Lab: Generating Message Payloads

In this lab, you will build the second stage of the messaging integration point: generating the message payload. A message payload contains dynamically generated data. Guidewire offers two easy ways to define how the dynamic content of the message payload using either a Gosu template or a XML Model. In later message labs, you will continue to build and configure the other messaging stages.

For existing ABContacts with a modified Tax ID and for newly created contacts of the type ABContact, TrainingApp must determine if the given contact has been involved with a previous act of insurance fraud. The external system that checks for fraud requires that there are specific format for message payloads for various types of entities.

Requirements

This lab requires that you use TrainingApp 8.0, Guidewire Studio 8.0, and Chrome as your web browser.

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.

1. Generating the message payload

In this exercise, you will define two different message payload formats for sending messages to the external system for contacts of the type ABPerson.

Specification: ABPerson payload

The external system requires that the following format for a message payload for a contact of the type ABPerson:

* The names in each name/value pair must be taxID and fullName
* A comma must separate each name/value pair
* A semi-colon must delimit each name-value pair

Format example:

taxID,999-99-9999;fullName,William Andy

Use the following fields for the field values:

* anABContact.TaxID
* anABContact.FullName

Specification: ABCompany payload

The external system requires a XML format for the message payload for contacts of type ABCompany:

* Parent XML element for ABContact
* Sub elements for TaxID and DisplayName
* Sub elements can contain respective values for TaxID and DisplayName.

Format example:

<?xml version="1.0"?>

<ABContact>

<TaxID>---value---</TaxID>

<DisplayName>---value---</DisplayName>

</ABContact>

Use the following fields for the field values:

* anABCompany.TaxID
* anABCompany.DisplayName

Tasks

Implement the specifications for both ABPerson and ABCompany message payloads.

1. Create a the fraud package
2. In Guidewire Studio, create the fraud package with the fully qualified name of: acmelab.ta.messaging.fraud
3. Create a the Gosu template
4. According to the specification, create the necessary Gosu Template in the package.
5. Create the Guidewire XML model files
6. According to the specification, create the necessary GX Model in the package.
7. Modify the Event Fired rules
8. When the contact is an ABPerson, use the Gosu template for the payload.
9. When the contact is an ABCompany, use the GX Model for the payload.
10. Check the subtype of an ABContact by using the Subtype property or the by using the typeis operator.
11. Print to the console the payload output.

Verification

Recall that ABContact is a safe ordered entity. To verify your implementation task, you must use a different contact for each verification test that you execute. For this verification task, use your preferred method of running TrainingApp and view the console output.

1. Verify the payload for contacts of the type ABPerson
2. In TrainingApp, create a new ABPerson contact.
3. Verify the console output of the message.
4. Edit the Tax ID of another contact of the type ABPerson.
5. Verify the console output of the message.
6. Verify the payload for contacts of the type ABCompany
7. In TrainingApp, create a new ABCompany contact.
8. Verify the console output of the message
9. Edit the Tax ID of another contact of the type ABCompany.
10. Verify the console output of the message.

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