of the SDK bundle. Then, follow the instructions in `ImaToolsPreview/ImaPlugin/README.txt` to import the sample plug-in projects that are provided with the SDK bundle.

Download and install one of the following supported versions of Java™:

- IBM SDK, Java Technology Edition, Version 7 (64-bit)
- Oracle Java SE Development Kit, Version 7 (64-bit)

About this task

You can use the new IBM MessageSight protocol plug-in SDK to extend IBM MessageSight protocol support beyond the protocols that are natively supported. Use the SDK to write Java plug-ins that you can then deploy in IBM MessageSight.

If you want to write your own protocol plug-in to use with IBM MessageSight, you need to develop the following files:

- A plug-in configuration file that is written in JSON
- A set of JAR files to implement the plug-in

All of these files must be at the root level of the zip archive.

Procedure

1. Implement the ImaPluginListener and ImaConnectionListener interfaces and create one or more JAR files that contain the classes that you implemented.

   ImaPluginListener interface.

   The ImaPluginListener interface provides callbacks that allow custom protocol implementations to be associated with an `ImaPlugin` object that is provided by IBM MessageSight. Refer to the Javadoc information that is provided with the SDK for complete details about the methods that need to be implemented for these interfaces.

   - The `initialize()` method is used to start the custom protocol implementation when IBM MessageSight is started.
   - The `terminate()` method is used to stop the custom protocol implementation when IBM MessageSight shuts down.
   - The `startMessaging()` method is used to run any protocol-specific tasks that are required after the IBM MessageSight messaging engine is started.
   - The `onProtocolCheck()` method is used to determine whether a connection belongs to the protocol. If a connection belongs to the protocol, the ImaPluginListener enables the protocol to instantiate an implementation of the ImaConnectionListener interface with the `onConnection()` method that associates the newly accepted `ImaConnection` object with the protocol.

ImaConnectionListener interface.

The ImaConnectionListener interface allows a physical connection to be associated with a custom protocol by using the `ImaConnection` object that is received from IBM MessageSight. The ImaConnectionListener interface provides callbacks for managing connections to the custom protocol after the protocol plug-in accepts a connection.

- The `onConnected()` method is used to take appropriate actions when a connection is established.
- The `onClose()` method is used to take appropriate actions when a connection is closed.
- The `onData()` or `onHttpData()` method is used to manage communication activities such as handling incoming data.
- The `onMessage()` method is used to handle messages that are received from IBM MessageSight.

2. Create a `plugin.json` descriptor file for the new plug-in.

The `plugin.json` descriptor file provides information to IBM MessageSight about the plug-in so that it can be started when IBM MessageSight is started. While the descriptor supports several configuration properties, four property values must be specified for any protocol plug-in that you implement. These values are **`Name`**, **`Protocol`**, **`Class`**, and **`Classpath`**.

- The **`Name`** property specifies the plug-in name and must be unique.
- The **`Protocol`** property specifies a protocol family. Each plug-in represents a protocol. However, multiple plug-ins can be associated with a single protocol family. Protocol families allow multiple protocols to share authorization rules that are based on endpoint configuration and on connection and messaging policies. For example, MQTT over TCP and MQTT over WebSockets protocols share the same authorization rules as MQTT.
- The **`Class`** property must contain the name of the class that implements the ImaPluginListener interface. This class name must include the complete, dot separated path to the class.
- The **`Classpath`** property must include the full list of JAR files that implement the protocol plug-in that is represented as a JSON array of strings. All JAR files that are used by the plug-in must appear in the **`Classpath`** property so that they can be loaded by IBM MessageSight.

The following table shows the complete list of properties that can be used in the descriptor file:

Table 1. Plug-in configuration file properties.

Name	**Type**	**Default**	**Description**
Class	String	required	The name of the initial class to load for a plug-in. This class must be an instance of **`ImaPluginListener`**. This value must be a valid Java package name and class name, which is separated by dots.
Classpath	String array	required	A set of JAR files that contain the Java classes that are needed for the plug-in. The path must not be included, and the JAR files must be in the root directory of the zip file that is used to define the plug-in. These JAR files are used only by this plug-in, and use a separate directory and class loader for each plug-in. The JAR files in the list must exist in the zip file that is used for installation.
InitialByte	Array	none (required for TCP connections)	A set of initial bytes that can be specified as an array of strings of length 1 byte or as integers of the value 0-255. A single entry with the value `ALL` indicates that any initial byte is selected. If this value is not specified or the array is empty, then TCP connections are not accepted.
Name	String	required	The name of the plug-in. This name must be unique among all installed plug-ins. The name is limited to 64 characters and must be a valid Java name. It can start with any alphabetic character, currency symbol, or underscore, and continue with any such character or a digit.

Table 1. Plug-in configuration file properties (continued).

Name	Type	Default	Description
Properties	Object	none	A set of properties that are sent to the plug-in as configuration. The names and types of the properties are not known to the IBM MessageSight server
Protocol	String	required	The protocol family against which to authorize. Each plug-in represents a single protocol family, but the same protocol family can be used by several plug-ins. The plug-in can also use one of the system protocol families. All policy checking for protocol is done based on this protocol. The maximum length of the name is 32 characters.
UseBrowse	Boolean	false	The protocol uses queue browser function.
UseQueue	Boolean	false	The protocol uses queue send and receive.
UseTopic	Boolean	true	The protocol uses publish/subscribe topic support.
WebSocket	String array	none (required for WebSockets connections)	An array of WebSockets subprotocols supported by this plug-in. The subprotocols are checked in a case independent manner and must be unique among the installed plug-in set. If this property is missing or the array is empty, then no WebSockets protocol can connect to this plug-in. The maximum length of the name is 64 characters. The name must contain only ASCII-7 characters not including control characters (`0x00` to `0x1F` and `x07F`), space, or the separator characters `'()<>[]{},;:\/?="`.

