ORACLE CRM On Demand

Oracle CRM On Demand Desktop Customization Guide

Version 5.2 Revision A

October 2018

Contents

1 What's New in This Release

What's New in Oracle CRM On Demand Desktop Customization Guide, Version 5.2 Revision A

Table 1 lists changes in this version of the documentation to support Version 5.2 Revision A of the software.

Table 1. What's New in Oracle CRM On Demand Desktop Customization Guide, Version 5.2 Revision A

Topic	Description
"Adding Custom Objects to Oracle CRM On Demand Desktop" on page 73	Modified topic. This procedure has been updated to add in the following metadata files that are required for the procedure: ■ forms_12.xml ■ business_logic.js ■ forms.js ■ package_res.xml ■ views.xml
Appendix A, "XML Code for Service Request Types"	Modified topic. The examples in this appendix have been updated.

What's New in Oracle CRM On Demand Desktop Customization Guide, Version 5.2

No new features have been added to this guide for this release. This guide has been updated to reflect only product version changes.

2 Adding One-to-Many and Many-to-Many Associations Between Different Object Types

This chapter describes the procedures to add one-to-many (1:M) associations and many-to-many (M:M) associations between different object types. It includes the following topics:

Adding One-to-Many Associations Between Different Object Types

A one-to-many association type describes the association between two different object types when a parent object can have many child objects, but each child object can be linked only to one parent object; that is, the child object can have only one link to the parent object.

Before You Begin

Review the chapter on customization in *Oracle CRM On Demand Desktop Administration Guide*.

This procedure shows how to implement a link from an Account object to a CustomObject4 object. Use this example as a reference to add links between other parent and child objects, and adjust them for your specific requirements. In this procedure, you update the following metadata files:

- od_basic_mapping.xml
- od_meta_info.xml
- connector_configuration.xml
- business_logic.js
- views.xml
- forms_12.xml

For information on the contents of these files, see About the Metadata Files Updated During Customization on page 33.

To add a one-to-many association between different object types

1. Add a link field on the parent object:
   a. Add new fields to the Account type in the od_basic_mapping.xml file, where:
      - *CustomObject4Id* is the name of the link field where the id attribute of the child object is stored.

- *CustomObject4Name* is the name of the field where the name of child object is stored, as shown in the following XML code:

```
<field id="CustomObject4Id">
   <reader>
      <mapi_user>
         <user_field id="od CustomObject4Id" ol_field_type="1"></user_field>
         <convertor>
            <binary_hexstring/>
         </convertor>
      </mapi_user>
   </reader>
   <writer>
      <binhex_link>
         <link_writer>
            <outlook_user>
               <user_field id="od CustomObject4Id" ol_field_type="1"></
user_field>
               <convertor>
                  <binary_hexstring/>
               </convertor>
            </outlook_user>
         </link_writer>
         <resolved_writer>
            <outlook_user>
               <user_field id="od CustomObject4Name" ol_field_type="1"></
user_field>
               <convertor>
                  <string/>
               </convertor>
            </outlook_user>
         </resolved_writer>
      </binhex_link>
   </writer>
 </field>
<field id="CustomObject4Name">
   <reader>
      <mapi_user>
         <user_field id="od CustomObject4Name" ol_field_type="1"></
user_field>
         <convertor>
            <string/>
         </convertor>
      </mapi_user>
   </reader>
   <writer>
      <outlook_user>
         <user_field id="od CustomObject4Name" ol_field_type="1"></
user_field>
         <convertor>
            <string/>
         </convertor>
```

```
        </outlook_user>
      </writer>
    </field>
```

You can use the ver attribute on the <type> and <field> elements in the od_basic_mapping.xml file to apply the changes made inside those elements:

- <type> element. If the ver attribute is used during development on the <type> element, then Oracle CRM On Demand Desktop does not apply any change to the description for the object that the id attribute defines until the value of the ver attribute is increased.
- <field> element. If the ver attribute is used during development on the field element, then Oracle CRM On Demand Desktop does not apply any change to the field description until the value of the ver attribute is increased.

b Add the new fields to Account type in the od_meta_info.xml file:

```
<field Name="CustomObject4Id" Label="CustomObject4Id" DataType="DTYPE_ID"
IsRefObjId="yes" RefObjTypeId="CustomObject4" IsFilterable='no'/>
```

c Extend the <links> section for Account type in the connector_configuration.xml file with the CustomObject4Id field shown in bold font in the following sample XML code:

```
<type id="Account" state_field="ObjectState">
   <view label="#obj_account" label_plural="#obj_account_plural"
small_icon="type_image:Account:16" normal_icon="type_image:Account:24"
large_icon="type_image:Account:48"></view>
   <synchronizer name_format=":[:(AccountName):]" threshold="3">
      <links>
         <link>PrimaryContactId</link>
         <link>ParentAccountId</link>
         <link>OwnerId</link>
         <link>CustomObject4Id</link>
      </links>
      <natural_keys>
         <natural_key>
            <field>AccountName</field>
            <field>Location</field>
         </natural_key>
      </natural_keys>
   </synchronizer>
</type>
```

2 Define the one-to-many relation in the JavaScript code by extending the create_ondemand_meta_scheme2 function in the business_logic.js file with the following functions: add_direct_link() and add_simple_trigger():

- add_direct_link("Account", "CustomObject4", "CustomObject4Id", false, null, true, "AccountStatus");

  The add_direct_link function defines the one-to-many relation in the data model and accepts the following parameters, where:

  - *from_type* is the parent object type, for example, Account.
  - *to_type* is the child object type, for example, CustomObject4.

- *link_field* is the link field name on the parent object, for example, CustomObject4Id.
- *required_link* is a Boolean data type.
- *refresh_required* is used to refresh other form when link changed, if required.
- *primary_on_parent* is a Boolean data type.
- *status* is the service string field name of the parent object type.

■ `add_simple_trigger(form_helpers.native_show, null, "CustomObject4", null, "show");`

The add_simple_trigger function determines the action that is executed when the child object is opened from the parent form so that the child object's form is opened. The function accepts following parameters, where:

- *handler* is the function that is called when the trigger is called. Returns a function type.
- *type* is the object type on which the trigger is called. Returns a string type.
- *link_to* is the link type on which the trigger is called. Returns a string type.
- *tag* is the link tag on which trigger is called. Returns a string type.
- *operation* is the operation on which the trigger is called. Returns a string type.

In this case, the add_simple_trigger() function defines that the show operation on the CustomObject4 object triggers a call to the native_show() function in the form_helpers.js file when a UI form of CustomObject4 is displayed. The call passes null values for the type and tag parameters because they are irrelevant for the show operation.

3 Add the selectors options that specify the settings for the Salesbook dialog box, which display the CustomObject4 records by locating the `//selectors options comment` in the business_logic.js file, and add your selector options below the comment. For example:

```
scheme.objects.get_object("CustomObject4").selectors_options = {
    "source": {
       "caption": "obj_customobj4_plural",
       "view_id": "custom_objects4:salesbook",
       "search_by": ["Name"],
       "online": {
          "like_template": "*{keyword}*"
       }
    }
};
```

where:

■ *custom_objects4:salesbook* is the id attribute of the child object's Salesbook view.

■ *Name* is the field defined for the CustomObject4 object to be used for searching purposes.

4 Add the view definition for the child object to be displayed on the Salesbook view to the views.xml file by locating the <res_root><data_view> elements in views.xml, and placing the following XML code in the <data_view> element:

```
<view id="view_od_custom_objects4">
    <image_list>
       <res_id type="normal">type_image:Generic:16</res_id>
    </image_list>
    <columns>
       <column width="305" sort="asc">
          <heading type="string">head_name</heading>
          <field>Name</field>
       </column>
    </columns>
</view>
```

where:

- <*column*> is the column element of the grid.
- <*heading*> contains the resource id attribute of the column heading.
- <field> indicates the object's field name taken from the od_basic_mapping.xml file.
- sort is the sort parameter, which can be asc (ascending) or desc (descending).

