Lesson 14 Message Payload Transformation

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

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 message payload transformation

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.

14.1.1 Requirements

Spec 1 Transform the payload to include the SenderRefID reference value.

Tip

Use BankAccountVerificationRequest class as an example for transforming an XML payload.

14.1.2 Tasks

1. **Create message request plugin class.**
2. **Create and configure request plugin registry.**
3. **Configure the destination.**
4. **Deploy code changes.**
5. **Perform verification steps.**

14.1.3 Verification steps

1. **Launch TrainingApp.**

Tip

To further test the same contact, you need to acknowledge any outstanding messages for the contact. This is because the application enforces a concept known as Safe Ordering – which means that a new message will not be sent for a specific contact until all previous outstanding messages for the specific contact have been acknowledge. This concept will be covered in detail in the Sending Messages lesson.

2. **Acknowledge message(s) in Message Table screen.**
    a) Navigate to **Administration ➔ Training: Messaging**.
    b) Select message(s) with **Pending acknowledged** status.
    c) Click the **Skip Selected Message(s)** button.
    d) Click **OK** to the popup window.
3. **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 **111-11-3333**.

d) Click **Update** button.

e) In Studio, verify the message and was created and transformed in the console output. The transformed payload should include the **SenderRefID**.

```
Executing DefaultPrintToConsoleTransport send() method
Payload for message 201: <?xml version="1.0"?>
<ContactDetails xmlns="http://guidewire.com/xsd/si.ta.contact-1.0">
  <Name>John Snow</Name>
  <SenderRefID>ab:201</SenderRefID>
  <TaxID>111-11-3333</TaxID>
</ContactDetails>
```

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

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

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

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

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

5. **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 message payload transformation

1. **Create message request plugin class.**

   a) Right-click on **gsrc** package and select **New ➔ Package**.

   b) Enter **si.ta.messaging.fraudcheck** as the new package name.

   c) Right-click on **fraudcheck** package and select **New ➔ Gosu Class**.

   d) Enter **FraudCheckRequest** as the new class name.

   e) Implement the **MessageRequest** interface and configure the **beforeSend** method.

```
package si.ta.messaging.fraudcheck

uses gw.plugin.messaging.MessageRequest
```

```
class FraudCheckRequest implements MessageRequest
{

  override function beforeSend(message : Message) : String {
    // Set the SenderRefID
    if (message.SenderRefID == null) {
      message.SenderRefID = message.PublicID
    }

    // Transform the payload
    var placeholder = message.MessageRoot.SenderRefIDPlaceholder_Ext
    var transformedPayload = message.Payload.replace(placeholder, message.SenderRefID)
    return transformedPayload
  }

  override function afterSend(message : Message) {  }
  override function shutdown() {  }
  override function suspend() {  }
  override function resume() {  }
  override property set DestinationID(destinationID : int) {  }

}
```

2. Create and configure request plugin registry.

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

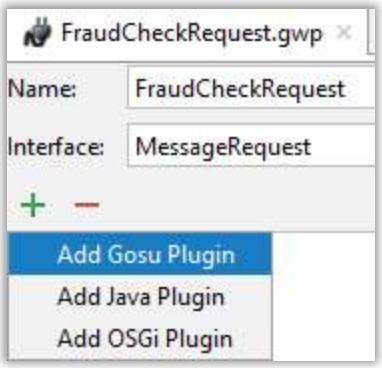
b) Right-click on **registry** and select **New ➔ Plugin**.