3. Create a zip archive that contains the JAR file or files and the `plugin.json` descriptor file.

   Note:

   To deploy your protocol plug-in, you must create a zip archive file that contains the JAR file (or files) from step 1 and the `plugin.json` file from step 2. All of these files must appear at the root level of the zip archive so that they can be loaded.

Example

An example of a plug-in configuration file for JSON-based messaging.

```
/*
 * Sample plug-in configuration file for JSON based messaging
 */
 {
    "Name":        "json_msg",
    "Protocol":    "json_msg",
    "Classpath":   [ "jsonprotocol.jar" ],
    "Class":       "com.ibm.ima.samples.plugin.jsonmsg.JMPlugin",
    "WebSocket":   [ "json-msg" ],
    "InitialByte": [ "{" ],  /* The json_msg always starts with a JSON object */
    "UseQueue":    false,    /* This plug-in does not implement queues */
    "UseTopic":    true,     /* This plug-in implements topics */
    "Properties":  {
        "Debug": true
    }
 }
```

What to do next

Apply your plug-in to IBM MessageSight. For more information about installing a protocol plug-in, see Installing a protocol plug-in by using the command line.

Debug environments for the IBM MessageSight protocol plug-in SDK

You can debug the protocol plug-in that you are developing in two ways. You can choose to run the protocol plug-in server in Eclipse, or you can run the plug-in server and the plug-in on your IBM MessageSight virtual appliance. Both of these debugging methods are only supported in the virtual appliance environment available in IBM MessageSight for Developers.

By using the plug-in process classes in the `imaPlugin.jar` that is included in the SDK bundle, you can debug your protocol plug-in. This JAR file enables two methods of debugging protocol plug-ins: Running the protocol plug-in server in Eclipse, or running the plug-in server and the plug-in on your IBM MessageSight virtual appliance.

If you run the protocol plug-in server in Eclipse, you can run the protocol plug-in server process remotely and use the IBM MessageSight virtual appliance to forward messages to the external plug-in server process. This approach is intended for use during the early stages of development. With this approach, you can deploy a plug-in zip file that contains only a `plugin.json` descriptor file and no JAR files. Then, you can debug and update your plug-in classes in your local development environment without redeploying a plug-in zip file on the virtual appliance. When you use this method of debugging, you must use a non-standard version of the `plugin.json` file. This version of the file provides the class path for your plug-in classes in your development environment.

After the first phase of development is finished, you must test your production protocol plug-in by installing it and running it on your IBM MessageSight virtual appliance. If you encounter unexpected problems while you run in that environment, you can use the second debugging approach to connect to the plug-in server that runs on the virtual appliance. When you debug during this phase, you must create a new plug-in zip file and reinstall it on the IBM MessageSight virtual appliance each time that you make updates to your plug-in classes.

Debugging a protocol plug-in by running the plug-in server in Eclipse

You can debug your protocol plug-in by running the plug-in server in Eclipse.

Before you begin

Download and extract the contents of the SDK bundle. Then, follow the instructions in `ImaToolsPreview/ImaPlugin/README.txt` to import the sample plug-in projects that are provided with the SDK bundle.

About this task

If you run the protocol plug-in server in Eclipse, you can run the protocol plug-in server process remotely and use the IBM MessageSight virtual appliance to forward messages to the external plug-in server process. This approach is intended for use during the early stages of development. With this approach, you can deploy a plug-in zip file that contains only a `plugin.json` descriptor file and no JAR files. Then, you can debug and update your plug-in classes in your local development environment without redeploying a plug-in zip file on the virtual appliance. When you use this method of debugging, you must use a non-standard version of the `plugin.json` file. This version of the file provides the class path for your plug-in classes in your development environment.

In later stages of development, you must test your production protocol plug-in by installing it and running it on your IBM MessageSight virtual appliance. For more information, see "Debugging a protocol plug-in by running the plug-in server and the plug-in on the virtual appliance" on page 6.

Procedure

1. Run the protocol server in Eclipse:
   a. From the **Run** menu, select **External Tools > External Tools Configurations...**

   b. Right-click **Program** and select **New**
   c. In the **Name** field, enter `ImaPluginServerInEclipse`
   d. In the **Location** field, enter the full path to the Java executable that starts the plug-in server.

      For example, on Windows, your path might look like the following path: `C:\eclipse\jdk\jre\bin\java.exe`
   e. In the **Arguments** field, enter the following arguments to run the protocol plug-in server: `-Xdebug -Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=n -jar` *`path_to_SDKInstall`*`/ImaTools/ImaPlugin/lib/imaPlugin.jar -i @`

      Note:
      - You must enter the correct value for *path_to_SDKInstall* for your environment.
      - The plug-in server uses port 9091. You must ensure that this port is available before you start the process.
      - The value that is specified for address must match the port value that is specified for the Eclipse remote debugger configuration. This port value is specified in step 2e.
   f. Click **Apply**
   g. Click **Run** to start the protocol plug-in server.
2. Start the Eclipse remote Java debugger for the plug-in server that is running in Eclipse:
   a. From the **Run** menu, select **Debug Configurations...**
   b. Right-click **Remote Java Application** and select **New**
   c. In the **Name** field, enter `ImaPluginOnEclipse`
   d. In the **Host** field, enter the IP address of the host where you are running Eclipse
   e. In the **Port** field, enter `8000`

      Note: This port value must match the value that was specified for the address in the arguments for running the plug-in server in Eclipse. This value was specified in Step 1e.
   f. Select the **Source** tab and click **Add**
   g. Select **Java Project** and click **OK**
   h. Select the project where the plug-in that you want to debug is and click **OK**
   i. Click **Apply**
   j. Click **Debug**
3. To use the plug-in server that is running in Eclipse to debug your plug-in, you must create a debug version of your `plugin.json` descriptor file. Set the `Classpath` property to the location of the compiled classes for the plug-in that you are debugging in Eclipse.