5 Add controls to the forms in the forms_12.xml file as in the following sample code that extends the Account (parent object) form with labels, lookup control, and buttons:

```
<cell size="25">
    <stack layout="horz" spacing="5" padding="2">
       <!-- left side captions -->
       <cell size="112">
          <stack spacing="5" layout="vert" padding="4">
             <cell size="21">
                <static id="lbl_co1_name">
                   <text>#lbl_co4</text>
                </static>
             </cell>
          </stack>
       </cell>
       <!-- left side fields -->
       <cell>
          <stack layout="vert" spacing="5">
             <cell size="21">
                <stack layout="horz" spacing="3">
                   <cell>
                      <autocomplete id="CustomObject4Id">
                         <field>CustomObject4Id</field>
                         <source type="CustomObject4" format=":[:(Name):]"></
source>
                      </autocomplete>
                   </cell>
                   <cell size="22" attraction="far">
                      <button id="btn_CustomObject4Id" image="lookup_button">
                         <text>...</text>
                      </button>
                   </cell>
                </stack>
             </cell>
```

```
            </stack>
          </cell>
        </stack>
    </cell>
```

In the XML code, <autocomplete> is a control that the user can use to choose any object. Oracle CRM On Demand Desktop uses an autocomplete control to establish a relationship between objects, for example, to link an account with a CustomObject4. The autocomplete control XML parameters are as follows:

- <source>. Indicates the object type of the records that are displayed using this control. This parameter has the following attributes:
  - type. The type of the objects displayed in the control.
  - format. The display format.
- <field>. The name of the field attached to this control. This parameter can be empty.

6 To register autocomplete control in the forms.js file, add the following code to the parent Account object's form handler:

```
register_autocomplete_control(ctx, "CustomObject4", "CustomObject4Id",
"btn_CustomObject4Id");
```

The generic syntax of the function is as follows:

```
function register_autocomplete_control(ctx, link_to, autocomplete_ctrl,
show_salesbook_btn)
```

The register_autocomplete_control() function accepts the following parameters, where:

- *ctx* is the form context object, which is typically passed to the function without changing it.
- *link_to* is the type of the child object.
- *autocomplete_ctrl* is the id attribute for the autocomplete control in the forms12.xml file.
- *show_salesbook_btn* is the id attribute for the Salesbook button in the forms12.xml file.

Adding Many-to-Many Associations Between Different Object Types

A many-to-many (M:M) association type describes the association between two different object types when an object of each type can have many related objects of another type.

Before You Begin

Review the chapter on customization in *Oracle CRM On Demand Desktop Administration Guide*.

To add an association to Microsoft Outlook you must start by defining two objects that make up the association. Because associations are not available as standalone types in Oracle CRM On Demand, Oracle CRM On Demand Desktop adds them to Microsoft Outlook by querying the subtypes and then joining them to one object. For example, if you want to add an association between Account and CustomObject types, then you make two queries, one that returns all objects of CustomObject type under Account type, while the second returns all objects of type Account under Custom Object type. Data sets returned by these two queries are then joined as a single query, which is an Account.CustomObject.Association type in Microsoft Outlook. To define settings for these requests, several auxiliary types are defined in the od_meta_info.xml file. However, in the connector_configuration.xml and the od_basic_mapping.xml files, only one association type is defined, which is shown in this topic. The following procedure adds a many-to-many association between an Account object and a CustomObject object.

In this procedure, you update the following metadata files:

- od_meta_info.xml
- od_basic_mapping.xml
- connector_configuration.xml
- business_logic.js
- views.xml
- data_sources.xml
- business_logic.js
- forms_12.xml

For information on the contents of these files, see About the Metadata Files Updated During Customization on page 33.

Adding a many-to-many association between different object types

1. Define the association and auxiliary types in the od_meta_info.xml file:

   a. Add the object types that make up the association by adding the following XML code that defines two objects in the od_meta_info.xml. For brevity, these type definitions contain only the required fields and a name field.

```
<object TypeId="Account" Label="#obj_account"
LabelPlural="#obj_account_plural" ViewMode="Sales Rep" IntObjName="Account"
XmlElemName="Account" XmlCollectionElemName="ListOfAccount"
EnableGetIDsBatching="yes" ProtectFkVals="yes">
   <extra_command_options>
      <option Name="UseDefaultViewMode" Value="true" Scopes="Dedup" />
   </extra_command_options>
   <field Name="Id" Label="Id" DataType="DTYPE_ID" IsPrimaryKey="yes"
IsFilterable='no'/>
   <field Name="ModId" Label="Mod Id" DataType="DTYPE_INTEGER"
IsTimestamp="yes" IsFilterable='no' IsHidden="yes"/>
   <field Name="AccountName" Label="Account Name" DataType="DTYPE_TEXT"/>
</object>
```

```
<object TypeId="CustomObject" Label="#obj_customobject"
LabelPlural="#obj_CustomObject_plural" ViewMode="Sales Rep"
IntObjName="CustomObject" XmlElemName="CustomObject"
XmlCollectionElemName="ListOfCustomObject" EnableGetIDsBatching="yes">
   <extra_command_options>
      <option Name="UseDefaultViewMode" Value="true" Scopes="Dedup" />
   </extra_command_options>
   <field Name="Id" Label="Id" DataType="DTYPE_ID" IsPrimaryKey="yes"
IsFilterable='no'/>
   <field Name="ModId" Label="Mod Id" DataType="DTYPE_INTEGER"
IsTimestamp="yes" IsFilterable='no' IsHidden="yes"/>
   <field Name="CustomObjectName" Label="CustomObject Name"
DataType="DTYPE_TEXT"/>
</object>
```

b Add the object types that are used to compose queries in this example: Account.CustomObject and CustomObject.Account.

Adding the Account.CustomObject and CustomObject.Account object types is the same as adding other object types, except that these object type definitions include the following two fields with special fields:

- The Id field for the first object in the association.

  For the Account.CustomObject object, the Id field for the first object is AccountId. For the CustomObject.Account object, the Id field for the first object is CustomObjectId.

- The Id field for the second object in the association.

  For the Account.CustomObject object, the Id field for the second object is CustomObjectid and for the CustomObject.Account object, the Id field for the second object is AccountId.

The Id field for the first object in the association must always include the attribute values described in the following table.

Attribute Name	Attribute Value	Attribute Description
IsRefObjId	yes	Indicates to Oracle CRM On Demand Desktop that this field references a real object in the data model. **NOTE:** Always set this value to yes.
RefObjTypeId	For Account.CustomObject: Account For CustomObjectAccount: CustomObject	The type identifier of the first object that makes up an association.
RefObjIsParent	yes	Indicates to Oracle CRM On Demand Desktop that the referenced object is the parent object. **NOTE:** Always set this value to yes.

The Id field for the second object must include the attribute *IsPrimaryKey* with the value set to yes.

The following XML code defines the two objects that are used to compose requests in the od_meta_info.xml file. For the sake of brevity, these type definitions contain only the required fields and the name fields:

```
<object TypeId='Account.CustomObject Label=' CustomObject' LabelPlural='
CustomObject' ViewMode='Sales Rep' IntObjName='Account' XmlElemName='
CustomObject' XmlCollectionElemName='ListOf CustomObject'>
   <extra_command_options>
      <option Name="UseDefaultViewMode" Value="true" Scopes="Dedup" />
   </extra_command_options>
   <field Name="Id" Label="Id" DataType="DTYPE_ID" />
   <field Name="ModId" Label="Mod Id" DataType="DTYPE_INTEGER"
IsTimestamp="yes" />
   <field Name='AccountId' Label='Account Id' DataType='DTYPE_ID'
IsRefObjId="yes" RefObjTypeId="Account" RefObjIsParent="yes"/>
   <field Name=' CustomObjectId' Label=' CustomObject Id' DataType='DTYPE_ID'
IsPrimaryKey='yes' />
```

```
   <field Name="AccountName" Label="Account Name" DataType="DTYPE_TEXT" />
   <field Name=" CustomObjectName" Label=" CustomObject Name"
DataType="DTYPE_TEXT" />
</object>

<object TypeId="CustomObject.Account" Label="Account" LabelPlural="Account"
ViewMode="Sales Rep" IntObjName="Contact" XmlElemName="Account"
XmlCollectionElemName="ListOfAccount" EnableGetIDsBatching="yes">
   <extra_command_options>
      <option Name="UseDefaultViewMode" Value="true" Scopes="Dedup" />
   </extra_command_options>
   <field Name="Id" Label="Id" DataType="DTYPE_ID" />
   <field Name="ModId" Label="Mod Id" DataType="DTYPE_INTEGER"
IsTimestamp="yes" IsFilterable='no'/>
   <field Name=" CustomObjectId" Label="Contact Id" DataType="DTYPE_ID"
IsRefObjId="yes" RefObjTypeId=" CustomObject" RefObjIsParent="yes" />
   <field Name="AccountId" Label="Account Id" DataType="DTYPE_ID"
IsPrimaryKey="yes"  />
   <field Name="AccountName" Label="Account Name" DataType="DTYPE_TEXT" />
   <field Name=" CustomObjectName" Label="Contact Full Name"
DataType="DTYPE_TEXT" />
</object>
```

c Add the intersection object types that are translated into Microsoft Outlook association types in this example: Account.CustomObject.Association and CustomObject.Account.Association.

