Lesson 13 Creating Messages

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 Event Fired rules

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.

13.1.1 Requirements

Spec 1 Trigger a message for new contacts of type ABContact.

Spec 2 Trigger a message for existing contacts that have modified their tax ID.

Spec 3 The XML message payload must contain the following two fields:

- Name
- SenderRefID
- Tax ID

Tip

1. Use the **Integration View** and **fraud_check** filter created in the Integration View exercise.
2. Use **Bank Account Verification** Event Fired rules as an example for structure best practices.

13.1.1 Tasks

1. **Create the Event Fired rules for Fraud Check.**
2. **Deploy code changes.**
3. **Perform verification steps.**

13.1.2 Verification steps

1. **Launch TrainingApp.**
2. **Create a new contact named John Snow and verify message is created.**

a) From the **Actions** menu, select **New Person ➔ Person**.
b) Complete required fields
c) Edit the **Tax ID** field. Enter **111-11-1111**.
d) Click **Update** button.
e) In **Studio**, verify the message was created by checking the console output.

```
Executing DefaultPrintToConsoleTransport send() method
Payload for message 101: <?xml version="1.0"?>
<ContactDetails xmlns="http://guidewire.com/xsd/si.ta.contact-1.0">
  <Name>John Snow</Name>
  <SenderRefID>@@SenderRefID@@</SenderRefID>
  <TaxID>111-11-1111</TaxID>
</ContactDetails>
```

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.

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

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

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

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

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

4. **Edit the tax ID of the new contact.**

   a) Search for **John Snow** contact.

   b) Navigate to the **Details** screen.

   c) Edit **Tax ID** = **111-11-2222**.

   d) Click the **Update** button.

   e) In **Studio**, verify the message was created in the console output.

```
Executing DefaultPrintToConsoleTransport send() method
Payload for message 102: <?xml version="1.0"?>
<ContactDetails xmlns="http://guidewire.com/xsd/si.ta.contact-1.0">
  <Name>John Snow</Name>
  <SenderRefID>@@SenderRefID@@</SenderRefID>
  <TaxID>111-11-2222</TaxID>
</ContactDetails>
```

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

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

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

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

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

Solution 1: Configure Event Fired rules

1. **Create the Event Fired rules for Fraud Check.**

   a) Navigate to **Event Fired** rule set.

   b) Create the following rules based on best practices structure.

   - Right-click on **Event Fired** rule set and select **New Rule**.
   - Enter **ABEF2000 – Fraud Check** for the rule name.
   - Move the new rule under **ABEF1000 – Vendor Recommendat**ion
     - Rule condition should check for the destination.


```
1    USES:
2
3    CONDITION (messageContext  :  entity.MessageContext):
7    return messageContext.DestID == 30
8    ACTION (messageContext  :  entity.MessageContext, actions : gw.rules.Action):
13   // execute child rules
14   END
```

