Marketing Customization 3

What this exercise is about

In this tutorial, you will implement a new marketing activity trigger. Adding a new trigger is one of many customization options available for the Marketing tool in version 7. This is the third in a series of three labs designed to demonstrate marketing customization. The labs can be completed individually but each will build on concepts and techniques introduced in the previous lab.

The new trigger you will create allows marketing managers to start or advance in a Dialog activity when a shopper adds a product to their wish list. This trigger might be used to send a coupon to a customer to encourage them to purchase the wish list item. In this lab you will also learn how to create a more advanced marketing element template that uses shopper behavior to advance the trigger.

What you should be able to do

After completing this exercise, you should be able to:

- Define and register a marketing element implementation template that checks for specific shopper behavior
- Configure a new behavior rule
- Add a new trigger to the Management Center Dialog activity palette

Introduction

The following naming conventions are used in the exercises:

Reference Variable	Description
<WCDE_INSTALL_DIR>	WebSphere Commerce Developer installation directory
<WCDE_HOST>	Hostname for WebSphere Commerce Developer
<LAB_FILE_DIR>	Location lab files were extracted to

The MarketingCustomizationLabs.zip file that is provided with this lab contains all the new files necessary to complete this lab. There is also a snippets file provided which contains just the text to be typed in. Unless otherwise stated in the instructions, you should type in the code provided. If you find this is too much typing or need assistance solving a problem, the solution files and snippets file are available.

Requirements

Before beginning this lab, ensure you have:

- Installed WebSphere Commerce Developer 7.0

Part 1: Define the new trigger

In this part of the lab, you will plan your new trigger and collect all the information you need to proceed with the customization. This information can include parameters specified by the business user when setting up the trigger and the command that will validate the business user input. It is also common for triggers to react to specific customer behavior. If your trigger is a response to a shopper's action then you need to identify the command associated with that action. The trigger element template links all the information together.

____ 1. Identify what parameters the business user needs to specify to create the trigger. In this example, the trigger is activated based on a customer adding an item to their wish list so no additional parameters are needed.

____ 2. Since no parameters are specified for this trigger, a new task command is not needed. The purpose of the task command is only to implement the `validateParameters` method.

____ 3. Define the implementation template for the trigger. The trigger implementation template looks different than the one for targets and actions. In addition to specifying parameters and a validation task command, you can specify a task command that is called once the customer moves past the trigger. This is the `callCmdOnMatch` attribute. The trigger implementation template is shown below.

```
<Trigger type="UNIQUE_TRIGGER_NAME"
callCmdOnMatch="COMMAND_INTERFACE_NAME">
     <Parameter name="PARAMETER1" value="VALUE_OF_PARAMETER1"/>
     <Parameter name="PARAMETER2" value="VALUE_OF_PARAMETER2"/>
     <Implementation invocationType="TaskCommand">
        <Class name="TASK_COMMAND_CLASS_NAME"/>
     </Implementation>
</Trigger>
```

__ a. The completed implementation template for this trigger is shown below. It is common for a trigger definition to be quite simple when the trigger is matching a customer behavior.

```
<Trigger type="AddsToWishList"></Trigger>
```

____ 4. Define the behavior rule for the trigger. This rule specifies what type of customer behavior should activate the trigger. The behavior rule template is also an XML snippet and looks like this:

```
<BehaviorRule
   command= "COMMAND_TO_MATCH"
   processOnCommandExit="true/false"
   action="send/record/custom"
   comparison="MARKETING_comparison"
   caseSensitive="true/false"
   maxSize="MARKETING_numberOfTimes"
   maxTotalSize="MARKETING_numberOfTimes"
   relativeDays="MARKETING_daysOperator"
   numberOfTimesOperator="MARKETING_numberOfTimesOperator"
   callCmdOnMatch="BEHAVIOR_RULE_TASK_COMMAND"  >
   <Variable
      name="VARIABLE1_NAME_TO_MATCH"
      value="VARIABLE1_CONTENT_TO_MATCH"
      type="TYPE_OF_VARIABLE1"/>
</BehaviorRule>
```

__ a. The `command` attribute is the name of the command to match. When this command is invoked, the trigger is evaluated. For this lab, the command you want to match is **InterestItemAddCmd**. The `processOnCommandExit` attribute indicates whether it is a URL or controller command that is being matched. InterestItemAddCmd is a controller command so this value will be **true** indicating that the behavior rule matches when the command exits successfully. The `action` attribute specifies what to do when the behavior rule is matched. In this example you want to

signal the marketing runtime to process the associated trigger so the value is **send**. More detail on other behavior rule attributes and possible values can be found in the Information Center.

