Lesson 15 Sending Messages

This exercise requires that you use **TrainingApp**, **Guidewire Studio**, **ExternalApp**, and a supported web browser. Start **Guidewire Studio for TrainingApp**. Start the server as **Debug** ‘Server’.

Start **ExternalApp** using the **Start ExternalApp** shortcut.

The default URL for **TrainingApp** is: http://localhost:8880/ab/ContactManager.do. Log in to **TrainingApp** as Super User whose login/password is **su/gw**.

Exercise 1: Configure sending a message

Exercise

Succeed Insurance must determine if a given contact has been involved with a previous act of insurance fraud. To implement their fraud prevention system, they must send a message to an external system for every new contact or for an existing contact whose tax ID is updated. The payload must be in XML format and must contain the contact’s full name, tax ID, and a reference value generated by Guidewire to help identify the message. The external system must respond immediately to the fraud investigation request with a fraud report code.

External system information:

The WSDL URL to the external system is:

http://localhost:8890/ab/ws/externalapp/webservice/FraudReportAPI?WSDL

The authentication parameters are:

- Username: externalappuser
- Password: gw

Do not code the authentication parameters in the implementation code – use plugin parameters that are passed to the code.

API method checkForFraudReport requires the transformed payload as its argument.

The valid acknowledgment report codes are as follows:

- **1** – Request processed; no fraud report found
  - Acknowledge the message.
- **2** – Request processed; fraud report found!
  - Acknowledge the message.
- **4** – Request could not be processed (Payload Format Error)
  - Acknowledge the message with error using error category **Payload Format**.
- **5** – Request could not be processed (Database Unavailable)
  - Acknowledge the message with error using error category **Database Contention**.

- **Default** – Request could not be processed (Acknowledgment Code Invalid). If the error code returned from the external system is not valid, then acknowledge the message with error using a new error category called **Acknowledgement Code Invalid**.

For training purposes, output an acknowledgment message to console using the print statement.

15.1.1 Requirements

Spec 1 Configure the system to send the payload to the external system.

Spec 2 API method checkForFraudReport requires the transformed payload as its argument.

Spec 3 Do not code the authentication parameters in the implementation code – use plugin parameters that are passed to the code.

15.1.2 Tasks

1. **Create a web service collection.**

Tip

Review the **SOAP Web Services** lesson for instructions on how to create a web service collection.

2. **Create message transport plugin class that implements InitializablePlugin interface.**

Tip

Review the **Plugins** lesson for instructions on how to implement the **InitializablePlugin** interface.

3. **Create and configure transport plugin registry.**
4. **Configure the destination.**
5. **Deploy code changes.**
6. **Perform verification steps.**

15.1.3 Verification steps

1. **Launch TrainingApp.**
2. **Verify Message and MessageHistory table screens are clear.**
3. **Clear pending messages in Message Table screen.**

   b) Navigate to **Administration ➔ Training: Messaging ➔ Message Table**.

   c) Select the new message with **Pending acknowledged** status.

   d) Click the **Skip Selected Message(s)** button.

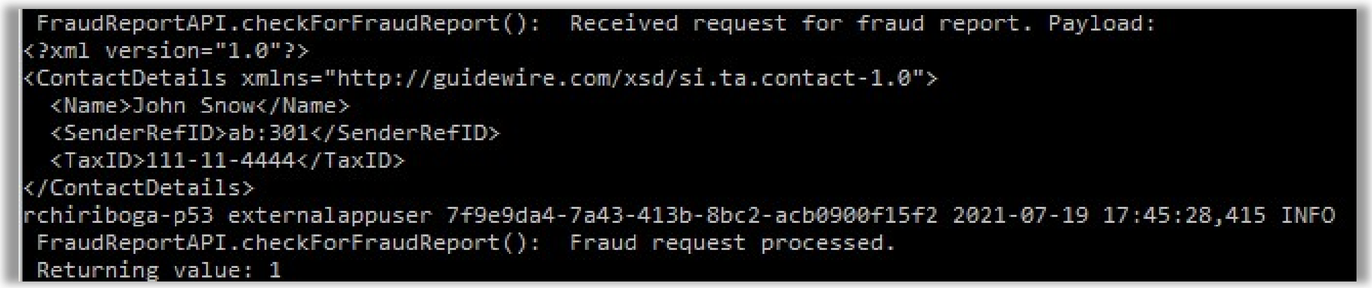
e) Click **OK** to the popup window.

4. **Clear messages in MessageHistory Table screen.**

   f) Navigate to **Administration ➔ Training: Messaging ➔ MessageHistory Table**.
   g) Select all messages.
   h) Click **Delete Selected Message Histories** button.
   i) Click **OK** to the popup window.

