Lab: Messaging Sending

In this lab, you will build the next stage of the messaging integration point: sending the message. You will implement the plugins to send messages for the Fraud Check integration point.

Requirements

This lab requires that you use TrainingApp 8.0, ExternalApp 8.0, Guidewire Studio 8.0, and Chrome as your web browser. ExternalApp 8.0 serves as an external system that processes requests for fraud reports.

To view, edit, and delete various contacts, log in to TrainingApp as Super User. The default URL for TrainingApp is <http://localhost:8880/ab/ContactManager.do>. The login/password for Super User is su/gw.

To run ExternalApp, navigate to c:\Guidewire\ExternalApp and double-click the Start ExternalApp shortcut. The start-up process is complete when the message "\*\*\*\* ContactManager ready \*\*\*\*" appears in the console.

Configuration

1. Start ExternalApp 8.0
2. Navigate to c:\Guidewire\ExternalApp\.
3. Double-click the Start ExternalApp shortcut.
4. Create a web service collection for FraudReportAPI in TrainingApp
5. In Guidewire Studio for TrainingApp, create a web service collection for the ExternalApp FraudReportAPI web service.
6. Add a resource using the following ExternalApp WSDL:  
   <http://localhost:9990/ab/ws/external/FraudReportAPI?WSDL>
7. Confirm the fetched resources.
8. Message Request plugin

In this exercise, you will create a message request plugin that performs the late binding and transformation of the message payload. The message request plugin will set the SenderRefID for message payloads related to fraud checks. For synchronous messaging, setting the SenderRefID is normally not necessary. Using the SenderRefIDs will simplify the testing of message acknowledgements in the next lab.

Configuration

1. Create the message request plugin class
2. In Guidewire Studio, create the message request class.
3. Use the recommended naming convention for the class.
4. Implement the necessary interface and override the required function for late binding.
5. In your overridden function, set the SenderRefID to "fraud-<X>" where <X> is the message's ID.
6. Create the plugin registry
7. Implement the required interface.
8. Specify the message request plugin.
9. Modify the Fraud Check message destination
10. Specify your Fraud Check message request plugin for the Request Plugin field.
11. Deploy your changes
12. Message Transport plugin

In this exercise, you will create and deploy message transport plugin that sends the payload to ExternalApp's FraudReportAPI. You message transport plugin class must call the checkForFraudReport(messageID, payload) web service function that returns an integer. The following table describes the meaning of the return values.

|  |  |
| --- | --- |
| Integer | Meaning |
| 1 | Request processed, no fraud report found |
| 2 | Request processed, fraud report found! |
| 3 | Request is an improperly formatted string |
| 4 | Request is improperly formatted XML |
| 5 | Request could not be processed (database unavailable) |
| 6 | Request could not be processed (a permanent error occurred) |
| 9 | An unknown error occurred |

Configuration

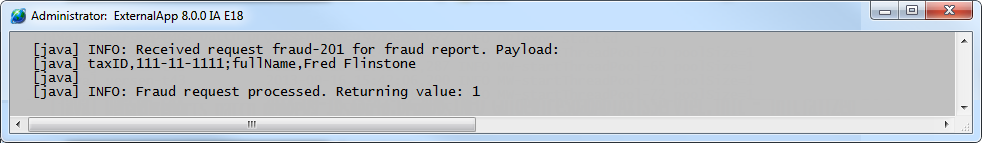
1. Create the message transport plugin class
2. In Guidewire Studio, create the message transport class.
3. Use the recommended naming convention for the class.
4. Implement the necessary interface and override the required function.
5. Send the payload using the following web service method:   
   checkForFraudReport(messageID, payload).
6. Use the authentication credentials of ExternalAppUser for the username and gw as the password.
7. Do NOT write any code to process the returned integer values. (You will acknowledge the return values in the next lab).
8. Create the plugin registry
9. Implement the required interface.
10. Specify the message transport plugin.
11. Modify the Fraud Check message destination
12. Specify your Fraud Check message transport plugin for the Transport Plugin field.
13. Deploy your changes

Verification

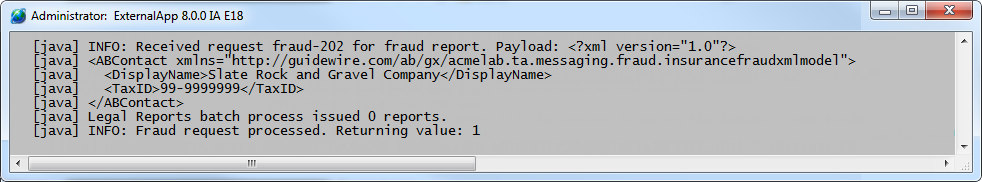
The ExternalApp FraudReportAPI web service prints to console every message payload it receives. It also prints to the console the return code it generates.

To verify your work, you will create several contacts in TrainingApp. For each contact, verify the message payload and the return value in the ExternalApp console.

1. In TrainingApp, create a new ABPerson
2. Log in to TrainingApp as Super User (su/gw).
3. Create a new ABPerson.
4. Verify the message in the ExternalApp command window
5. Verify that the message payload appears.
6. Verify the format of the payload.
7. Verify the SenderRefID.
8. Verify that the returning value is not equal to 3.

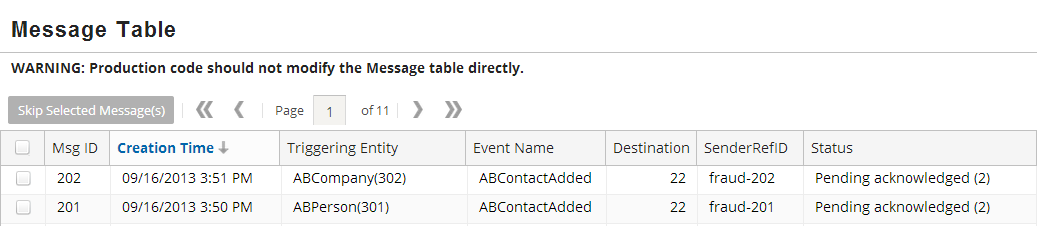


1. In TrainingApp, create a new ABCompany
2. Create a new ABCompany.
3. Verify the message in the ExternalApp command window
4. Verify the format of the payload.
5. Verify the SenderRefID.
6. Verify that the returning value is not equal to 4.



1. In TrainingApp, check the Message Table Screen
2. Navigate to the Message Table: Administration tab -> Training: Messaging -> Message Table.
3. Verify that your messages show a status of Pending acknowledged.

[Pic here]



|  |  |
| --- | --- |
|  | Stop and ask your instructor to review your completed lab. |