   For example, to debug the sample plug-in that is provided in the SDK bundle, set the `Classpath` to *`path_to_SDKInstall`*`/ImaTools/ImaPlugin/samples/jsonmsgPlugin/bin`

   Note: If you run Eclipse on Windows, you must either convert the backslashes (\) that are used in the path to forward slashes (/), or use double backslashes (\\).
4. Create a zip file that contains only the debug `plugin.json` file in the root directory of the zip.
5. Install the zip file on your IBM MessageSight virtual appliance. For more information, see "Configuring protocol plug-ins" on page 9.
6. Configure your IBM MessageSight virtual appliance to forward messages to the debug plug-in server that you are running on Eclipse:
   a. Ensure that the `PluginRemoteDebugPort` property is not set by entering the following command:

      ```
      imaserver set "PluginRemoteDebugPort="
      ```
   b. Set the `PluginDebugServer` property to the IP address of the host where Eclipse is running the plug-in server by entering the following command:

      ```
      imaserver set "PluginDebugServer=IPAddress"
      ```

Where *IPAddress* is the IP address of the host where Eclipse is running the plug-in server.

Tip: You can check the value of the `PluginDebugServer` property by using the **`imaserver get PluginDebugServer`** command.

   c. Stop and restart the server by entering the following commands:

```
imaserver stop
imaserver start
```

7. Set breakpoints in your plug-in source code and debug it by using the client applications for the protocol that is implemented in your plug-in.
8. Optional: If you stop the IBM MessageSight server after you start debugging, or if you stop the plug-in server process or the remote Java debugger, you must restart all three processes:
   a. In Eclipse, from the **Run** menu, select **External Tools**
   b. Select the `ImaPluginServerInEclipse` configuration that you created.
   c. From the **Run** menu, select **Debug Configurations..**
   d. Find **Remote Java Application** and select `ImaPlugInOnEclipse`. Then, click **Debug**.
   e. On the IBM MessageSight appliance, stop and restart the server by entering the following commands:

```
imaserver stop
imaserver start
```

What to do next

1. When you complete this phase of debugging, reset the `PluginDebugServer` parameter on the IBM MessageSight virtual appliance. Enter the following command:

```
imaserver set "PluginDebugServer="
```

2. Move to the next phase of debugging, by running the plug-in server and your plug-in on an IBM MessageSight virtual appliance. For more information, see "Debugging a protocol plug-in by running the plug-in server and the plug-in on the virtual appliance."

Debugging a protocol plug-in by running the plug-in server and the plug-in on the virtual appliance

You can debug your protocol plug-in by running the plug-in server and your plug-in on your IBM MessageSight virtual appliance.

Before you begin

Download and extract the contents of the SDK bundle. Then, follow the instructions in `ImaToolsPreview/ImaPlugin/README.txt` to import the sample plug-in projects that are provided with the SDK bundle.

About this task

After the first phase of development is finished, you must test your production protocol plug-in by installing it and running it on your IBM MessageSight virtual appliance. If you encounter unexpected problems while you run in that environment, you can use the second debugging approach to connect to the plug-in server that runs on the virtual appliance. When you debug during this phase, you must create a new plug-in zip file and reinstall it on the IBM MessageSight virtual appliance each time that you make updates to your plug-in classes.

For more information about how to debug your protocol plug-in in the first phase of development, see "Debugging a protocol plug-in by running the plug-in server in Eclipse" on page 4.

Procedure

1. Deploy the plug-in on the IBM MessageSight virtual appliance and configure the appliance to enable remote debugging from Eclipse:
   a. Create a standard zip archive file with the `plugin.json` descriptor file and the JAR files that implement the plug-in. For more information, see "Developing a protocol plug-in" on page 1.
   b. Install the plug-in zip file on your IBM MessageSight virtual appliance. For more information, see "Configuring protocol plug-ins" on page 9.
   c. Ensure that the `PluginDebugServer` property is not set by entering the following command:
   ```
   imaserver set "PluginDebugServer="
   ```
   d. Set the `PluginRemoteDebugPort` property to the port that Eclipse is using for remote debugging by entering the following command on the IBM MessageSight command line:
   ```
   imaserver set "PluginRemoteDebugPort=8000"
   ```

   Note: The value that is specified for the `PluginRemoteDebugPort` must match the port value that is specified for the Eclipse remote debugger configuration in Step 2e.

   Tip: You can check the value of the `PluginRemoteDebugPort` property by using the **`imaserver get PluginRemoteDebugPort`** command.
   e. Stop and restart the server by entering the following commands:
   ```
   imaserver stop
   imaserver start
   ```
2. Start the Eclipse remote Java debugger for the plug-in server that runs on the IBM MessageSight virtual appliance.
   a. In Eclipse, from the **Run** menu, select **Debug Configurations...**
   b. Right-click **Remote Java Application** and select **New**
   c. In the **Name** field, enter `ImaPluginOnMessageSight`
   d. In the **Host** field, enter the IP address of the virtual appliance
   e. In the **Port** field, enter `8000`

   Note: This port value must match the `PluginRemoteDebugPort` that was set for the IBM MessageSight virtual appliance in step 1d.
   f. Select the **Source** tab and click **Add**
   g. Select **Java Project** and click **OK**
   h. Select the project where the plug-in that you want to debug is and click **OK**
   i. Click **Apply**
   j. Click **Debug**
3. Set breakpoints in your plug-in source code and debug it by using the client applications for the protocol that is implemented in your plug-in.
4. Optional: If you stop the IBM MessageSight server after you start debugging, or if you stop the remote Java debugger, you must restart both processes:
   a. On the IBM MessageSight appliance, stop and restart the server by entering the following commands:
   ```
   imaserver stop
   imaserver start
   ```
   b. In Eclipse, from the **Run** menu, select **Debug Configurations..**
   c. Find **Remote Java Application** and select `ImaPlugInOnMessageSight`. Then, click **Debug**.