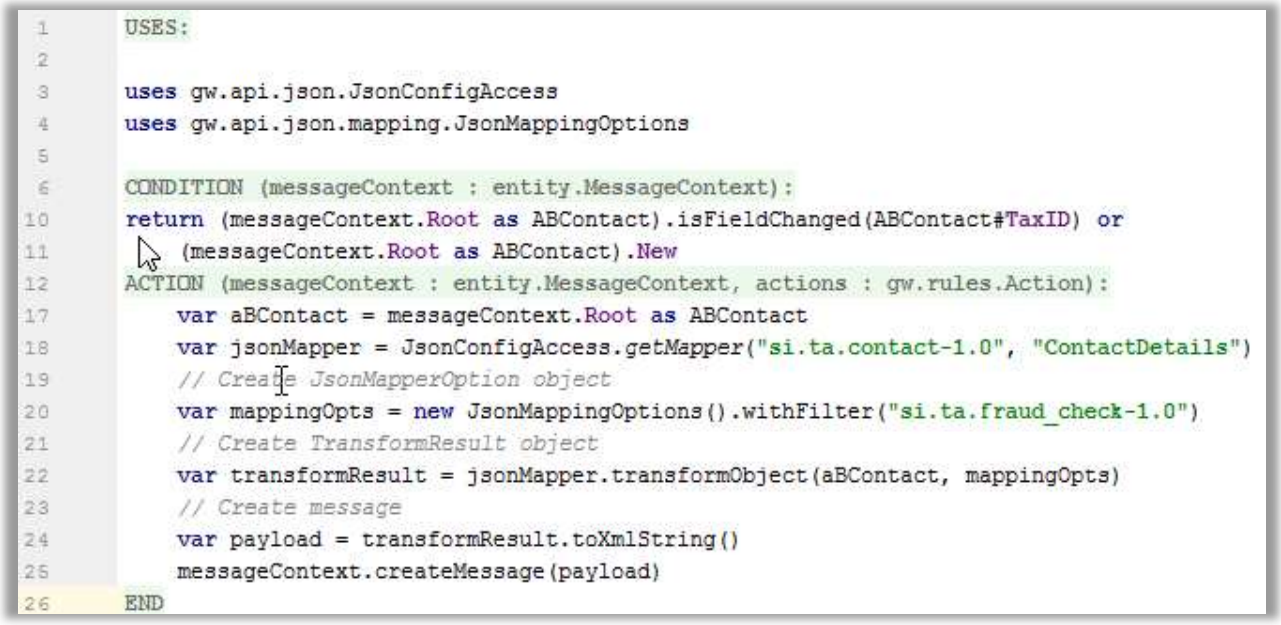
   - Create **ABEF2100 – ABContact** child rule.
     - Rule condition should check for entity type.

```
1   USES:
2
3   CONDITION (messageContext  :  entity.MessageContext):
7   return messageContext.Root typeis ABContact
8   ACTION (messageContext  :  entity.MessageContext, actions : gw.rules.Action):
13  // execute child rules
14  END
```

- Create **ABEF2110 – Added or Changed** child rule.
  - Rule condition should check for event type.


```
1   USES:
2
3   CONDITION (messageContext    :    entity.MessageContext):
7   return messageContext.EventName == "ABContactAdded" or
8          messageContext.EventName == "ABContactChanged"
9   ACTION (messageContext    :    entity.MessageContext, actions : gw.rules.Action):
14  // execute child rules
15  END
```

- Create **ABEF2111 – Field Changed** child rule.
  - Rule condition should check if tax ID field has changed or if new contact.
  - Rule condition should create a message whose payload is in XML format.



```
USES:

uses gw.api.json.JsonConfigAccess
uses gw.api.json.mapping.JsonMappingOptions

CONDITION (messageContext : entity.MessageContext):
return (messageContext.Root as ABContact).isFieldChanged(ABContact#TaxID) or
    (messageContext.Root as ABContact).New
ACTION (messageContext : entity.MessageContext, actions : gw.rules.Action):
    var aBContact = messageContext.Root as ABContact
    var jsonMapper = JsonConfigAccess.getMapper("si.ta.contact-1.0", "ContactDetails")
    // Create JsonMapperOption object
    var mappingOpts = new JsonMappingOptions().withFilter("si.ta.fraud_check-1.0")
    // Create TransformResult object
    var transformResult = jsonMapper.transformObject(aBContact, mappingOpts)
    // Create message
    var payload = transformResult.toXmlString()
    messageContext.createMessage(payload)
END
```

```
var aBContact = messageContext.Root as ABContact
    var jsonMapper = JsonConfigAccess.getMapper("si.ta.contact-1.0", "ContactDetails")
    // Create JsonMapperOption object
    var mappingOpts = new JsonMappingOptions().withFilter("si.ta.fraud_check-1.0")
    // Create TransformResult object
    var transformResult = jsonMapper.transformObject(aBContact, mappingOpts)
    // Create message
    var payload = transformResult.toXmlString()
    messageContext.createMessage(payload)
```

2. **Deploy code changes.**

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

3. **Perform verification steps.**