

# 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.

