Lesson 12 Triggering 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 an event aware entity and destination

Important!

The comprehensive scenario below is applicable to all the messaging exercises. Therefore, the specifications applicable to each messaging exercise will be listed in the Requirements section.

End of important information.

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.

12.1.1 Requirements

Spec 1 Verify ABContact entity is configured to trigger messages based on default events.

Spec 2 Create a new messaging destination called Fraud Check.

- ID = 30
- Name = Fraud Check
- Transport Plugin = DefaultPrintToConsoleTransport
- Subscribe to events that trigger for new or modified contacts.

Tip

Do not change default connectivity and error parameters.

12.1.2 Tasks

1. **Verify ABContact entity delegates to EventAware delegate.**
2. **Create a new messaging destination called Fraud Check.**
3. **Deploy code changes.**
4. **Perform verification steps.**

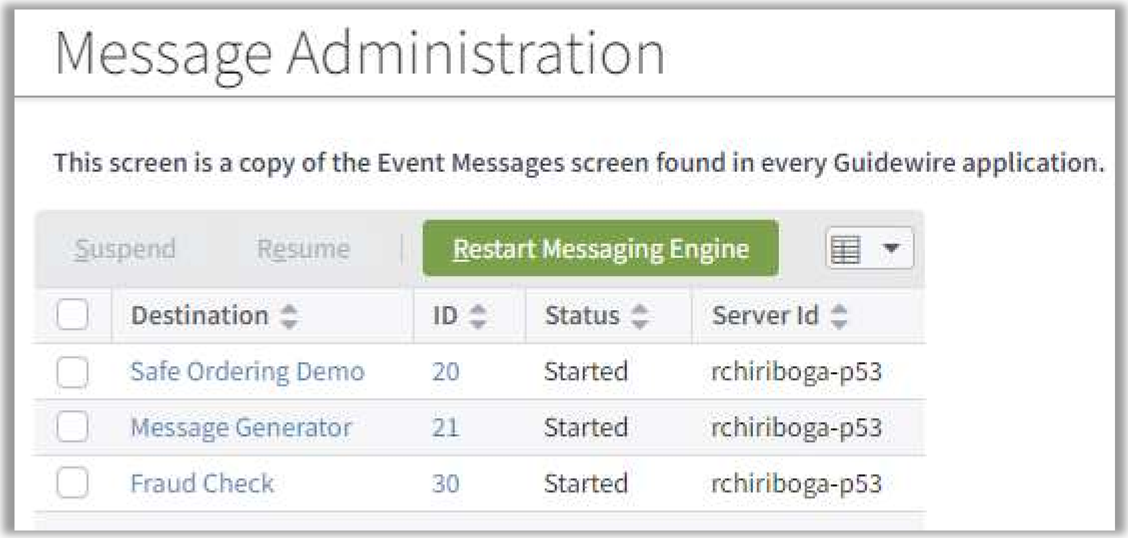
12.1.3 Verification steps

1. **Launch TrainingApp.**
2. **Verify the new destination in Administration screen.**

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

b) Click on the **Message Administration** link.

c) Verify the **Fraud Check** destination status is **Started**.



Message Administration

This screen is a copy of the Event Messages screen found in every Guidewire application.