What to do next

- When you complete this phase of debugging, reset the `PluginRemoteDebugPort` parameter on the IBM MessageSight virtual appliance. Enter the following command:

  ```
  imaserver set "PluginRemoteDebugPort="
  ```

Configuring protocol plug-ins

The protocol plug-in provides the ability to extend the protocols that are supported by the IBM MessageSight appliance. You can configure protocol plug-ins by using the IBM MessageSight command line.

You can use protocol plug-ins to add industry-specific messaging protocols to IBM MessageSight in a controlled way that does not compromise the security of the appliance environment. The protocol plug-in can also support legacy sensors that are not able to change to MQTT.

The protocol plug-in facility is not available in high availability environments.

Configuring protocol plug-ins by using the command line

Configure protocol plug-ins by using the **`file get`**, **`imaserver create Plugin`**, **`imaserver update Plugin`** and **`imaserver delete Plugin`** commands on the command line.

Installing a protocol plug-in by using the command line

Import and install a new or updated protocol plug-in by using the **`file get`**, **`imaserver create Plugin`**, and **`imaserver update Plugin`** commands on the command line.

Before you begin

To run a protocol plug-in, you must deploy the plug-in in IBM MessageSight. The plug-in must be archived in a zip file. This zip file must contain the JAR file (or files) that implement the plug-in for the target protocol. The zip file must also contain a JSON descriptor file that describes the plug-in content. This descriptor file is required in the zip archive that is used to deploy a protocol plug-in. For more information about developing a protocol plug-in, see “Developing a protocol plug-in” on page 1.

Procedure

1. Import the plug-in zip file to IBM MessageSight by entering the following command:

   ```
   file get zipfileURL localFile
   ```

   where

   - **zipfileURL**

     Specifies the location of the plug-in file in the following format: `scp://username@hostname:/filepath`

     For example, `scp://user@192.0.2.32:/tmp/jsonmsg.zip`

   - **localFile**

     Specifies the name to use for the file when it is uploaded to IBM MessageSight.

     The name cannot include any path elements.

     You can use a period (.) to specify that the name of the uploaded file matches the name of the file on the remote server.

2. Optional: Verify that the file is correctly loaded on the appliance by entering the following command:

   ```
   file list
   ```

   The file is displayed in the resulting list.

3. Create the plug-in by entering the following command:

   `imaserver create Plugin "File=`*`zipfilename`*`" "PropertiesFile=`*`propertiesfilename`*`"`

   Where:

   - **zipfilename**

     Specifies the name of the protocol plug-in file that you want to install. This is the name of the file that you uploaded by using the **`file get`** command.
   - **propertiesfilename**

     Optional.

     Specifies the protocol plug-in properties file name that you want to associate with the plug-in. You can develop a properties file in JSON format and the properties that it contains override any properties in the plug-in configuration file.
4. Stop and restart the IBM MessageSight plug-in server process by using the following commands:

   ```
   imaserver stop plugin
   ```

   ```
   imaserver start plugin
   ```

   The installation of the plug-in takes effect only when the IBM MessageSight plug-in server process is restarted. The plug-in can be updated at any time, but the update is not effective until the IBM MessageSight plug-in server process is restarted.
5. Verify that the plug-in is successfully deployed by entering the following command:

   ```
   imaserver list Plugin
   ```

   The plug-in is displayed in the resulting list. This value is specified by the **`Name`** property in the descriptor file.
6. Verify that the new protocol is successfully registered by entering the following command:

   ```
   imaserver list Protocol
   ```

   The protocol is displayed in the resulting list. This value is specified by the **`Protocol`** property in the descriptor file.
7. Update the configuration of your endpoint, connection policy, and messaging policy to authorize the new protocol. For more information about configuring endpoints, connection policies, and messaging policies, see Configuring message hubs in the IBM MessageSight documentation in IBM Knowledge Center.

Example

The following example shows the steps to import and create a protocol plug-in file that is called `jsonmsg`. This plug-in is contained in a zip archive called `jsonmsg.zip`.

```
admin@(none)>  file get scp://user@192.0.2.32:/tmp/jsonmsg.zip jsonmsg.zip
jsonmsg.zip  100%   10KB  10.0KB/s 00:00
Wrote 10272 bytes to local storage
admin@(none)>  file list
jsonmsg.zip 10272 bytes created May 28, 2014 8:14:39 PM
admin@(none)> imaserver create Plugin "File=jsonmsg.zip"
The requested configuration change has completed successfully.
The plug-in server must be restarted.
admin@(none)> imaserver stop plugin
The plug-in server is stopped.
admin@(none)> imaserver start plugin
The plug-in server is started.
admin@(none)> imaserver list Plugin
jsonmsg
```

```
admin@(none)> imaserver list Protocol
mqtt
jms
jsonmsg
```

Updating a protocol plug-in by using the command line

Update an installed protocol plug-in by using the **`imaserver update Plugin`** command on the command line.

Before you begin

To update a protocol plug-in, it must previously have been deployed in IBM MessageSight. The plug-in must be archived in a zip file. This zip file must contain the JAR file (or files) that implement the plug-in for the target protocol. The zip file must also contain a JSON descriptor file that describes the plug-in content. This descriptor file is required in the zip archive that is used to deploy a protocol plug-in. For more information about developing a protocol plug-in, see “Developing a protocol plug-in” on page 1.