5. **Edit the Tax ID of the contact named John Snow.**

   a) Search for **John Snow**.
   b) Navigate to the **Details** screen.
   c) Edit the **Tax ID** field. Enter **111-11-4444**
   d) In **ExternalApp** console verify the message received. The payload should include the **SenderRefID**. The returning value will vary based on the **Tax ID** number entered.



```
 FraudReportAPI.checkForFraudReport():  Received request for fraud report. Payload:
<?xml version="1.0"?>
<ContactDetails xmlns="http://guidewire.com/xsd/si.ta.contact-1.0">
  <Name>John Snow</Name>
  <SenderRefID>ab:301</SenderRefID>
  <TaxID>111-11-4444</TaxID>
</ContactDetails>
rchiriboga-p53 externalappuser 7f9e9da4-7a43-413b-8bc2-acb0900f15f2 2021-07-19 17:45:28,415 INFO
 FraudReportAPI.checkForFraudReport():  Fraud request processed.
 Returning value: 1
```

6. **Clear pending messages in Message Table screen.**

   a) Navigate to **Administration ➔ Training: Messaging ➔ Message Table**.
   b) Select all messages with **Pending acknowledged** status.
   c) Click the **Skip Selected Message(s)** button.
   d) Click **OK** to the popup window.

7. **Clear messages in MessageHistory Table screen.**

   a) Navigate to **Administration ➔ Training: Messaging ➔ MessageHistory Table**.
   b) Select all messages.
   c) Click **Delete Selected Message Histories** button.
   d) Click **OK** to the popup window.

Solution 1: Configure sending a message

1. **Create a web service collection.**

   a) Right-click on **si.ta.messaging.fraudcheck** package and select **New ➔ Web Service Collection**.

   b) Enter **fraudcheckwsc** as the new WS-I web service collection name.

   c) Click the **PLUS** symbol to add a new resource.

   d) Enter the resource URL.

   http://localhost:8890/ab/ws/externalapp/webservice/FraudReportAPI?WSDL

   e) Click the **Fetch** icon to retrieve external resources. Make sure ExternalApp is running.

2. **Create message transport plugin class that implements InitializablePlugin interface.**

   a) Right-click on **si.ta.messaging.fraudcheck** package and select **New ➔ Gosu Class**.

   b) Enter **FraudCheckTransport** as the new class name.

   c) Implement the **MessageTransport** and **InitializablePlugin** interfaces.

   d) Configure **set Parameters** property.

   e) Configure the **send** method.

```
package si.ta.messaging.fraudcheck

uses gw.plugin.InitializablePlugin
uses gw.plugin.messaging.MessageTransport
uses si.ta.messaging.fraudcheck.fraudcheckwsc.fraudreportapi.FraudReportAPI

class FraudCheckTransport implements MessageTransport, InitializablePlugin{
  // Plugin parameters
  private static final var USERNAME = "Username"
  private static final var PASSWORD = "Password"
  // API parameters
  private static var _username : String
  private static var _password : String

  var _fraudReportApi: FraudReportAPI

  override function send(message : Message, transformedPayload : String) {
    var ackCode = _fraudReportApi.checkForFraudReport(transformedPayload)
  }

  override function shutdown() {  }

  override function suspend() {
    _fraudReportApi = null
  }

  override function resume() {
    _fraudReportApi = new FraudReportAPI()
    _fraudReportApi.Config.Http.Authentication.Basic.Username = _username
    _fraudReportApi.Config.Http.Authentication.Basic.Password = _password
  }

  override property set DestinationID(destinationID : int) {  }

  override property set Parameters(map : Map<Object, Object>) {
    _username = map.get(USERNAME) as String
    _password = map.get(PASSWORD) as String
  }

}
```

3. **Create and configure transport plugin registry.**

   a) Navigate to **configuration.config.Plugins.registry** package.

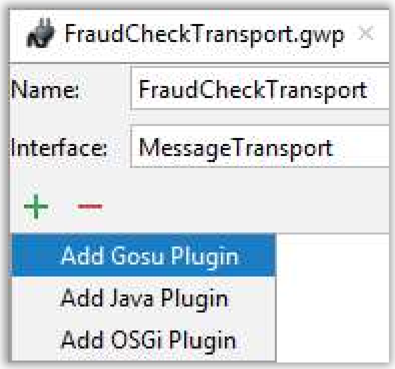
   b) Right-click on **registry** and select **New ➔ Plugin**.

   c) Enter **FraudCheckTransport** for the plugin name.

   d) Enter **MessageTransport** for the Interface.

   e) Configure the plugin.

      ▪ Add a Gosu plugin.