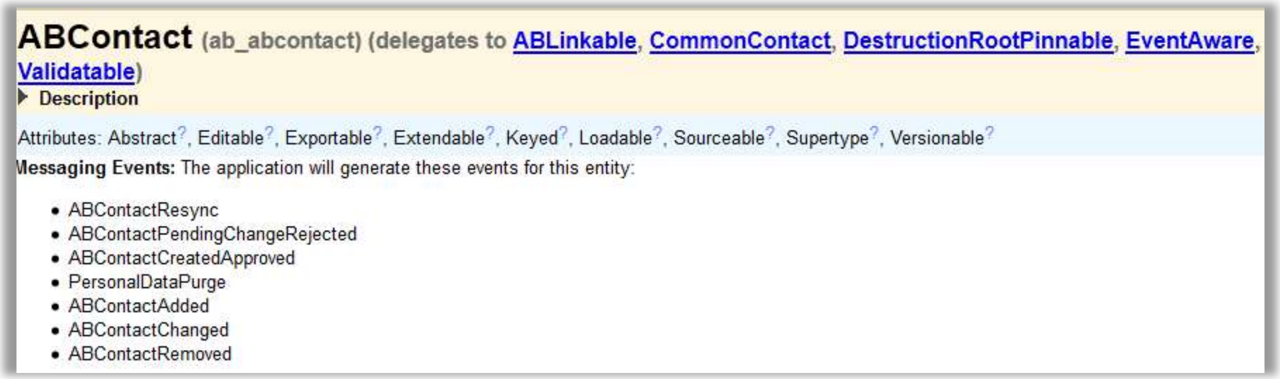
□	Destination	ID	Status	Server Id
□	Safe Ordering Demo	20	Started	rchiriboga-p53
□	Message Generator	21	Started	rchiriboga-p53
□	Fraud Check	30	Started	rchiriboga-p53

Solution 1: Configure an event aware entity and destination

1. **Verify ABContact entity delegates to EventAware delegate.**

   There are two approaches to verify that **ABContact** delegates to **EventAware**:

   - Use the data dictionary, or



ABContact (ab_abcontact) (delegates to ABLinkable, CommonContact, DestructionRootPinnable, EventAware, Validatable)

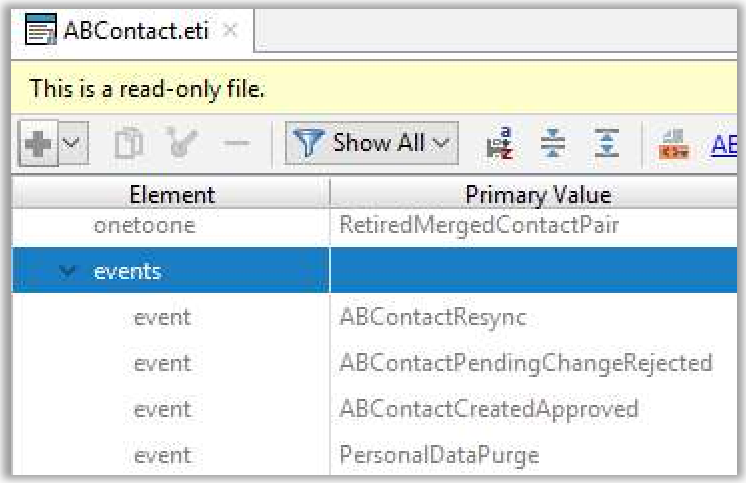
▸ Description

Attributes: Abstract?, Editable?, Exportable?, Extendable?, Keyed?, Loadable?, Sourceable?, Supertype?, Versionable?

Messaging Events: The application will generate these events for this entity:

- ABContactResync
- ABContactPendingChangeRejected
- ABContactCreatedApproved
- PersonalDataPurge
- ABContactAdded
- ABContactChanged
- ABContactRemoved

   - Verify event delegate was added to **ABContact.eti** file.



This is a read-only file.

Element	Primary Value
onetoone	RetiredMergedContactPair
events	
event	ABContactResync
event	ABContactPendingChangeRejected
event	ABContactCreatedApproved
event	PersonalDataPurge

Important!

Four custom events also exist.

2. **Create a new messaging destination called Fraud Check.**

   a) Open **messaging-config.xml** under **configuration.config.Messaging**.