Procedure

1. Update the plug-in by entering the following command:

   `imaserver update Plugin "File=`*`zipfilename`*`"`

   Where:

   -

     zipfilename
         Specifies the name of the protocol plug-in file that you want to update. This is the name of the file that you previously uploaded by using the **`file get`** command.
2. Stop and restart the IBM MessageSight plug-in server process by using the following commands:

   `imaserver stop plugin`

   `imaserver start plugin`

   The update of the plug-in takes effect only when the IBM MessageSight plug-in server process is restarted. The plug-in can be updated at any time, but the update is not effective until the IBM MessageSight plug-in server process is restarted.
3. Update the configuration of your endpoint, connection policy, and messaging policy to authorize the new protocol if necessary. For more information about configuring endpoints, connection policies, and messaging policies, see Configuring message hubs in the IBM MessageSight documentation in IBM Knowledge Center.

Example

The following example shows the steps to update a protocol plug-in file that is called `jsonmsg`.

```
admin@(none)> imaserver update Plugin "File=jsonmsg.zip"
The requested configuration change has completed successfully.
The plug-in server must be restarted.
admin@(none)> imaserver stop plugin
The plug-in server is stopped.
admin@(none)> imaserver start plugin
The plug-in server is started.
jms
jsonmsg
```

Updating a protocol plug-in properties file

Update an installed protocol plug-in properties file by using the **`imaserver update Plugin`** command on the command line.

Before you begin

To update a protocol plug-in properties file, the plug-in with which it is associated must previously have been deployed in IBM MessageSight. The plug-in must be archived in a zip file. This zip file must contain the JAR file (or files) that implement the plug-in for the target protocol. The zip file must also contain a JSON descriptor file that describes the plug-in content. This descriptor file is required in the zip archive that is used to deploy a protocol plug-in. For more information about developing a protocol plug-in, see “Developing a protocol plug-in” on page 1.

Procedure

1. Update the plug-in properties file by entering the following command:

   ```
   imaserver update Plugin "Name=pluginname" "PropertiesFile=propertiesfilename"
   ```

   Where:

   -
     pluginname
     Specifies the name of the protocol plug-in that is associated with the properties file. This can be found in the `plugin.json` configuration file if it exists.
   -
     propertiesfilename
     Specifies the protocol plug-in properties file name that you want to update.
2. Stop and restart the IBM MessageSight plug-in server process by using the following commands:

   ```
   imaserver stop plugin
   ```

   ```
   imaserver start plugin
   ```

   The update of the plug-in properties file takes effect only when the IBM MessageSight plug-in server process is restarted. The plug-in properties file can be updated at any time, but the update is not effective until the IBM MessageSight plug-in server process is restarted.

Example

The following example shows the steps to update a protocol plug-in properties file that is called `pluginprops.json`.

```
admin@(none)> imaserver update Plugin "File=pluginprops.json"
The requested configuration change has completed successfully.
The plug-in server must be restarted.
admin@(none)> imaserver stop plugin
The plug-in server is stopped.
admin@(none)> imaserver start plugin
The plug-in server is started.
admin@(none)>
```

Deleting a protocol plug-in by using the command line

Delete a protocol plug-in by using the **`imaserver delete Plugin`** command on the command line.

Procedure

1. Optional: Find the name of the plug-in you want to delete. To delete a protocol plug-in, you must specify the name of the plug-in. This value is specified by the **`Name`** property in the descriptor file. You can find this value by entering the following command:

   ```
   imaserver list Plugin
   ```
2. Delete the protocol plug-in by entering the following command:

   ```
   imaserver delete Plugin "Name=plugin_name"
   ```

   Where

plugin_name
Specifies the name of the plug-in that you want to delete.

The value of *plugin_name* is specified by the **`Name`** property in the descriptor file.

3. Restart the IBM MessageSight plug-in server by using the following commands:

```
imaserver stop plugin
```

```
imaserver start plugin
```

Example

The following example shows the steps to delete a protocol plug-in file that is called `jsonmsg`.

```
admin@(none)> imaserver delete Plugin "Name=jsonmsg"
The requested configuration change has completed successfully.
The plug-in server must be restarted.
admin@(none)> imaserver stop plugin
The plug-in server is stopped.
admin@(none)> imaserver start plugin
The plug-in server is started.
admin@(none)>
```

Starting and stopping the protocol plug-in server process

System administrators can start the protocol plug-in server process by using the **`imaserver start`** command on the command line. System administrators can stop the protocol plug-in server process by using the command on the command line.

About this task

You can start and stop the protocol plug-in server process by using the command line. The protocol plug-in server process is a Java process that loads all installed protocol plug-ins. The protocol plug-in server ensures that messages are processed by the appropriate protocol plug-in. After you create, update, or delete protocol plug-ins, you must restart the protocol plug-in server.

Procedure

- To start the protocol plug-in server process, enter the following command:

  ```
  imaserver start Plugin
  ```

- To stop the protocol plug-in server process, enter the following command:

  ```
  imaserver stop Plugin
  ```

Protocol Plug-in commands

The protocol plug-in commands are commands that you can use to apply, update, delete, and list plug-ins. You can also use these commands to list and show protocols. You can also use these commands to start and stop the protocol plug-in server process. You can also view and set the host IP address of the debug plug-in server, the maximum size of the JVM heap size of the debug plug-in server, and the port number of the JVM remote debugger.

imaserver create Plugin

Purpose

Create a protocol plug-in in IBM MessageSight. The protocol plug-in provides the ability to extend the protocols that are supported by the IBM MessageSight appliance, without compromising the security of the appliance environment.

Syntax

`imaserver create Plugin "File=`*zipfilename*`" [ "PropertiesFile=`*propertiesfilename*`" ]`

Parameters

zipfilename
: Specifies the name of the protocol plug-in file that you want to create. The *zipfilename* is the name of the file that you uploaded by using the **`file get`** command.

propertiesfilename
: Specifies the name of the protocol plug-in properties file that you want the protocol plug-in to be associated with.