Adding the Account.CustomObject.Association and CustomObject.Account.Association object types is the same as adding other object types, except that these object type definitions must include the following fields:

- The association Id field
- The association ModId field
- The Id field for the first object in the association

  For the Account.CustomObject.Association object, the Id field for the first object is AccountId. For the CustomObject.Account.Association object, the Id field for the first object is CustomObjectId.

- The Id field for the second object in the association.

  For the CustomObject.Account.Association object, the Id field for the second object is CustomObjectid. For the CustomObject.Account object, the Id field for the second object is AccountId.

The Id field for the first object in the association must always include the attribute values described in the following table.

Attribute Name	Attribute Value	Attribute Description
IsRefObjId	yes	Indicates to Oracle CRM On Demand Desktop that this field references a real object in the data model. **NOTE:** Always set the attribute name value to yes.
RefObjTypeId	For Account.CustomObject.Association: Account For CustomObject.Account.Association: CustomObject	The type identifier of the first object that makes up an association.
RefObjIsParent	yes	Indicates to Oracle CRM On Demand Desktop that the referenced object is the parent object. **NOTE:** Always set the attribute name value to yes.
ActualFldName	Id	The Id field name of the object type specified in the RefObjTypeId attribute. Typically the value is *Id*.
RefExposedToUI	true	Indicates to Oracle CRM On Demand Desktop that it must query the referenced object.

Attribute Name	Attribute Value	Attribute Description
ActualObjTypeId	For Account.CustomObject.Association: Account For CustomObject.Account.Association: CustomObject	The type Id of the first object that makes up an association.
IsReadonly	yes	Indicates to Oracle CRM On Demand Desktop that the field is read only.
IsRequired	yes	Indicates to Oracle CRM On Demand Desktop that the field is required.
IsFake	yes	Indicates to Oracle CRM On Demand Desktop that the field is not used.

The Id field for the second object in the association must always include the attributes described in the following table.

Attribute Name	Attribute Value	Attribute Description
IsRefObjId	yes	Indicates to Oracle CRM On Demand Desktop that this field references a real object in the data model. **NOTE:** Always set the attribute name value to yes.
RefObjTypeId	For Account.CustomObject.Association: Account.CustomObject For CustomObject.Account.Association: CustomObject.Account	The type Id of the first auxiliary object that makes up an association from Step b on page 14.
RefObjIsParent	yes	Indicates to Oracle CRM On Demand Desktop that the referenced object is the parent object. **NOTE:** Always set the attribute name value to yes.
ActualFldName	For Account.CustomObject.Association: AccountId For CustomObject.Account.Association: CustomObjectId	The name of the id field for the first object in association from Step b on page 14.
RefExposedToUI	true	Indicates to Oracle CRM On Demand Desktop that it must query the referenced object.
ActualObjTypeId	For Account.CustomObject.Association: Account.CustomObject For CustomObject.Account.Association: CustomObject.Account	The type Id of the first auxiliary object that makes up an association from Step b on page 14.

Attribute Name	Attribute Value	Attribute Description
IsReadonly	yes	Indicates to Oracle CRM On Demand Desktop that the field is read only.
IsRequired	yes	Indicates to Oracle CRM On Demand Desktop that the field is required.
IsFake	yes	Indicates to Oracle CRM On Demand Desktop that the field is not used.

The ModId field in the association object's definition must also include the IsFake attribute with the value set to yes. The Id field in the association object's definition must typically include the IsCompositeId attribute with the value set to yes, and the IsFake attribute with the value set to yes.

The following is the XML code that defines the two objects that are used to compose requests in the od_meta_info.xml file:

```
<object TypeId="Account.CustomObject.Association"
Label="Account.CustomObject.Association"
LabelPlural="Account.CustomObject.Associations" ViewMode="Sales Rep"
IsAssociation="yes" EnableMirrorRequests="no" EnableGetIDsBatching="true">
   <extra_command_options>
      <option Name="UseDefaultViewMode" Value="true" Scopes="Dedup" />
   </extra_command_options>
   <field Name="AccountId" Label="Account Id" ActualFldName="Id"
ActualObjTypeId="Account" IsReadonly="yes" IsRequired="yes"
DataType="DTYPE_ID" IsNullable="no" IsFilterable="no" IsRefObjId="yes"
RefObjTypeId="Account" RefObjIsParent="yes" IsFake="yes" />
   <field Name="CustomObjectId" Label="CustomObject Id"
ActualFldName="CustomObjectId" ActualObjTypeId="Account.CustomObject"
IsReadonly="yes" IsRequired="yes" DataType="DTYPE_ID" IsNullable="no"
IsFilterable="no" IsRefObjId="yes" RefObjTypeId="Account.CustomObject"
RefObjIsParent="yes" IsFake="yes" />
   <field Name="Id" Label="Id" IsPrimaryKey="yes" IsReadonly="yes"
IsRequired="yes" DataType="DTYPE_ID" IsFilterable="no" IsCompositeId="yes"
IsFake="yes" />
   <field Name="ModId" Label="Mod Id" DataType="DTYPE_INTEGER"
IsTimestamp="yes" IsFake='yes'/>
</object>
<object TypeId="CustomObject.Account.Association"
Label="CustomObject.Account.Association"
LabelPlural="CustomObject.Account.Associations" ViewMode="Sales Rep"
IsAssociation="yes" EnableMirrorRequests="yes" EnableGetIDsBatching="true">
   <extra_command_options>
```

```
      <option Name="UseDefaultViewMode" Value="true" Scopes="Dedup" />
    </extra_command_options>
    <field Name="CustomObjectId" Label="CustomObject Id" ActualFldName="Id"
RefExposedToUI="true" ActualObjTypeId="CustomObject" IsReadonly="yes"
IsRequired="yes" DataType="DTYPE_ID" IsNullable="no" IsFilterable="no"
IsRefObjId="yes" RefObjTypeId="CustomObject" RefObjIsParent="yes"
IsFake="yes" />
    <field Name="AccountId" Label="Account Id" ActualFldName="AccountId"
RefExposedToUI="true" ActualObjTypeId="CustomObject.Account"
IsReadonly="yes" IsRequired="yes" DataType="DTYPE_ID" IsNullable="no"
IsFilterable="no" IsRefObjId="yes" RefObjTypeId="CustomObject.Account"
RefObjIsParent="yes" IsFake="yes" />
    <field Name="Id" Label="Id" IsPrimaryKey="yes" IsReadonly="yes"
IsRequired="yes" DataType="DTYPE_ID" IsFilterable="no" IsCompositeId="yes"
IsFake="yes" />
    <field Name="ModId" Label="Mod Id" DataType="DTYPE_INTEGER"
IsTimestamp="yes" IsFake='yes'/>
</object>
```

2 Add the intersection object definition to the od_basic_mapping.xml file by adding the code for the <type> element between the <database><types>...</types></database> elements in the following sample code:

```
<type id="Account.CustomObject.Association" icon=" type_image:Generic:16">
     <field id ="AccountId">
       <type>
          <simple type="binary"/>
       </type>
     </field>
     <field id ="CustomObjectId">
       <type>
          <foreign_key>
             <type_id>CustomObjectId</type_id>
          </foreign_key>
       </type>
     </field>
     <field id ="LeftStatus">
       <type>
          <simple type="string"/>
       </type>
     </field>
     <field id ="RightStatus">
       <type>
          <simple type="string"/>
```

```
        </type>
      </field>
</type>
```

The <type> element has the attributes described in the following table.