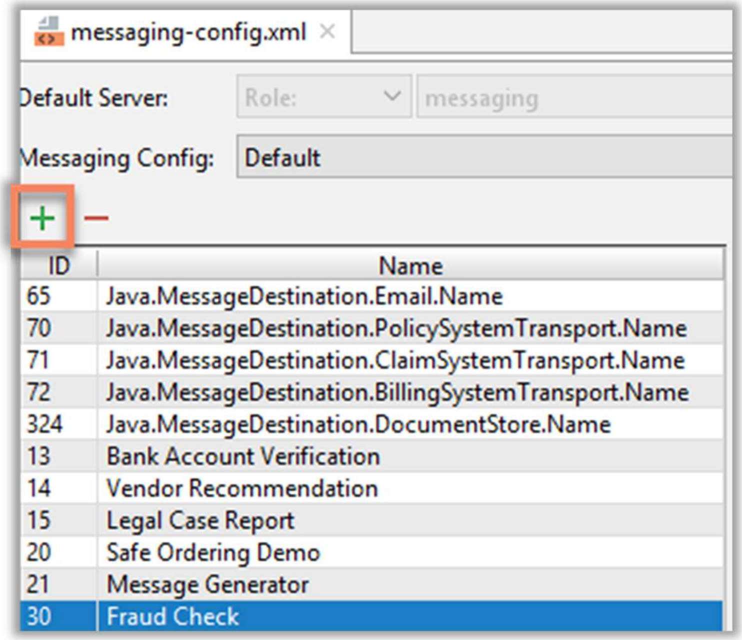
   b) Click the **PLUS** symbol to create a new destination.

   c) Populate the required fields:

   - ID = 30
   - Name = Fraud Check
   - Transport Plugin = DefaultPrintToConsoleTransport
   - Events
     - ABContactChanged
     - ABContactAdded

Important!

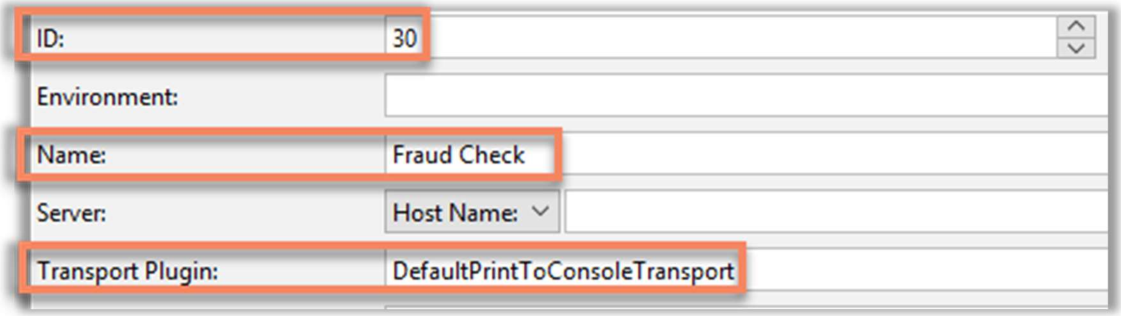
Do not change default connectivity and error parameters.


messaging-config.xml

Default Server: Role: messaging

Messaging Config: Default

ID	Name
65	Java.MessageDestination.Email.Name
70	Java.MessageDestination.PolicySystemTransport.Name
71	Java.MessageDestination.ClaimSystemTransport.Name
72	Java.MessageDestination.BillingSystemTransport.Name
324	Java.MessageDestination.DocumentStore.Name
13	Bank Account Verification
14	Vendor Recommendation
15	Legal Case Report
20	Safe Ordering Demo
21	Message Generator
30	Fraud Check

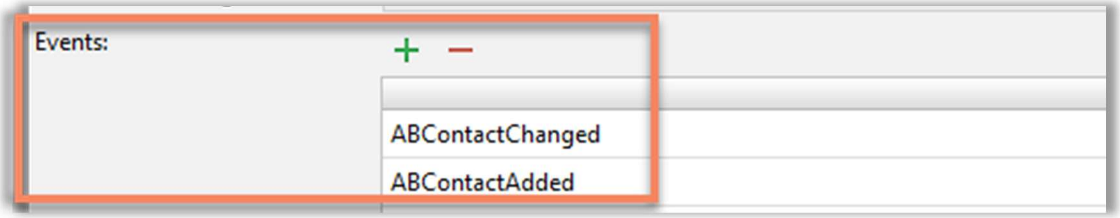

ID: 30

Environment:

Name: Fraud Check

Server: Host Name:

Transport Plugin: DefaultPrintToConsoleTransport


Events:

ABContactChanged

ABContactAdded

3. **Deploy code changes.**

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

4. **Perform verification steps**