Usage Notes

- You must upload the plug-in before you can create it in IBM MessageSight. You can upload the plug-in by using the **`file get`** command.
- The command must be capitalized as shown.
- The command must use the double quotation marks as shown.
- If a plug-in with the same name is already installed in IBM MessageSight, a plug-in will not be created and you will be prompted to use the **`imaserver update Plugin`** command.
- After you create the protocol plug-in in IBM MessageSight, you must stop and restart the plug-in server.

Related Commands

- file get
- "imaserver delete Plugin" on page 16
- "imaserver list Plugin" on page 18
- "imaserver start Plugin" on page 24
- "imaserver stop Plugin" on page 24

Example

Create a protocol plug-in that is already uploaded to IBM MessageSight, and stop and restart the plug-in server.

```
admin@(none)> imaserver create Plugin "File=jsonplugin.zip"
The requested configuration change has completed successfully.
The plug-in server must be restarted.
admin@(none)> imaserver stop plugin
The plug-in server is stopped.
admin@(none)> imaserver start plugin
The plug-in server is started.
```

---

imaserver delete Plugin

Purpose

Deletes a specified protocol plug-in from IBM MessageSight.

Syntax

`imaserver delete Plugin` "Name=*pluginname*"

Parameters

pluginname
Specifies the name of the protocol plug-in that you want to delete. The value of *pluginname* is specified by the **`Name`** property in the `plugin.json` file. You can find this value by using the **`imaserver list Plugin`** command.

Usage Notes

- You must restart the plug-in server. The deletion of the plug-in takes effect only when the plug-in server is restarted.

Related Commands

- “imaserver create Plugin” on page 15
- “imaserver list Plugin” on page 18
- “imaserver start Plugin” on page 24
- “imaserver stop Plugin” on page 24
- “imaserver update Plugin” on page 25

Example

Delete a file that is previously installed by the **`imaserver create Plugin`** command

```
admin@(none)> imaserver delete Plugin "Name=jsonmsg"
The requested configuration change has completed successfully.
The plug-in server must be restarted.
admin@(none)> imaserver stop plugin
The plug-in server is stopped.
admin@(none)> imaserver start plugin
The plug-in server is started.
admin@(none)>
```

---

imaserver get PluginDebugServer

Purpose

Shows the host IP address of the protocol plug-in debug server.

Syntax

`imaserver get PluginDebugServer`

Parameters

None.

Related Commands

- "imaserver set PluginDebugServer" on page 19
- "imaserver get PluginMaxHeapSize"
- "imaserver set PluginMaxHeapSize" on page 20
- "imaserver get PluginRemoteDebugPort" on page 18
- "imaserver set PluginRemoteDebugPort" on page 21

Example

Shows the host IP address of the protocol plug-in debug server.

```
admin@(none)> imaserver get PluginDebugServer
PluginDebugServer = 192.0.2.0
admin@(none)>
```

Note: If the protocol plug-in debug server is not configured, the following response is returned:

```
admin@(none)> imaserver get PluginDebugServer
The requested item "PluginDebugServer" is not found.
admin@(none)>
```

---

imaserver get PluginMaxHeapSize

Purpose

Shows the JVM maximum heap size, in megabytes, for the protocol plug-in debug server.

Syntax

```
imaserver get PluginMaxHeapSize
```

Parameters

None.

Related Commands

- "imaserver set PluginMaxHeapSize" on page 20
- "imaserver get PluginDebugServer" on page 16
- "imaserver set PluginDebugServer" on page 19
- "imaserver get PluginRemoteDebugPort" on page 18
- "imaserver set PluginRemoteDebugPort" on page 21

Example

Shows the value, in megabytes, of the JVM maximum heap size for the protocol plug-in debug server.

```
admin@(none)> imaserver get PluginMaxHeapSize
PluginMaxHeapSizeDebugServer = 512
admin@(none)>
```

Note: If the JVM maximum heap size for the protocol plug-in debug server is not configured, the default value of 512 MB is used, and the following response is returned:

```
admin@(none)> imaserver get PluginMaxHeapSize
The requested item "PluginMaxHeapSize" is not found.
admin@(none)>
```

imaserver get PluginRemoteDebugPort

Purpose

Shows the port number of the JVM remote debugger of the protocol plug-in debug server.

Syntax

`imaserver get PluginRemoteDebugPort`

Parameters

None.

Related Commands

- "imaserver set PluginRemoteDebugPort" on page 21
- "imaserver get PluginDebugServer" on page 16
- "imaserver set PluginDebugServer" on page 19
- "imaserver get PluginMaxHeapSize" on page 17
- "imaserver set PluginMaxHeapSize" on page 20

Example

Shows the port number of the JVM remote debugger of the protocol plug-in debug server.

```
admin@(none)> imaserver get PluginRemoteDebugPort
PluginRemoteDebugPort = 8000
admin@(none)>
```

Note: If the port number of the JVM remote debugger of the protocol plug-in debug server is not configured, the following response is returned:

```
admin@(none)> imaserver get PluginRemoteDebugPort
The requested item "PluginRemoteDebugPort" is not found.
admin@(none)>
```

imaserver list Plugin

Purpose

Lists all protocol plug-ins on IBM MessageSight.

Syntax

`imaserver list Plugin`

Parameters

None

Usage Notes

- The command must be capitalized as shown.

Related Commands

- "imaserver create Plugin" on page 15
- "imaserver update Plugin" on page 25
- "imaserver delete Plugin" on page 16

Example

Show the plug-ins that are currently on the appliance.

```
admin@(none)> imaserver list Plugin
jsonmsg
admin@(none)>
```

imaserver list Protocol

Purpose

Lists all protocols on IBM MessageSight.

