Adding a Custom Listener

This topic explains how to add a custom listener that enables a virtual asset to receive and respond to messages over a transport that is not supported by default. Sections include:

About Custom Listeners

Each virtual asset's configuration panel contains a **Transports> Custom** tab, which can be customized to display a message listener implementation to meet your group's specific needs.

Once the custom message listener implementation has been plugged in to Virtualize, the virtualization framework can correlate and generate response messages. The actual delivery and reception of messages over the necessary protocol is handled by the message listener implementation.

Note that Parasoft Marketplace (accessible at marketplace.parasoft.com or your organization's CTP) provides examples for TCP/IP, FTP, and other fixed-length message types such as Equifax.

Interfaces to Implement for Custom Listeners

After setting up your environment as described in General Procedure of Adding an Extension, implement the following interfaces (described in the Extensibility API documentation):

- com.parasoft.api.responder.ICustomMessageListener
- com.parasoft.api.ICustomMessage<T> (or reuse/extend com.parasoft.api.DefaultCustomMessage<T>)

ICustomMessageListener Implementation

This is a required class. The `isReady()` method should return true if the configuration has been properly configured, and `false` if the required settings have not been set. The `startup()` method is invoked when the listener should start listening. It takes:

- A configuration (never null) that provides the transport implementation class with the values coming from the responder deployment configuration GUI.
- A handler, an implementation of ImessageHandler (never null) that can accept a request message and returns a response message.

ICustomMessage

This interface needs to be implemented so that an instance of it can be accepted by the `IMessageHandler.handleMessage()` implementation. Alternatively, if your message content format is best represented with a string, you can use the default implementation DefaultCustomMessage<T>. The parameterized type can be used for the property (or header) object types keyed with a string.

Defining parasoft-extension.xml for a Custom Listener

After you have implemented the necessary classes, define parasoft-extension.xml (introduced in General Procedure of Adding an Extension) as follows:

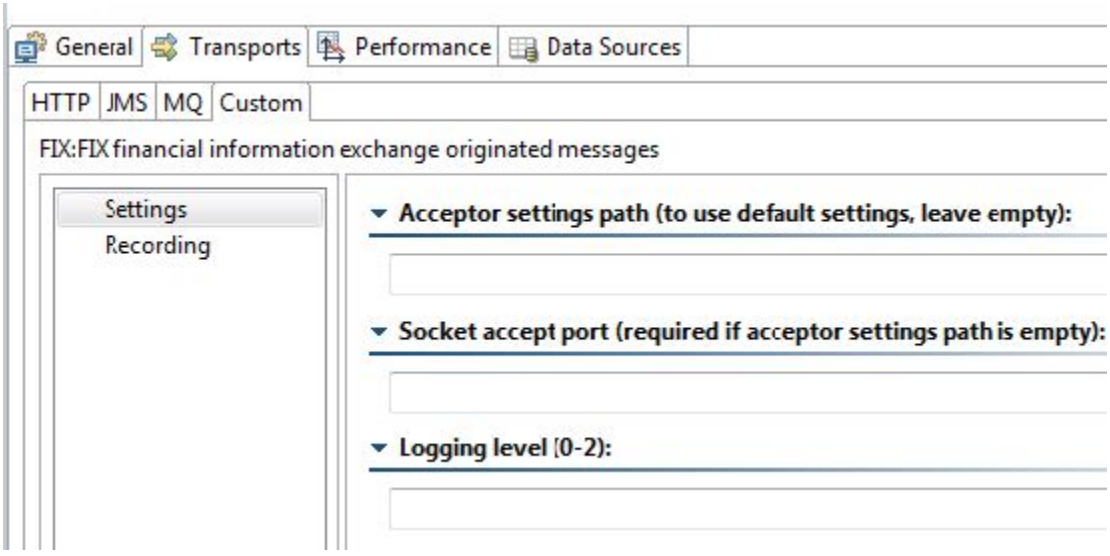
```
<?xml version="1.0" encoding="UTF-8"?>
<extension xmlns="urn:com/parasoft/extensibility-framework/extension"
       type="messageListener"
       name='The name of your message listener, appears in the transports menu'
       description='A more detailed description'>
  <class>com.mycompany.MyMessageListener</class> <!-- implements ICustomMessageListener-->
  <form xmlns="urn:com/parasoft/extensibility-framework/gui">
       <!-- This describes the fields you wish to appear in your transport GUI -->
       <section label="field group 1">
          <field id="key 1" label="field 1"/>
          <field id="key 2" label="field 2"/>
          ...
          <section label="field group 2">
          <field id="key 3" label="field 3" />
      </section>
      <section label="field group 2">
          <field label="field 1" />
          ...
      </section>
      ...
  </form>
</extension>
```

The field ids under the section elements are used to retrieve values from the ICustomMessageListenerConfiguration instance that is passed in to the various API methods, and that allows for the values provided by the user to be passed in to your implementation. They are also used to persist the user-provided values to the responder deployment descriptor file when it is saved. The field labels appear in the GUI that is constructed based on this XML. The section layout does not have a programmatic impact; it merely helps organize the various GUI fields into sections or categories for easy access and usability by the end user.