__ b. The completed behavior rule for this trigger is shown below.

```
<BehaviorRule
      command="InterestItemAddCmd"
      action="send"
      processOnCommandExit="true">
</BehaviorRule>
```

____ 5. Insert the implementation template and behavior rule in the marketing element template table, DMELETEMPLATE.

__ a. Choose a name for the trigger. This name will be used to identify the trigger in Management Center. For this lab, use the name **customAddToWishList.**

__ b. Access the development database using this URL :http://localhost/webapp/wcs/admin/servlet/db.jsp

__ c. Run the following command to insert the implementation template into the DMELETEMPLATE table. The value **1002** was chosen as a unique primary key and the value **1** signifies that this is a trigger template.

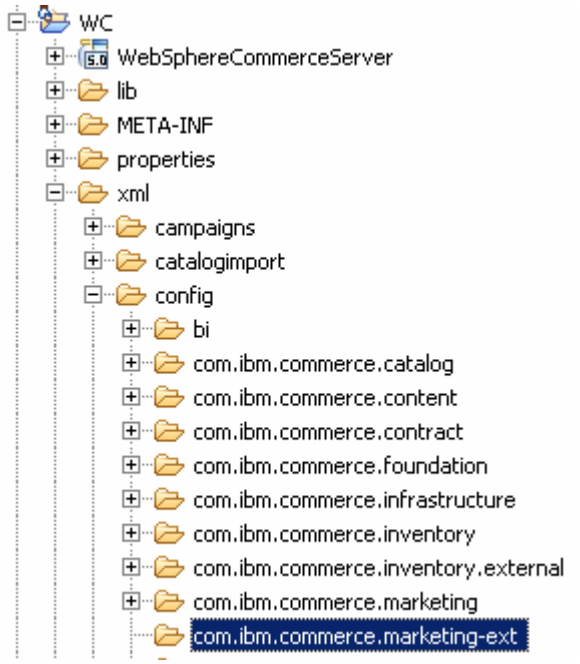
```
insert into dmeletemplate (dmeletemplate_id, dmelementtype_id, name,
implxml, behaviorxml) values (1002, 1, 'customAddToWishList',
'<Trigger type="AddsToWishList"></Trigger>', '<BehaviorRule
command="InterestItemAddCmd" action="send"
processOnCommandExit="true"></BehaviorRule>');
```

Part 2: Configure the behavior rule

In Part 1, you added a behavior rule to your trigger element template to recognize the InterestItemAddCmd command. You now need to specify the controller command name to match the behavior rule.

____ 1. Open WebSphere Commerce Developer and navigate to the **WC** package.

____ 2. Navigate to **WC > xml > config**. If it does not exist, create the folder **com.ibm.commerce.marketing-ext**.



____ 3. Within the **com.ibm.commerce.marketing-ext** folder, create the file **wc-admin-component.xml** if it does not already exist.

____ 4. The following two lines are used to configure your behavior rule. The first line specifies the name of the controller command to match. The second line specifies which Web application paths to exclude when looking for a match. Rules will not match against controller commands from the specified Web applications.

```
<_config:property name="processOnCommandExitInterestItemAddCmd"
value="com.ibm.commerce.interestitems.commands.InterestItemAddCmd"/>
<_config:property
name="processOnCommandExitWebappFilterInterestItemAddCmd"
value="/webapp/wcs/tools/servlet,/webapp/wcs/orgadmin/servlet,/webapp
/wcs/admin/servlet"/>
```

____ 5. If you already had the file **wc-admin-component.xml** defined, you can add the above lines to your existing file within the `<_config:configgrouping name="MarketingRuntime">` section. If you just created it in Step 2 above, copy the contents of the file **<LAB_FILE_DIR>/MarketingCustomization3/com.ibm.commerce.marketing-ext/wc-admin-component.xml** into your new file.