Attribute Name	Attribute Value	Attribute Description
id	Account.CustomObject.Association	The unique ID of this object type in Oracle CRM On Demand Desktop. Set this attribute to the same value specified in the TypeId attribute of the <object> element for the object in the od_meta_info.xml file.
icon	type_image:Generic:16	Defines the icon that is displayed for this type of object, for example, on a form Caption, in Outlook views, and so on. The value of this attribute is the resource key of an image, for example, a PNG file, 16 x 16 pixels, to be used and defined in any Oracle CRM On Demand Desktop resource file.

You must define the fields of the new object. For more information, see Chapter 3, "Adding Custom Fields to Oracle CRM On Demand Desktop."

This example adds the following fields to the Microsoft Outlook data model:

- AccountId
- CustomObjectId

The example also adds the following service fields (fields used by Oracle CRM On Demand Desktop internally that are not synchronized with Oracle CRM On Demand):

- LeftStatus
- RightStatus

These fields must be added to each child object and are processed by Oracle CRM On Demand Desktop.

Depending on whether the object to which you are adding an ID field for an association is stored in Microsoft Outlook storage or DB FACADE storage, the syntax of the <field> element is going to be one of the following:

- If the CustomObject is stored in DB FACADE storage, as in this example, then use the following XML code:

```
<type>
    <foreign_key>
       <type_id>CustomObjectId</type_id>
    </foreign_key>
</type>
```

- If the CustomObject is stored in Microsoft Outlook storage, then use the following XML code:

```
<type>
    <simple type="binary"/>
</type>
```

For more information on Microsoft Outlook and DB FACADE storage, see About Microsoft Outlook and DB FACADE Storage on page 37.

3 Update the connector_configuration.xml with the new intersection object definition.

The changes made to the od_meta_info.xml and od_basic_mapping.xml files describe only the new object in a data structure, in either Microsoft Outlook or on a server. However, to synchronize items of this object type between Oracle CRM On Demand and Outlook, you must add the new object definition by adding the following structure to the <types> element in the connection_configuration.xml file:

```
<type ... >
    <view ... />
    <synchronizer ... />
       <links>
          <link ... />
          ...
       </links>
       <natural_keys>
          <natural_key>
             <field ... />
             ...
          </natural_key>
          ...
       </natural_keys>
    </synchronizer>
</type>
```

For this specific example, use the following sample XML code:

```
<type id="Account.CustomObject.Association">
    <view label="Account.CustomObject.Association" label_plural="Account
CustomObjects" small_icon="type_image:Generic:16"
normal_icon="type_image:Generic:24" large_icon="type_image:Generic:48"
suppress_sync_ui="true"></view>
    <synchronizer name_format=":[:(CustomObjectName) :]" threshold="20">
       <links>
          <link required="true" owned="true">AccountId</link>
          <link required="true" owned="true">CustomObjectId</link>
       </links>
       <natural_keys>
          <natural_key>
             <field>AccountId</field>
             <field>CustomObjectId</field>
          </natural_key>
```

```
      </natural_keys>
    </synchronizer>
</type>
```

In this example, note the following:

- The <type> element has a single attribute defined as in the following table.

Attribute Name	Attribute Value	Attribute Description
id	Account.CustomObject. Association	The unique ID of this object type in Oracle CRM On Demand Desktop. Set the ID value to the same value specified previously for the id attribute of the <type> element in od_basic_mapping.xml in Step a on page 13.

- The <view> element defines some of the UI settings. It contains the attributes in the following table that you must define.

Attribute Name	Attribute Value	Attribute Description
label	Account.CustomObject. Association	The label that Oracle CRM On Demand Desktop uses for this type of object in the user interface. In particular, for synchronization issues, Oracle CRM On Demand Desktop uses this attribute to resolve duplicates and conflicts and to synchronize confirmation tabs of the Control Panel.
label_plural	Account CustomObjects	A plural label that Oracle CRM On Demand Desktop uses for this type of object in the user interface. In particular, for synchronization issues, Oracle CRM On Demand Desktop uses this attribute to resolve duplicates and conflicts and to synchronize confirmation tabs of the Control Panel.

Attribute Name	Attribute Value	Attribute Description
small_icon	type_image:Generic:16	The definition of the icon to display for this type of object on a Control Panel, as a child element icon on a filter tree, in various lists of the Control Panel. The value of this attribute is a resource key of an image file, for example, a PNG file, 16 x 16 pixels, to be used and defined in any Oracle CRM On Demand Desktop resource file.
normal_icon	type_image:Generic:24	Definition of the icon displayed for this type of object on a Control Panel, as a top level element icon on a filter tree. The value of this attribute is a resource key of an image file, for example, a PNG file, 24 x 24 pixels, to be used and defined in any Oracle CRM On Demand Desktop resource file.
large_icon	type_image:Generic:48	Defines the icon to be displayed for this type of object, for example, the icon on a Sync dialog box. The value of this attribute is a resource key of an image file, for example, a PNG file, 40 x 40 pixels, to be used and defined in any Oracle CRM On Demand Desktop resource file.
suppress_sync_ui	true	Setting to a value of true hides this object from the Filters Panel.

- The <synchronizer> element defines the sync settings for the object, such as link fields, natural keys, how this object is displayed if it appears in any list of the Control Panel, and so on. This example defines the attributes in the following table.

Attribute Name	Attribute Value	Attribute Description
name_format	:[:(CustomObjectName):]	This attribute's value defines a mask that is used to build this object identifier in any list of a Control Panel, such as, sync issues, duplicates, and so on. In this case, the value of the CustomObjectName field is displayed.
threshold	20	The minimum number of objects that Oracle CRM On Demand Desktop can delete at once without a user response. Oracle CRM On Demand Desktop synchronizes the deletions of less than 20 custom objects from Microsoft Outlook to Oracle CRM On Demand. If it does not delete objects automatically, it displays a confirmation dialog box for the user to confirm the deletion of more than 20 custom objects. This feature protects a user from deleting many custom objects accidentally. If a user deletes objects accidentally, then those objects are deleted from Oracle CRM On Demand after synchronization.

The <synchronizer> element also contains <links> and <natural_keys> elements.

  - The <links> element defines a set of link fields of the object. For this example, the link fields are AccountId and CustomObjectId. Oracle CRM On Demand Desktop requires knowledge of these fields to build an objects dependency tree during synchronization. Oracle CRM On Demand Desktop also must be able to determine which fields contain Ids of other objects, so that their values are converted properly from the Oracle CRM On Demand Id format to the Microsoft Outlook Id format and from the Microsoft Outlook Id format to the Oracle CRM On Demand.
  - The <natural_keys> element contains a set of field sets (<natural_key>) that are used to identify duplicated records during synchronization. A record is identified as a duplicate if there are two matching records in Oracle CRM On Demand and Microsoft Outlook, in which the match is defined by the fields specified in natural keys. In this example, the fields are AccountId and CustomObjectId.

4 Add the selectors' options in the SalesBook view. SalesBook is a dialog box with a built-in view control that displays a list of available objects for linking; with the SalesBook dialog, a user can pick any record and associate selected object:

a Add selectors options to the business_logic.js file.

The options in this example specify the basic settings for the Salesbook dialog box, which displays the CustomObject1 records. In the business_logic.js file, locate the `//selectors options comment`, and add your selectors options below the comment, as follows:

```
scheme.objects.get_object("CustomObject4").selectors_options = {
   "source": {
      "caption": "obj_customobj4_plural",
      "view_id": "custom_objects4:salesbook",
      "search_by": ["Name"],
      "online": {
         "like_template": "*{keyword}*"
      }
   }
};
```

where:

- *custom_objects4:salesbook* is the identifier of the child object's Salesbook view.
- *Name* is the field defined for the CustomObject4 object to be used for search purposes.

b Define the SalesBook view and form in the views.xml file by locating the `<res_root><data_view>` elements and placing the following XML code sample in the `<data_view>` element:

```
<view id="view_od_custom_objects4">
   <image_list>
      <res_id type="normal">type_image:Generic:16</res_id>
   </image_list>
   <columns>
      <column width="305" sort="asc">
         <heading type="string">head_name</heading>
         <field>Name</field>
      </column>
   </columns>
</view>
```

where:

- `<column>` is a column element of the grid
- `<heading>` contains The resource Id of the column heading
- `<field>` indicates the object's field name taken from the od_basic_mapping.xml file

You can set the sort attribute to asc (ascending) or desc (descending).