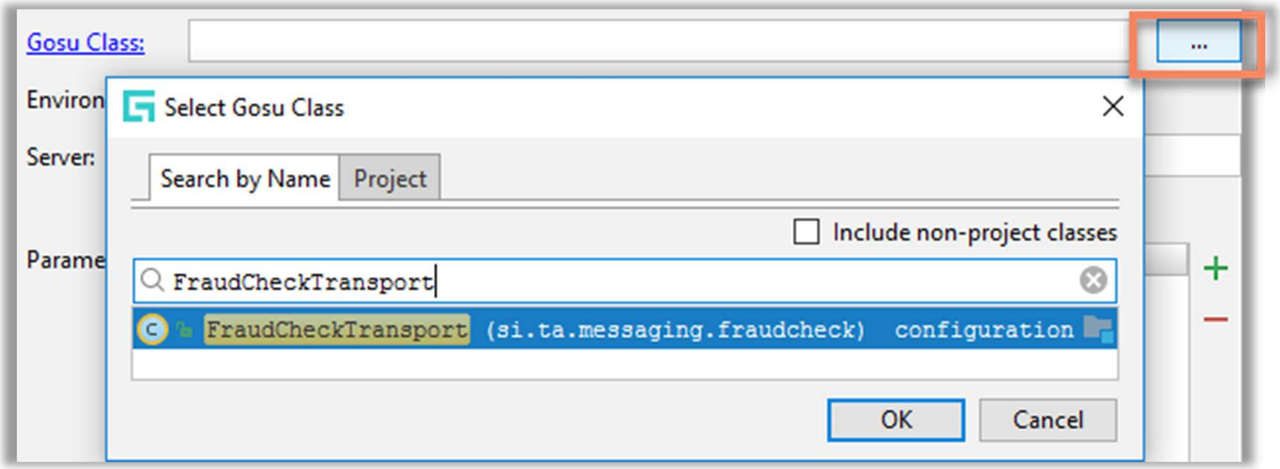


FraudCheckTransport.gwp

Name: FraudCheckTransport

Interface: MessageTransport

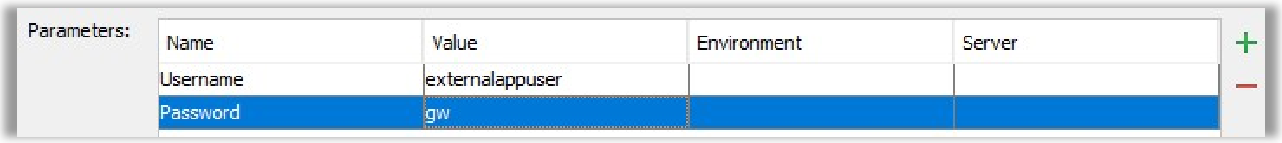
      ▪ Configure the Gosu Class field.



FraudCheckTransport

FraudCheckTransport (si.ta.messaging.fraudcheck) configuration

      ▪ Add plugin parameters.



Parameters:

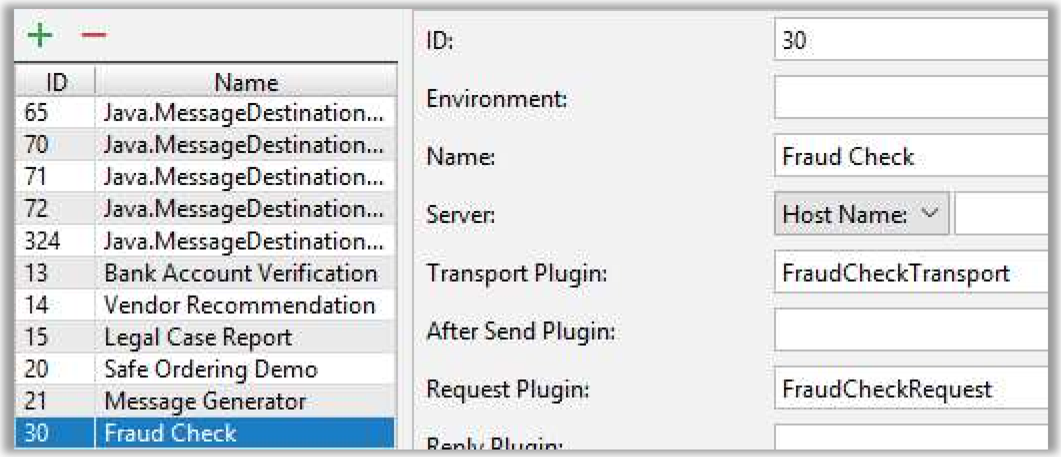
Name	Value	Environment	Server
Username	externalappuser		
Password	gw		

4. **Configure the destination.**

   a) Open **messaging-config.xml** file.

   b) Select **FraudCheck** destination.

   c) Add **FraudCheckTransport** in **Transport Plugin** field.



ID	Name
65	Java.MessageDestination...
70	Java.MessageDestination...
71	Java.MessageDestination...
72	Java.MessageDestination...
324	Java.MessageDestination...
13	Bank Account Verification
14	Vendor Recommendation
15	Legal Case Report
20	Safe Ordering Demo
21	Message Generator
30	Fraud Check

ID: 30

Environment:

Name: Fraud Check

Server: Host Name:

Transport Plugin: FraudCheckTransport

After Send Plugin:

Request Plugin: FraudCheckRequest

Reply Plugin:

5. **Deploy code changes.**

   a) From the **Studio** menu, **Restart** the server.

6. **Perform verification steps.**