Syntax

`imaserver list Protocol`

Parameters

None

Usage Notes

- The command must be capitalized as shown.
- Use the **`imaserver show Protocol`** command to view specific protocol information.

Related Commands

- "imaserver create Plugin" on page 15
- "imaserver update Plugin" on page 25
- "imaserver delete Plugin" on page 16
- "imaserver show Plugin" on page 22
- "imaserver show Protocol" on page 23

Example

List the protocols on IBM MessageSight.

```
admin@(none)> imaserver list Protocol
mqtt
jms
admin@(none)>
```

imaserver set PluginDebugServer

Purpose

Sets the host IP address of the debug plug-in server.

Syntax

`imaserver set "PluginDebugServer=`*IPAddress*"

Parameters

IPAddress
Specifies the host IP address of the debug plug-in server.

By default, this value is not set.

Usage Notes

- The command must be capitalized as shown.
- The command must use the double quotation marks as shown.
- Specify a valid IP address if the debug plug-in server is running in Eclipse.
- If you are debugging remotely, ensure that a value is not set for **`PluginDebugServer`** by specifying `imaserver set "PluginDebugServer="` and restart the IBM MessageSight server.
- After you change the host IP address of the debug plug-in server, the IBM MessageSight server must be restarted.

Related Commands

- "imaserver get PluginDebugServer" on page 16
- "imaserver get PluginMaxHeapSize" on page 17
- "imaserver set PluginMaxHeapSize"
- "imaserver get PluginRemoteDebugPort" on page 18
- "imaserver set PluginRemoteDebugPort" on page 21

Example

Set the IP address of the debug plug-in server that is running locally on Eclipse.

```
admin@(none)> imaserver set "PluginDebugServer=9.3.179.129"
The requested configuration change has completed successfully.
The IBM MessageSight Server must be restarted.
admin@(none)>
```

---

imaserver set PluginMaxHeapSize

Purpose

Sets the JVM maximum heap size of the debug plug-in server.

Syntax

`imaserver set "PluginMaxHeapSize=`*size*"

Parameters

size Specifies the JVM maximum heap size, in megabytes (MB), of the debug plug-in server.

The default maximum heap size is 512 MB.

Usage Notes

- The command must be capitalized as shown.
- The command must use the double quotation marks as shown.
- Specify `imaserver set "PluginMaxHeapSize="` to reset the value so that the value of the default JVM maximum heap size is used.
- After you change the JVM maximum heap size of the debug plug-in server, the IBM MessageSight server must be restarted.

Related Commands

- "imaserver get PluginMaxHeapSize" on page 17
- "imaserver get PluginDebugServer" on page 16
- "imaserver set PluginDebugServer" on page 19
- "imaserver get PluginRemoteDebugPort" on page 18
- "imaserver set PluginRemoteDebugPort"

Example

Set the JVM maximum size of the plug-in debug server.

```
admin@(none)> imaserver set "PluginMaxHeapSize=1024"
The requested configuration change has completed successfully.
The IBM MessageSight server must be restarted.
admin@(none)>
```

---

imaserver set PluginRemoteDebugPort

Purpose

Sets the port number of the JVM remote debugger of the plug-in server.

Syntax

`imaserver set "PluginRemoteDebugPort=`*portNumber*"

Parameters

portNumber

Specifies the port number of the JVM remote debugger of the plug-in server. The value must be the same as the port number that Eclipse is using for remote debugging.

By default, this value is not set.

Usage Notes

- The command must be capitalized as shown.
- The command must use the double quotation marks as shown.
- Specify a valid port number if the debug plug-in server is running on your IBM MessageSight virtual appliance.
- If the debug plug-in server is running on Eclipse, ensure that a value is not set for **`PluginRemoteDebugPort`** by specifying `imaserver set "PluginRemoteDebugPort="` or `imaserver set "PluginRemoteDebugPort=0"`
- After you change port number of the JVM remote debugger of the debug plug-in server, the IBM MessageSight server must be restarted.

Related Commands

- "imaserver get PluginRemoteDebugPort" on page 18
- "imaserver get PluginDebugServer" on page 16
- "imaserver set PluginDebugServer" on page 19
- "imaserver get PluginMaxHeapSize" on page 17
- "imaserver set PluginMaxHeapSize" on page 20

Example

Set the port number of the JVM remote debugger of the plug-in server running on your IBM MessageSight virtual appliance.

```
admin@(none)> imaserver set "PluginRemoteDebugPort=8000"
The requested configuration change has completed successfully.
The IBM MessageSight server must be restarted.
admin@(none)>
```

---

imaserver show Plugin

Purpose

Shows the configuration file contents of IBM MessageSight protocol plug-ins.

Syntax

`imaserver show Plugin "Name=` *pluginname*"

Parameters

pluginname

Specifies the name of the protocol plug-in of which you want to view the configuration file contents.

Usage Notes

- The command must be capitalized as shown.
- The command must use the double quotation marks as shown.
- Use the **`imaserver list Plugin`** command to view all protocol plug-ins on IBM MessageSight.