The code to display Associated custom_objects1 view on the Account form is as follows:

```
<view id="custom_objects1:form_view">
   <image_list>
      <res_id type="normal">type_image:Generic:16</res_id>
   </image_list>
   <columns>
      <column width="305" sort="asc">
         <heading type="string">head_name</heading>
```

```
            <field>
               <poly_join link="CustomObjectId">
                  <if_type name="Custom Object 1">
                     <simple>Name</simple>
                  </if_type>
               </poly_join>
            </field>
         </column>
      </columns>
   </view>
```

5 Add the data source.

If the right object of the association is stored within Microsoft Outlook storage, then you must add the data source by adding the following in the <joint> element to the data_sources.xml file, as follows:

```
<data_source name="ActionJointCustomObject1" type_id="Account.Custom Object 1"
resolve_rate="1"/>
```

6 Add the JavaScript code that defines the new many-to-many association by adding the following code to the business_logic.js file:

```
var account_co1 =
    add_mvg_link("Account", "Custom Object 1", "CustomObject1Id", "AccountId",
                 "Account.Custom Object 1", "AccountId", "CustomObjectId",
                 "LeftStatus", "RightStatus",
                 null, null,
                 true, true, true, true);
    account_co1.mvg1.dialog_template_params = {
       "dialog_caption": "#obj_account_co1_plural",
       "autocomplete_display_format": ":[:(Name):]",
       "associations_view_caption": "#head_associations_co1",
       "associations_view_id": "custom_objects1:mvg",
       "primary_selector_display_format": ":[:(Name):]"
    }
account_co1.mvg2.dialog_template_params = {
       "dialog_caption": "#obj_contact_account_plural",
       "autocomplete_display_format": ":[:(Name) :]",
       "associations_view_caption": "#head_associations_accounts",
       "associations_view_id": "accounts:mvg",
       "primary_selector_display_format": ":[:(Name) :]"
 }
```

The add_mvg_link function accepts the following parameters, where:

- *left_type* is the type of the first linked object, for example, Account.
- *right_type* is the type of the second linked object, for example, CustomObject1.
- *left_obj_primary* is the field of the Account object. In the example, this field contains the primary identifier for the CustomObject1 object. If no primary is required, then set this argument to null.
- *right_obj_primary* is the field of the CustomObject1 object. This field contains the primary identifier for the Account object. If no primary is required, then set this argument to null.

- *assoc_type* is the identifier of the association type described in the od_meta_info.xml file and the od_basic_mapping.xml file.
- *left_link* is the field of the association object that contains the identifer of the Account.
- *right_link* is the field of the association object that contains the identifier of the CustomObject1.
- *left_assoc_status* is the field of the association that contains the status of the Account.
- *right_assoc_status* is the field of the association that contains the status of the CustomObject1.
- Null parameter (unused).
- Null parameter (unused).
- *left_primary_refresh_required* is used to update the object identifier for the primary field of the Account that you specify in the left_obj_primary attribute, if the primary object is changed, and if the left_primary_refresh_required attribute is true.

  If you use the <writer class="link_fields" ...> element in the od_basic_mapping.xml file for this field, then set the left_primary_refresh_required attribute to true.
- *right_primary_refresh_required* is used to update the object identifier for the primary field of the CustomObject1 that you specify in the right_obj_primary attribute, if the primary object is changed, and if the right_primary_refresh_required attribute is true.

  If you use the <writer class="link_fields"...> element in the od_basic_mapping.xml file for this field, then set the right_primary_refresh_required attribute to true.
- *assoc_left_link_refresh_required* is used to update the AccountId field of the association that you specify in the left_link attribute, if the Account object is changed.

  If you use the <writer class="link_fields" ...> element in the od_basic_mapping.xml file for this field, then set the assoc_left_link_refresh_required attribute to true.
- *assoc_right_link_refresh_required* is used toupdate the CustomObjectId field of the association that you specify in the right_link attribute, if the CustomObject1 object is changed.

  If you use the <writer class="link_fields" ...> element in the od_basic_mapping.xml file for this field, then set the assoc_right_link_refresh_required attribute to true.

To add a new association without primary links, see the description of the left_obj_primary and right_obj_primary arguments in this topic. The LeftStatus and RightStatus fields contain the status objects, such as unsaved, deleted, and so on.

7 Extend the user interface (UI) as follows:

a Add the MVG (multi-value group) control.

  An MVG control displays an association with other objects. This association can be a many-to-many association, or a many-to-one association.

  In the example, the CustomObject1 form contains the MVG control with the primary selector and button to expose many-to-many associations with Account to the UI. The advantage of this control is that it occupies less space.

- Extend the Custom Object 1 form in the forms_12.xml file with the following XML code sample:

```
<cell>
   <stack layout="horz" spacing="5">
      <cell>
         <mvg_primary_selector id="AccountToCustomObject1">
            <source type="Account.Custom Object 1" left_id="CustomObjectId"
item_value="AccountId" display_format=":[:(AccountName):]"></source>
            <field>AccountId</field>
         </mvg_primary_selector>
      </cell>
      <cell size="21">
         <button id="btn_mvgAccount">
            <text>...</text>
         </button>
      </cell>
   </stack>
</cell>
```

- Extend the Custom Object 1 form handler in forms.js with the following JavaScript code:

  If Custom Object 1 has a primary Account, then use the following

```
register_mvg_dialog(ctx, "Account", "AccountToCustomObject1",
"btn_mvgAccount");
```

  If Custom Object 1 has no primary Account, then use the following

```
register_mvg_dialog(ctx, "Account", "AccountToCustomObject1",
"btn_mvgAccount", { "tag": "mvg", "dialog_id": "mvg_dialog_no_primary"});
```

b Add the View control with buttons.

In this example, the Account form contains a View control with buttons to expose many-to-many associations from CustomObject1 to the UI. The advantage of this control is that it displays all associated objects directly on a form.

- Extend the Account form in the forms_12.xml file with the following XML code:

```
<!-- CustomObject1 view -->
<cell size="13">
   <stack layout="horz">
      <cell size="105">
         <static id="0x20050"><text>#view_od_co1</text>
         </static></cell>
      <cell>
         <stack layout="vert">
            <cell size="7"></cell>
            <cell size="1">
               <edge id="0x20051"/></cell>
         </stack>
      </cell>
   </stack>
```

```
</cell>
<cell size="22">
   <stack layout="horz" padding="5">
      <cell size="150">
         <static id="0x20050"><text>#lbl_primary_co1</text>
         </static>
      </cell>
      <cell>
         <mvg_primary_selector id="AccountToCustomObject1">
            <source type="Account.Custom Object 1" left_id="AccountId"
item_value="CustomObjectId" display_format=":[:(Name):]"></source>
            <field>CustomObject1Id</field>
         </mvg_primary_selector>
      </cell>
   </stack>
</cell>
<cell>
   <stack layout="vert" spacing="5">
      <cell>
         <stack layout="horz" padding="5">
            <cell>
               <data_view id="co1_view" tab_order="56">
                  <source type="auto" name="Account.Custom Object 1"></source>
                  <view id="custom_objects1:form_view"></view>
                  <restriction>
                     <group link="and">
                        <binary field="AccountId" condition="eq">
                           <value type="variable">id()</value>
                        </binary>
                        <binary field="RightStatus" condition="ne">
                           <value type="string">deleted</value>
                        </binary>
                        <binary field="LeftStatus" condition="ne">
                           <value type="string">unsaved</value>
                        </binary>
                     </group>
                  </restriction>
               </data_view>
            </cell>
         </stack>
      </cell>
      <cell size="22" attraction="far">
         <stack layout="horz" spacing="5" padding="5">
            <cell size="140" attraction="far">
               <button id="btn_remove_co1"><text>#lbl_remove_co1</text>
               </button>
            </cell>
            <cell size="140" attraction="far">
               <button id="btn_co1"><text>#lbl_add_co1</text>
               </button>
            </cell>
         </stack>
      </cell>
```

```
      </stack>
   </cell>
   <!-- end CustomObject1 view -->
```

If the object on the right side of the association is stored within Microsoft Outlook storage, then locate the following XML element:

```
<source type="auto" name="Account.Custom Object 1"></source>
```

Replace the element with the following element:

```
<source type="auto" name="ActionJointCustomObject1"></source>
```

The Microsoft Outlook view control XML attributes are described in the following table.