____ 6. Save the file.

Part 3: Add the new trigger to Management Center

In this part of the lab, you will update the Management Center activity builder to support the new trigger.

____ 1. Create a new OpenLaszlo file for the trigger element object definition.

__ a. In the LOBTools project, navigate to the folder **…/lzx/mycompany/marketing/objectDefinitions/activityBuilder**. Right click and select **New > File.**

Note: Note: If you did not do the previous labs you will need to create the directory structures **mycompany/marketing/objectDefinitions/activityBuilder** and **mycompany/marketing/resources**.

__ b. Name your new file **AddToWishListFlowElementObjectDefinition.lzx**. Click **Finish** to create the file.

____ 2. Create the object definition. Triggers, targets and actions are all activity builder flow elements. When you add a new flow element it must extend the class `mktFlowElementObjectDefinition`

__ a. Add the following code to the file AddToWishListFlowElementObjectDefinition.lzx. Notice the `elemTemplateType` attribute is set to Trigger.

```
<library>
      <class name="extAddToWishListElementObject"
extends="mktFlowElementObjectDefinition"
            objectType="customAddToWishList"
            displayName="Customer adds to their wish list"
            headerIcon="customAddToWishListHeaderIcon"
            flowIcon="customAddToWishListIcon"
            paletteIcon="customAddToWishListPaletteIcon"
            elemTemplateType="Trigger">

      <mktFlowElementCreateService/>
      <mktFlowElementUpdateService/>

      <dataset name="template">
            <elemTemplateName>customAddToWishList</elemTemplateName>
      </dataset>
      </class>
</library>
```

__ b. Save the file.

____ 3. Define the display icons. In the object definition above, three icon resources are used. You will define those resources now.

__ a. Navigate to the **resources** directory open the file called **ExtMarketingResources.lzx**

Note: You need to create this file if you did not do the previous labs.

__ b. Add the following code to the file between the `<library>` tags. To simplify this example, you will reuse the purchase history target icon for your new trigger.

```
<resource name="customAddToWishListPaletteIcon"
src="../../../commerce/marketing/resources/pal/purchase_history.png"/
>
<resource name="customAddToWishListIcon"
src="../../../commerce/marketing/resources/dgm/purchase_history.png"/
>
```

```
<resource name="customAddToWishListHeaderIcon"
src="../../../commerce/marketing/resources/hdr/purchase_history.png"/
>
```

__ c. Save the file.

____ 4. Register your custom OpenLaszlo file with the Management Center Marketing tool**.**

__ a. Navigate to **…/lzx/commerce/marketing** and open the file **MarketingExtenstionsLibrary.lzx**

__ b. Add the name of the file you just created. If the resources file is not already included from the first lab, add it as well.

```
<include
href="../../mycompany/marketing/objectDefinitions/activityBuilder/Add
ToWishListFlowElementObjectDefinition.lzx "/>
```

__ c. Save the file.

____ 5. Create an instance of the trigger object**.**

__ a. Navigate to **…/lzx/commerce/marketing/objectDefinitions/activityBuilder** and open the file **FlowPathElementObjectDefinition.lzx**.