Related Commands

- "imaserver list Plugin" on page 18

Example

Show the contents of the `plugin.json` configuration file.

```
admin@(none)> imaserver show Plugin "Name=json_msg"
/*
* Licensed Materials - Property of IBM
* 5725-F96 IBM MessageSight
* (C) Copyright IBM Corp. 2014. All Rights Reserved.
*
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*
* This file defined the IBM MessageSight plug-in for the json_msg sample protocol.
*/
{
/*
* The name of the plug-in. This must be unique among all installed plug-ins. The name is limited
* to 64 characters and must be a valid Java name. It can start with any alphabetic character,
* currency symbol, or underscore, and continue with any such character or a digit.
*/
"Name": "json_msg",

/*
* The protocol family against which to authorize. Each plug-in represents a single protocol family,
* but the same protocol family may be used by several plug-ins. The plug-in can also use one of the
* system protocol families. All policy checking for protocol is done based on this protocol.
* The name must be limited to 64 characters and must contain only ASCII-7 characters not including
* control characters (0x00 to 0x1F and x07F), space, or the separator characters '()<>[]{},;:\/?=".
*/
"Protocol": "json_msg",

/*
* A set of jar files containing the Java classes needed for the plug-in as a JSON array of strings.
* This must not include the path, and these jar files should be in the root directory of the zip file
* used to define the plug-in. These jar files are used only by this plug-in using a separate directory
* and class loader for each plug-in. The file must exist in the zip file used for install.
*/
"Classpath": [ "jsonprotocol.jar" ],

/*
* The name of the initial class to load for a plug-in. This must be an instance of ImaPluginListener.
* This must be a valid Java package name and class name separated by dots.
*/
"Class": "com.ibm.ima.samples.plugin.jsonmsg.JMPlugin",

/*
* An array of WebSockets sub-protocols supported by this plug-in. These are checked in a case independent
* manor. These should be unique among the installed plug-in set. If this property is missing or the array
* is empty, then no WebSockets protocol will connect to this plug-in. The name must be limited to 64 characters
* and must contain only ASCII-7 characters not including control characters (0x00 to 0x1F and x07F),
* space, or the separator characters '()<>[]{},;:\/?=".
*/
"WebSocket": [ "json-msg" ],

/*
* A set of initial bytes which can be specified as an array of strings of length 1 byte or as integers
* of the value 0-255. A single entry with the value "ALL" indicates that any initial byte will be selected.
* If this is not specified or the array is empty, then TCP connections will not be accepted.
*/
"InitialByte": [ "{" ], /* The json_msg always starts with a JSON object */

/*
* Define the capabilities of the plug-in
*/
"UseQueue": false, /* This plug-in does not implement queues */
"UseTopic": true, /* This plug-in implemetns topics */

}

{
/*
* A set of properties which are sent to the plug-in as configuration.
* The names and types of the properties are not known to the IBM MessageSight server
*/
"Properties": {
"Debug": false
}
}admin@(none)>
```

imaserver show Protocol

Purpose

Shows the allowed actions for a specified protocol on IBM MessageSight.

Syntax

imaserver show Protocol "Name= *protocolname*"

Parameters

protocolname

Specifies the name of the protocol that you want to view the allowed actions for.

Usage Notes

- The command must be capitalized as shown.
- Use the **`imaserver list Protocol`** command to view all protocols on IBM MessageSight.
- You cannot use the **`imaserver show Protocol`** command on a newly installed plug-in until the server is stopped and restarted.

Related Commands

- "imaserver list Protocol" on page 19

Example

Show the available actions for the JMS protocol on IBM MessageSight.

```
admin@(none)> imaserver show Protocol "Name=jms"
Name = jms
ActionList = Publish,Subscribe,Receive,Control,Send,Browse
admin@(none)>
```

---

imaserver start Plugin

Purpose

Starts the protocol plug-in server process.

Syntax

`imaserver start Plugin`

Parameters

None.

Usage Notes

- The command must be capitalized as shown.

Related Commands

- "imaserver stop Plugin"

Example

Start the protocol plug-in server.

```
admin@(none)> imaserver start Plugin
The plug-in server is started.
admin@(none)>
```

---

imaserver stop Plugin

Purpose

Stops the protocol plug-in server process.

Syntax

`imaserver stop Plugin`

Parameters

None.

Usage Notes

- The command must be capitalized as shown.

Related Commands

- "imaserver start Plugin" on page 24

Example

Stop the protocol plug-in server.

```
admin@(none)> imaserver stop Plugin
The plug-in server is stopped.
admin@(none)>
```

---

imaserver update Plugin

Purpose

Update an installed protocol plug-in in IBM MessageSight, or its properties file, or both the protocol plug-in and its properties file.

Syntax

To update the installed protocol plug-in:

`imaserver update Plugin "File=` *zipfilename***`"`**

To update the protocol plug-in properties file:

`imaserver update Plugin "Name=` *pluginname***`" "PropertiesFile=`***propertiesfilename***`"`**

To update the protocol plug-in and the properties file:

`imaserver update Plugin "File=` *zipfilename***`" "PropertiesFile=`***propertiesfilename***`"`**

Parameters

zipfilename
Specifies the name of the protocol plug-in file that you want to update. The *zipfilename* is the name of the file that you uploaded by using the **`file get`** command.

pluginname
Specifies the name of the protocol plug-in. This can be found in the `plugin.json` configuration file if it exists.

propertiesfilename
Specifies the name of the protocol plug-in properties file.

Usage Notes

- The command must be capitalized as shown.
- The command must use the double quotation marks as shown.

Related Commands

- file get
- “imaserver create Plugin” on page 15
- “imaserver delete Plugin” on page 16
- “imaserver list Plugin” on page 18
- “imaserver start Plugin” on page 24
- “imaserver stop Plugin” on page 24

Example

Update a protocol plug-in and properties file that are already created and installed in IBM MessageSight.

```
admin@(none)> imaserver update Plugin "File=jsonplugin.zip" "PropertiesFile=pluginprops.json"
The requested configuration change has completed successfully.
The plug-in server must be restarted.
admin@(none)> imaserver stop plugin
The plug-in server is stopped.
admin@(none)> imaserver start plugin
The plug-in server is started.
admin@(none)>
```

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com®, are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at “Copyright and trademark information”www.ibm.com/legal/copytrade.shtml. Other product and service names might be trademarks of IBM or other companies.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.