Attribute	Values
name (for <source> element)	A valid object type.
id (for <view> element)	The view identifier from the views.xml file.

- Extend the Account form handler in the forms.js file with the following JavaScript code:

```
register_view_control_with_button(ctx, "Custom Object 1", "co1_view", null,
{"delete_action": "remove_association", "btn_remove_association":
ctx.form.btn_remove_co1});
```

3 Adding Custom Fields to Oracle CRM On Demand Desktop

To customize and extend Oracle CRM On Demand Desktop for your requirements you can add new fields on existing objects. This chapter describes the procedures to add new fields through examples. Before you begin this chapter, review the chapter on customization in *Oracle CRM On Demand Desktop Administration Guide*. This chapter contains the following topics:

About the Metadata Files Updated During Customization

You typically update the following metadata files available in the customization package, depending on the type of customization you are performing:

- **business_logic.js.** Defines the business logic for product behavior and the basic data model for object relations.
- **connector_configuration.xml.** Contains the definitions of objects that are synchronized, the criteria that Oracle CRM On Demand Desktop uses to detect duplicate objects in the Oracle CRM On Demand database, and defines the preset filters for a custom synchronization.
- **data_sources.xml.** Describes the data sources that are used for joining fields between objects that are used in online lookups.
- **forms.js.** Contains the JavaScript code to set up the multiselect picklist layout controls.

- **forms_12.xml.** Contains the UI layouts that define the form layout for each object, the field validation rules on a form, the business logic in JavaScript, and the controls that Oracle CRM On Demand Desktop uses on a form. This file also defines the fields that Oracle CRM On Demand Desktop uses to store references between objects.
- **od_basic_mapping.xml.** Contains the mapping data that defines the field mapping between Oracle CRM On Demand Desktop and Microsoft Outlook, and between Oracle CRM On Demand Desktop and Oracle CRM On Demand. This file describes objects to add to Microsoft Outlook, defines the form that Oracle CRM On Demand Desktop uses to display an object in Microsoft Outlook, and defines a set of custom Microsoft Outlook views that Oracle CRM On Demand Desktop applies to an object.
- **od_meta_info.xml.** Contains the data definitions for the object types that Oracle CRM On Demand Desktop supports, the object fields and their types, and the XML element names that Oracle CRM On Demand Desktop uses to build an Oracle CRM On Demand message.
- **package_res.xml.** Contains the symbolic string definitions that define various resources for the customization package.
- **views.xml.** Defines the views that Oracle CRM On Demand Desktop uses in Oracle CRM On Demand Desktop forms and Microsoft Outlook windows.

For more information on these files, see the chapter on customization in *Oracle CRM On Demand Desktop Administration Guide*.

Adding Custom Fields to Oracle CRM On Demand Desktop

This procedure shows how you customize Oracle CRM On Demand Desktop so that a new field can be synchronized with Oracle CRM On Demand in addition to the default fields. In this customization procedure, you add a new text field called Membership to the Contact page in Oracle CRM On Demand Desktop. This field is displayed under an existing field called Lead Source.

In this example, the contact object has a new field called Membership in Oracle CRM On Demand.

Before You Begin

To complete this customization procedure, you require the following:

- An XML editor of your choice to modify XML files.
- A file comparison (diff) tool to compare the changes you have made to the customized files with the default customization package and to verify that the changes are free of errors. Even a minor error such as adding or removing an extra space unintentionally is problematic. The comparison tool can be useful when trying to detect such problems. A tool similar to Windiff is recommended. Windiff is available from the following Web site:

  http://www.grigsoft.com/download-windiff.htm

To add a custom field to Oracle CRM On Demand Desktop

1. In Oracle CRM On Demand, create a new short text field called Membership for the Contact object.
2. Before customizing Oracle CRM On Demand Desktop, save a copy of the default customization package.

   After you begin customizing, you can compare the changed files with the default files to verify your changes.
3. To add a custom field, you must modify the following XML files in the customization package:
   - od_meta_info.xml

     For more information, see Changing the XML Code in the od_meta_info.xml File on page 35.
   - od_basic_mapping.xml

     For more information, see Changing the XML Code in the od_basic_mapping.xml File on page 38.
   - forms_12.xml

     For more information, see Changing the XML Code in the forms_12.xml File on page 40.
   - package_res.xml

     For more information, see Changing the XML Code in the package_res.xml File on page 42.

   For information on the contents of these files, see About the Metadata Files Updated During Customization on page 33.
4. After making the changes, package all the files using a ZIP utility, and upload the package to Oracle CRM On Demand.
5. Unpublish any previously published package, and publish this new customization package for the role associated with the user doing the customization.

   The next time you attempt to synchronize using Oracle CRM On Demand Desktop, the new customization package is automatically downloaded so that the contact page in Oracle CRM On Demand Desktop includes the new field, Membership.

Changing the XML Code in the od_meta_info.xml File

In this procedure, you add the field definition for the new Membership field in the od_meta_info.xml file that contains the data definitions. The field definition must be included in all metadata objects that are being exposed in the example:

- Contact
- Account.Contact

- Opportunity.Contact

NOTE: The field name used in the <field> element must match the name of the field in the object WSDL (Web Services Description Language). In this example, for the Membership field, the name in the contact custom WSDL is *stMembership*. So, before adding the element, you must look up and use the field name in the WSDL in Oracle CRM On Demand.

The convention used for WSDL names is as follows:

- For default Oracle CRM On Demand fields that have not been exposed to Oracle CRM On Demand Desktop, the naming convention is the name of the field without spaces. For example, Account Name, becomes AccountName in the WSDL.
- For custom fields that you have created in Oracle CRM On Demand, for the WSDL representation of these fields, spaces are replaced with underscores, and a prefix is added to the field name. The prefixes that are used for the field types are listed in Table 2. For example, if you create a field called JVD Text of type Text (Short) on an object, then in the WSDL, this field is called *stJVD_Text*.

Table 2. Prefixes and Data Types for Oracle CRM On Demand Field Types

Field Type	Prefix	Data Type
Checkbox	b	DTYPE_BOOL
Currency	c	DTYPE_NUMBER
Date	d	DTYPE_DATE
Date/Time	dt	DTYPE_UTCDATETIME
Integer	i	DTYPE_INTEGER
Multi-Select Picklist	mspl	DTYPE_CSVLIST
Number	n	DTYPE_NUMBER
Percent	pc	DTYPE_NUMBER
Phone	ph	DTYPE_TEXT
Picklist	pl	DTYPE_TEXT
Text (Long)	lt	DTYPE_TEXT
Text (Short)	st	DTYPE_TEXT

To change the XML code in the od_meta_info.xml file

1 Edit the od_meta_info.xml file, and locate the <object> element.

- For the Contact object, the <object> element looks similar to the following:

```
<object TypeId = "Contact"… >
```

- For the Account.Contact object, the <object> element looks similar to the following:

```
<object TypeId='Account.Contact' …>
```

- For the Opportunity.Contact object, the <object> element looks similar to the following:

```
<object TypeId='Opportunity.Contact' …>
```

2 Add a new <field> element to define the new Membership field in each object:

```
<field Name="stMembership" Label="stMembership" DataType="DTYPE_TEXT" />
```

The following XML shows the <object> element and the new <field> element in bold font:

```
<object TypeId='Account.Contact' Label='Contact' LabelPlural='Contact'
ViewMode='Sales Rep' IntObjName='Account' SiebMsgXmlElemName='Contact'
SiebMsgXmlCollectionElemName='ListOfContact'>
    <field Name='AccountId' Label='AccountId' DataType='DTYPE_ID'
IsRefObjId="yes" RefObjTypeId="Account" RefObjIsParent="yes"/>
    <field Name='AccountLocation' Label='AccountLocation' DataType='DTYPE_TEXT'
/>
    <field Name='AccountMainPhone' Label='AccountMainPhone'
DataType='DTYPE_TEXT' />
    <field Name='AccountName' Label='AccountName' DataType='DTYPE_TEXT' />
    …
    …
<field Name='PrimaryContact' Label='PrimaryContact' DataType='DTYPE_BOOL' />
    <field Name='Role' Label='Role' DataType='DTYPE_TEXT' />
    <field Name='RoleList' Label='RoleList' DataType='DTYPE_TEXT' />
    <field Name="stMembership" Label="stMembership" DataType="DTYPE_TEXT" />
</object>
```

In this sample XML code, the st prefix has been added for the new <field> element, stMembership. Add a prefix to a field only in the od_meta_info.xml file.