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

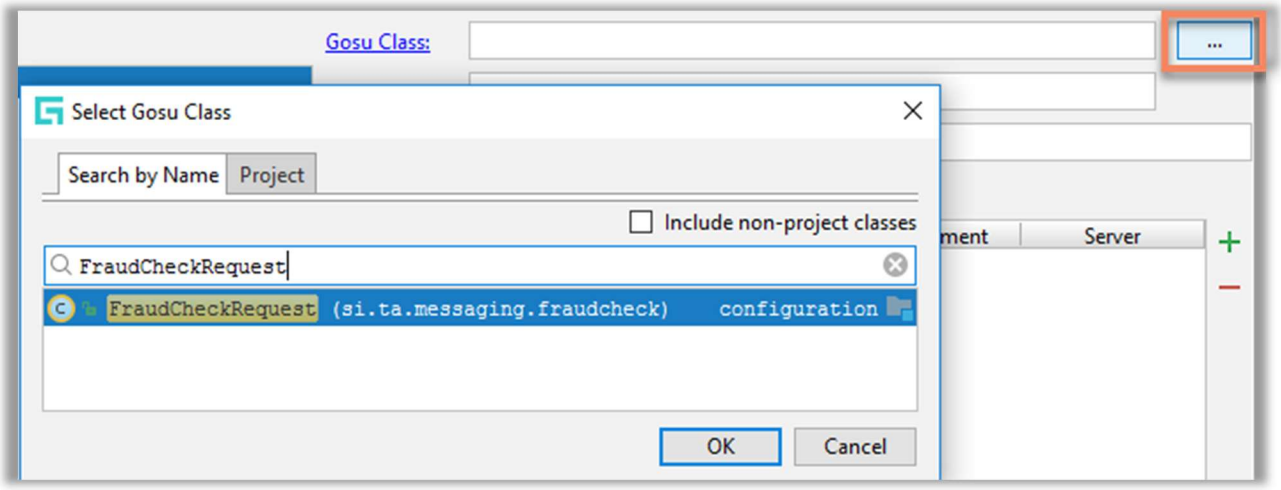
d) Enter **MessageRequest** for the Interface.

e) Configure the plugin.

- Add a Gosu plugin.



- Configure the Gosu Class field.

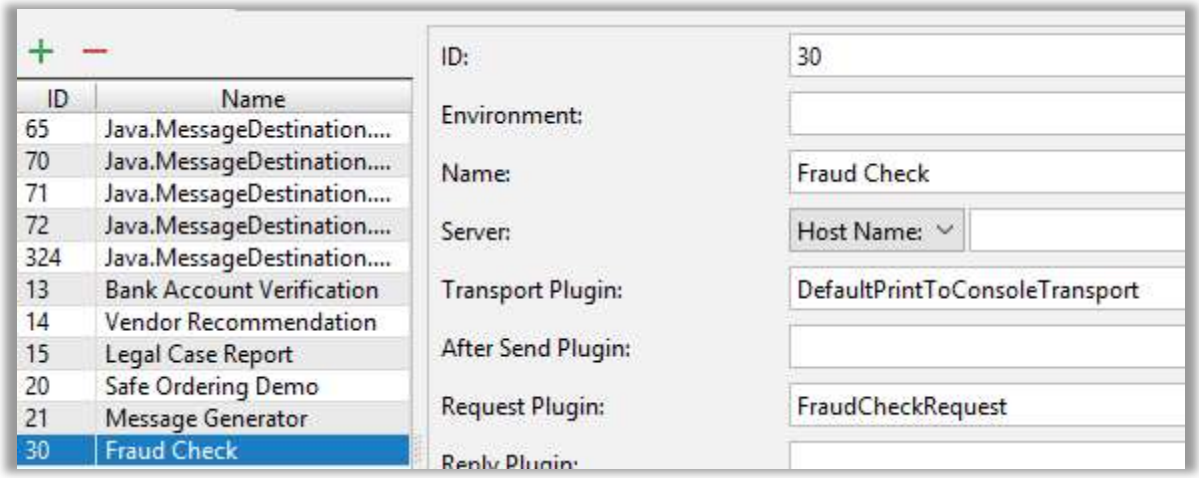


3. **Configure the destination.**

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

   b) Select **FraudCheck** destination.

   c) Add **FraudCheckRequest** in Request Plugin field.



4. **Deploy code changes.**

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

5. **Perform verification steps.**