__ b. Scroll to the bottom of the file and add an instance of the `extAddToWishListElementObject` class to the bottom of the list of element objects. Enter the following line of code.

```
<extAddToWishListElementObject />
```

__ c. Save the file.

____ 6. Add the trigger to the Dialog activity builder palette.

__ a. Navigate to **…/lxz/commerce/marketing/propertiesViews** and open the file **DialogActivityBuilder.lzx.**

__ b. Scroll through the file until you locate the triggers group.

```
<Group resourceBundle="mktMarketingResources" name="triggers" hel
    <Element objectType="wait"/>
    <Element objectType="registers"/>
    <Element objectType="purchases"/>
    <Element objectType="customerParticipatesInSocialCommerce"/>
    <Element objectType="customerAbandonsShoppingCart"/>
    <Element objectType="birthdayTrigger"/>
    <Element objectType="customerIsInSegment"/>
</Group>
```

__ c. At the bottom of the triggers group, add a new element with `objectType` **customAddToWishList**. The updated group should look like the following screen capture.

```
<Group resourceBundle="mktMarketingResources" name="triggers" he
    <Element objectType="wait"/>
    <Element objectType="registers"/>
    <Element objectType="purchases"/>
    <Element objectType="customerParticipatesInSocialCommerce"/>
    <Element objectType="customerAbandonsShoppingCart"/>
    <Element objectType="birthdayTrigger"/>
    <Element objectType="customerIsInSegment"/>
    <!-- New trigger -->
    <Element objectType="customAddToWishList"/>
</Group>
```

__ d. Save the file. You have now updated the palette for the Dialog activity builder. You can also add this trigger to the Dialog activity template builder located in a separate file in the same directory.

____ 7. Compile the Management Center project. Right click the **LOBTools** project and select **Build OpenLaszlo Project.**

____ 8. Update the Struts extension file. In order to read the new target information from the database and correctly display it in Management Center, you need to add a new trigger to the Struts configuration file.

__ a. Navigate to **LOBTools/WebContent/WEB-INF** and open the file **struts-extension.xml.**

__ b. Within the `<action-mappings>` block, add the following new trigger.

```
<action path="/SerializeActivityElement-customAddToWishList"
include="/jsp/commerce/marketing/restricted/SerializeActivitygenericE
lement.jsp" />
```

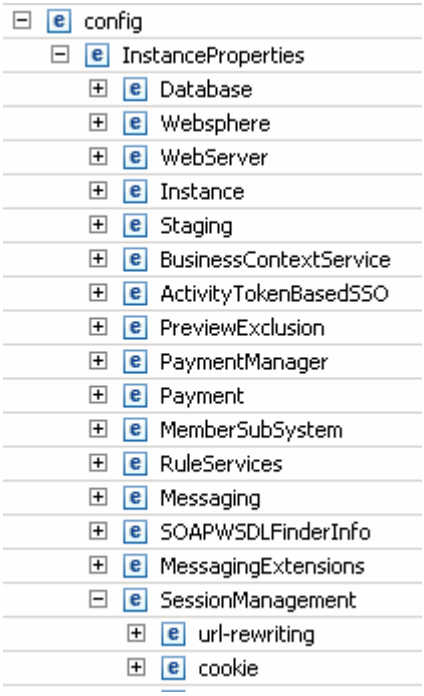
__ c. Save the file.

Part 4: Configure the server to track online behavior

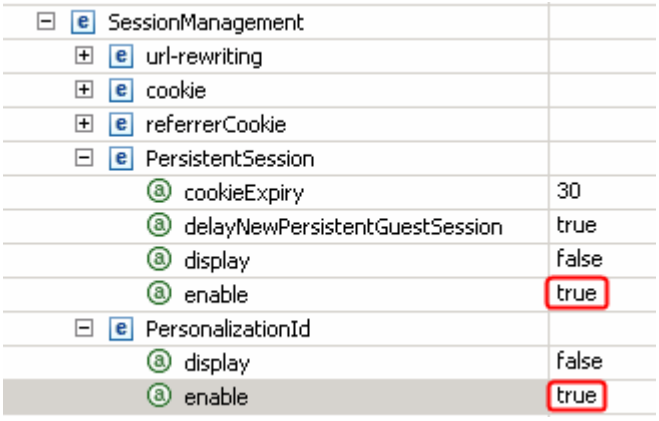
In this section, you will configure your WebSphere Commerce test environment to track online behavior of shoppers. This step is necessary for the “Add to wish list” trigger to work. If you have already completed the Precision Marketing lab you can skip this part but remember to restart your test server after completing Part 3.