3 Verify your changes, and save the od_meta_info.xml file.

About Microsoft Outlook and DB FACADE Storage

Oracle CRM On Demand Desktop supports two types of storage:

- Microsoft Outlook storage.

  Microsoft Outlook storage is the original legacy storage supported by Oracle CRM On Demand Desktop available through Microsoft Outlook. In prior releases of Oracle CRM On Demand Desktop, all objects participating in synchronization were stored in Microsoft Outlook storage. Microsoft Outlook storage is still used for top level object types such as Account, Contact, and so on, that must be displayed in a Microsoft Outlook folders list.

- DB FACADE storage.

  DB FACADE storage is an additional storage type supported by Oracle CRM On Demand Desktop Object types that are not visible in a Microsoft Outlook folders list and that do not have a UI form. Place types such as picklists, associations, and child objects into this storage, because DB FACADE storage is optimized for these types.

Changing the XML Code in the od_basic_mapping.xml File

The od_basic_mapping.xml file provides the data mapping between Oracle CRM On Demand and Oracle CRM On Demand Desktop. In this procedure, you add the Membership field mapping to the object <type> element.

If you are adding a field to the object stored in DB FACADE storage in Oracle CRM On Demand Desktop, then the syntax for adding the <field> element is as follows:

```
<field id ="Membership">
     <type>
        <simple type="string"/>
     </type>
</field>
```

For more information on DB FACADE storage, see About Microsoft Outlook and DB FACADE Storage on page 37.

To change the XML Code in the od_basic_mapping.xml file

1 In the od_basic_mapping.xml file, locate the Contact <type > element, which looks similar to the following:

```
<type id="Contact"…>
```

2 Add the following <field> mapping element after any existing <field> mapping elements:

```
<field id="stMembership">
    <reader>
       <mapi_user>
          <user_field id="od Membership" ol_field_type="1"></user_field>
          <convertor><string/></convertor>
       </mapi_user>
    </reader>
    <writer>
       <outlook_user>
          <user_field id="od Membership" ol_field_type="1"></user_field>
          <convertor> <string/></convertor>
       </outlook_user>
    </writer>
</field>
```

A field map has a reader and a writer definition for a custom field as shown in this example. You choose your own name for the custom field, and you must use the right converter for converting the field type. For example, to convert an incoming value of integer type to string, you must use the following structure:

```
<converter>
    <string/>
</converter>
```

The following XML code shows the Contact <type> element and the <field> mapping element highlighted in bold font:

```
<type id="Contact" predefined_folder="10" ver="7">
       <form message_class="IPM.Contact.OnDemand.Contact"
icon="type_image:Contact:16" large_icon="type_image:Contact:32"
display_name="#obj_contact">OnDemand Contact</form>
       <alt_messageclasses>
          <alt_messageclass ext="Private" display_name="#obj_private_contact"
icon="type_image:Contact.Private:16"
large_icon="type_image:Contact.Private:32">OnDemand Contact</alt_messageclass>
       </alt_messageclasses>
       <upgrade>
          <from message_class="IPM.Contact"
target_message_class="IPM.Contact.OnDemand.Contact.Private"></from>
       </upgrade>
       <custom_views default_name="#view_crm_and_personal_contacts">
          <view id="all_contacts" name="#view_crm_and_personal_contacts"></view>
       </custom_views>
       <field id="CellularPhone">
          <reader>
             <mapi_std>
                <mapi_tag id="0x3A1C0000"></mapi_tag>
                <convertor>
                   <string/>
                </convertor>
             </mapi_std>
          </reader>
          <writer>
             <outlook_std>
                <outlook_field id="MobileTelephoneNumber"></outlook_field>
                <convertor>
                   <string/>
                </convertor>
             </outlook_std>
          </writer>
       </field>

       <field id="stMembership">
          <reader>
            <mapi_user>
             <user_field id="od Membership" ol_field_type="1"></user_field>
             <convertor><string/></convertor>
            </mapi_user>
          </reader>
          <writer>
            <outlook_user>
             <user_field id="od Membership" ol_field_type="1"></user_field>
             <convertor> <string/></convertor>
            </outlook_user>
          </writer>
       </field>
…
…
</type>
```

3 Verify your changes, and save the forms_12.xml file.

Changing the XML Code in the forms_12.xml File

The new custom field must be displayed on the UI layout. The forms_*nn*.xml file is used to define the physical UI layout. Use Forms_12.xml for Microsoft Outlook 2007 and Microsoft Outlook 2010, which are the versions of Microsoft Outlook supported by Oracle CRM On Demand Desktop.

In this example, you add the new Membership field under the existing Lead Source field in the Contact page.

To change the XML Code in the forms_12.xml File

1 Edit the forms_12.xml file.

2 Place the new field in a cell.

A label or a field must always be placed in a cell in the XML structure. Cells are aligned vertically or horizontally by enclosing them in stacks. Complete stacks are placed in cells, allowing them to be placed either vertically or horizontally again. The size of the cell is also specified on the cell. If the cell is part of a horizontal stack, then the size specification determines the width of the cell. If the cell is part of a horizontal stack, then the size specification determines the height of the cell. When you add your cell to a stack, you must increase the size of the cell that contains that stack. Otherwise, your field might be hidden in the UI.

The cell for the field label of the Membership field is as follows:

```
<cell size="22">
    <static id="lbl_stmembership" tab_order="14">
       <text>#lbl_membership</text>
    </static>
</cell>
```

The #lbl_membership data is a reference to the string displayed in the UI that is set up in the package_res.xml file.

The cell with the field is similar to the following example:

```
<cell size="22">
    <edit id="stMembership" tab_order="14">
       <field value="string">stMembership</field>
    </edit>
</cell>
```

This XML section specifies a text box that is linked to the new field. The field name must match the field created in the od_meta_info.xml file in Changing the XML Code in the od_meta_info.xml File on page 35.

NOTE: In the above cell examples, the values of the id and tab_order parameters must be unique. However, the tab_order parameter is optional, so you can remove it when testing the changes.

3 Add the new Membership field under the existing Lead Source field in the Contact page, and increase the size of the horizontal stack so that the new field is visible between the existing Lead Source field and the Description text box.

The XML changes to the forms_12.xml file include:

- Increasing the size of the cell from 210 to 250 to accommodate the new custom field, Membership. The element for the cell that holds the horizontal stack is as follows:

```
<cell size="250">
```

- Inserting the field label for Membership under the existing label for Lead Source.
- Inserting the text box for Membership under the field for Lead Source.

These changes are shown in bold in the following example XML:

```
<cell size="250">
    <stack layout="horz">
      <cell size="7">
        <stack layout="vert" spacing="1">
          <cell size="1"/>
          <cell>
            <static id="lbl_required_sign">
              <text>#lbl_required_sign</text>
            </static>
          </cell>
        </stack>
      </cell>
      <cell>
        <stack layout="horz" spacing="5">
          <!-- left side captions -->
          <cell size="140">
            <stack spacing="5" layout="vert">
              …
              …
              <cell size="22">
                <static id="0x20016" tab_order="15"><text>#lbl_lead_source</
text>
                </static>
              </cell>
              <cell size="22">
                <static id="lbl_membership">
                  <text>#lbl_membership</text>
                </static>
              </cell>
            </stack>
          </cell>
          <!-- left side fields -->
          <cell>
            <stack layout="vert" spacing="5">
              …
              …
            <cell size="22">
              <combobox id="LeadSource" tab_order="16">
                <field>LeadSource</field>
                <source type="ContactLeadSourcePicklist"
field="Value"format=":[:(Label):]">
                </source>
              </combobox>
            </cell>
```

```
                    <cell size="22">
                         <edit id="stMembership" max_chars="100">
                            <field value="string">stMembership</field>
                         </edit>
                    </cell>
                       …
                       …
                    </stack>
                 </cell>
              </stack>
           </cell>
        </stack>
     </cell>
```

4 Verify your changes and save the forms_12.xml file.

Changing the XML Code in the package_res.xml File

The package_res.xml file contains the string translations for the string references that are used by the fields, which makes translating the UI easier because all the labels are in one place. You must add a new reference to this file for the new field label as described in this procedure.