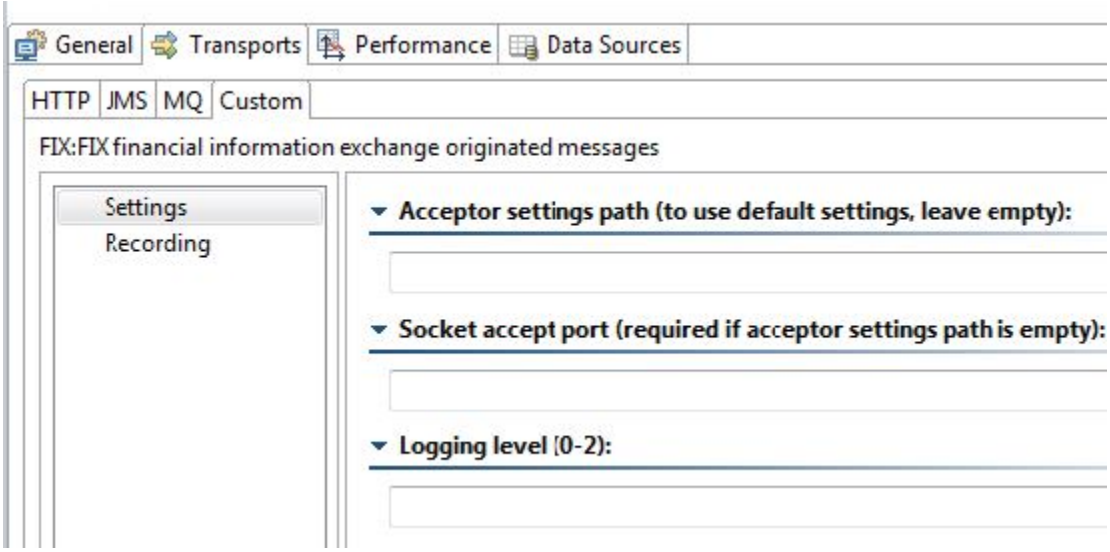
Verifying the New Listener

After building the project as described in General Procedure of Adding an Extension, restart Virtualize and verify that the custom listener name (as specified in `parasoft-extension.xml`) appears in the **Transports> Custom** tab of the Virtual Asset configuration panel (opened by double-clicking a virtual asset's Virtualize Server node).

If only one custom listener is available, this tab will be dedicated to configuration for that custom format:



If multiple custom listeners are available, you can select the one you want to use from the **Select Implementation** box:



Examples

A few custom message listener implementations are available on the Parasoft Marketplace (accessible at marketplace.parasoft.com or from your organization's CTP). See the documentation provided in those zip files for details on how to install and use these extensions.

Tips

- The values provided to the extension GUI are saved as a name-value String map. As a result, rearranging the fields in the form element in parasoft-extension.xml will not affect how the user values are saved; however, changing the ids will affect this. The ids are used to save/load the values so they need to be unique. If you change them, then previously-saved configurations will not load the previous values and will become empty. However, you can use a version updater to migrate old settings saved with old ids to a new set of ids.
- Only GUI fields with string values are supported in the custom form GUI. If your extension requires integers or other types, then you may convert the string content to the desired type in the extension implementation.
- If you want a GUI field to serve as a password field (with inputs masked and the specified password saved securely), give that field element a password attribute that is set to true. For example, the following sets the pwd field to password mode:

```
<form xmlns="urn:com/parasoft/extensibility-framework/gui">
     <section label="Main Settings">
        <field id="usr" label="Username"/>
        <field id="pwd" label="Password" password="true"/>
     </section>
</form>
```

- Tables or lists can be implemented as comma-separated values in the string fields.
- To provide visibility into events and errors related to this custom listener, you can call logging from ApplicationContext (in the Extensibility API). Events can be categorized as INFO, ERROR, WARN, and DEBUG—categories which can be used as filter criteria in the SOAtest Event Monitor. We recommend that you invoke the method reportEvent(String message, String type) directly on the context that is passed in because that will automatically populate the source of the event (i.e, which responder and PVA it is coming from).