____ 1. Navigate to **WC > xml >config** and open the file **wc-server.xml**.

____ 2. Expand the **config**, **InstanceProperties** and **SessionManagement** nodes



__ a. Enable PersistentSession and PersonalizationId by setting the enable attribute to true for each.



__ b. Switch to the **Source** view of the file. Enable the SensorEventListener by setting its enable attribute to true.

```
<component
compClassName="com.ibm.commerce.marketing.dialog.trigger.SensorEventL
istener"
enable="true" name="SensorEventListener">
      <property display="false">
        <start enabled="true" />
      </property>
</component>
```

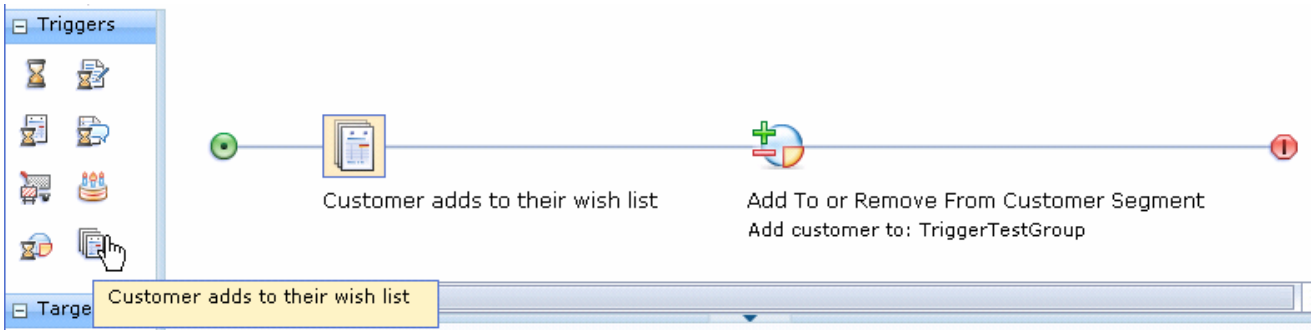
____ 3. Start your test server. If it is already running, restart it.

Part 5: Test your new trigger

In this part of the lab, you will test the changes you have made.

____ 1. Launch Management Center using the URL **https://<WCDE_HOST>:8000/lobtools**.

____ 2. Create a Dialog activity using your new trigger. Save and activate the activity.



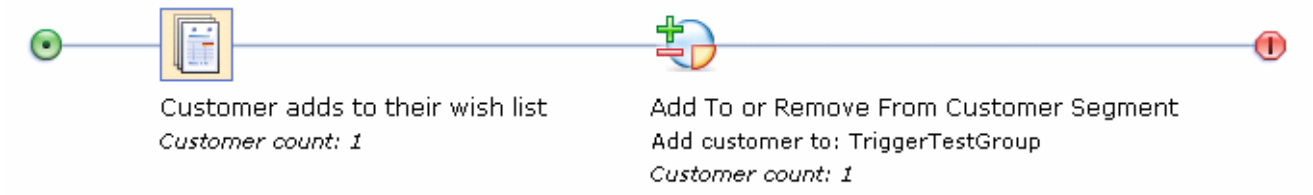
____ 3. Launch the Madisons starter store using the URL **http://<WCDE_HOST>/webapp/wcs/stores/servlet/Madisons/index.jsp**

____ 4. Register a new shopper or log in to an existing account.

____ 5. Navigate to a category page and add an item to your wish list.

____ 6. You can check that a trigger was activated by examining the DMTRIGSND table. The trigger will be sent the next time the SendMarketingTriggers scheduled job runs.

____ 7. Once the trigger has been sent, you can confirm the shopper was added to the member group by examining the DMMBRGRPPZN table. You can also check the statistics for the Dialog activity.



Part 6: What you did in this exercise

In this tutorial you learned how to create and test a new marketing trigger element.

You should now understand how to complete the following tasks:

- Define and register a marketing element implementation template that checks for specific shopper behavior
- Configure a new behavior rule
- Add a new trigger to the Management Center Dialog activity palette