The text that you enter for the <str> element is visible in the UI as the field name.

The value of the key attribute of the <str> element must be correctly referenced in the forms_12.xml file to display the field name, as shown in this example.

To change the XML code in the package_res.xml file

1 Edit the package_res.xml file.

2 Add the string reference for the Membership field as highlighted in bold font in the following example XML code:

```
<res_root>
    …
    …
 <!-- Contact form -->
    <str key="head_contact_details">Contact Details</str>
    <str key="lbl_job_title">Job title:</str>
    <str key="lbl_contact_team">Contact Team:</str>
    <str key="lbl_save_correspondence_to_crm">Automatically save contact emails
to CRM</str>
    <str key="lbl_never_email">Never Email</str>
    <str key="head_comments">Comments</str>
    <str key="head_business_addresses">Business Addresses</str>
    <str key="head_private_addresses">Personal Addresses</str>
    <str key="btn_account_address_caption">Account Address...</str>
    <str key="btn_contact_address_caption">Contact Address...</str>
    <str key="lbl_membership">Membership: </str>
```

```
        …
        …
    </res_root>
```

3 Verify your changes and save the package_res.xml file.

Adding Text Fields to Oracle CRM On Demand Desktop

This procedure shows how to add a text field called Department to an Oracle CRM On Demand Desktop form. Use this example as a reference to add other text fields and adjust the fields to your specific requirements. Before you begin this chapter, review the chapter on customization in *Oracle CRM On Demand Desktop Administration Guide*.

The following procedure adds a field called Department to the Contact form in the UI.

In this procedure, you make sure the field is available through Oracle CRM On Demand Web services and then you update the following metadata files:

- od_meta_info.xml
- od_basic_mapping.xml
- forms_12.xml
- package_res.xml

For information on the contents of these files, see About the Metadata Files Updated During Customization on page 33.

To add a text field to Oracle CRM On Demand Desktop

1 Make sure that the field, Department, is available through Oracle CRM On Demand Web services.

   For information on using Web services, see *Oracle Web Services On Demand Guide*.

2 Update the metadata files for the new Department field:

   a Update od_meta_info.xml by adding Department to the metadata objects where making the objects available is necessary, in this example, the Contact object.

   b Locate the <object> element by TypeId attribute set to Contact. For example:

```
<object TypeId="Contact"…>
```

   c Add the new <field> element (Department) to define the field:

```
<field Name='Department' Label='Department' DataType='DTYPE_TEXT'/>
```

   where:

   - *Name* is the name of new field. You can define any value, which is used to access this field value.

- *Label* is the label that is used to reference this field in the UI, for example, on the Oracle CRM On Demand Desktop Control Panel.
- *DataType* is the type of the field. For text fields, set its value to DTYPE_TEXT. For check boxes or other Boolean fields, set its value to DTYPE_BOOL. For more information on data types, see Table 2 on page 36.

3 Update the od_basic_mapping.xml file:

a Locate the <type> element by the id attribute set to Contact. For example:

```
<type id="Contact"…>
```

b Add the new <field> element for Department to define the mapping:

```
<field id="Department">
   <reader>
      <mapi_user>
         <user_field id="od Department" ol_field_type="1"></user_field>
         <convertor>
            <string/>
         </convertor>
      </mapi_user>
   </reader>
   <writer>
      <outlook_user>
         <user_field id="od Department" ol_field_type="1"></user_field>
         <convertor>
            <string/>
         </convertor>
      </outlook_user>
   </writer>
</field>
```

If you are adding a field to an object located in the DB FACADE storage, then the syntax for the field definition differs as in the following XML code:

```
<field id=" Department ">
   <type>
      <simple type="string"/>
   </type>
</field>
```

For more information on DB FACADE storage, see About Microsoft Outlook and DB FACADE Storage on page 37.

4 Update the forms_12.xml file:

Use this file to customize the Microsoft Outlook forms. The syntax to manipulate these files is similar to the syntax of the HTML tables. You can modify the forms_12.xml file in much the same way.

a Add a new label control and a text field control for the Department field below the Job Title field on the Contact form.

To do this, insert a label, and a text field and add the XML code for the label to the section denoted by the comment "left side captions" inside the form with id = "OnDemand Contact". Insert the following XML code just after the cell containing the #lbl_job_title label control to add the new Department label. The text #lbl_department is used to specify a key that is used in the package_res.xml file to determine the localized value for the label, which is covered in the next step:

```
<cell size="22">
   <static id="lbl_department">
      <text>#lbl_department</text>
   </static>
</cell>
```

b After adding the label, add the text field control, using the following XML code:

```
<cell size="22">
   <edit id="department">
      <field value="string">Department</field>
   </edit>
</cell>
```

You must add this XML under the section labeled by the comment: left side fields. In this case, you want to add the field just above the cell stack containing the ContactToAccount and btn_mvgAccount controls and just below the cell stack containing the status_image.

c You must increase the size of the cell that houses all of these child objects to make more room for this new field on the UI by changing the size value from 210 to 235, for this example:

```
<cell size="235">
```

5 Update the package_res.xml file by adding the following element to the package_res.xml file to have the Contact form render the Department label through the associated key value:

```
<str key="lbl_department">Department</str>
```

The package_res.xml file is used to provide localized values and images to Oracle CRM On Demand Desktop. Because you have added the new Department field to the Contact form, you must provide the text for the label. In the modification of the forms_12.xml file in Step 4 on page 44, you created a label control with the text value #lbl_department. The text value identifies the key to use in the package_res.xml file.

Adding Check Boxes to a Form

This topic describes how to display a check box on an Oracle CRM On Demand Desktop form. In this example, you add a Never Email check box to the UI form.

In this procedure, you update the following metadata files:

- od_meta_info.xml
- od_basic_mapping.xml

- forms_12.xml
- package_res.xml

For information on the contents of these files, see About the Metadata Files Updated During Customization on page 33.

To add a check box to a form

1 Update the od_meta_info.xml file:

   a Locate the <object> element by TypeId, for example, for a contact it is similar to the following:

   ```
   <object TypeId='Contact' Label='#obj_contact' LabelPlural='#obj_contact_plural'>
   ```

   b Add a new <field> element to define the field, as follows:

   ```
   <field Name='NeverEmail' Label='NeverEmail' DataType='DTYPE_BOOL' />
   ```

   In this code, DataType is the type of the field. For check boxes or other Boolean fields, set DataType to DTYPE_BOOL. TypeID is the unique identifier of a type in the od_meta_info.xml file. When a type is defined, then the name is specified by the TypeId attribute.

2 Update the od_basic_mapping.xml file:

   a Locate the <object> element by the id attribute, for example, for a contact it is similar to the following:

   ```
   <type id="Contact" predefined_folder="10" ver="2">
   ```

   b Add a new <field> element to define the mapping as follows:

   ```
   <field id="NeverEmail">
       <reader>
          <mapi_user>
             <user_field id="od NeverEmail" ol_field_type="6"></user_field>
             <convertor>
                <bool/>
             </convertor>
          </mapi_user>
       </reader>
       <writer>
          <outlook_user>
             <user_field id="od NeverEmail" ol_field_type="6"></user_field>
             <convertor>
                <bool/>
             </convertor>
          </outlook_user>
       </writer>
   </field>
   ```

   The id attribute is the unique identifier of a type in the od_basic_mapping.xml file. When a type is defined, then the name is specified by the id attribute.

3 Update the forms_12.xml file by locating the object form, using the id attribute, and adding the code for check box control as follows:

```
<cell size="22">
    <checkbox id="NeverEmail" tab_order="33">
      <field>NeverEmail</field>
      <text>#lbl_never_email</text>
    </checkbox>
</cell>
```

where:

- <*field*> is the field name from the od_basic_mapping.xml file.
- <*text*> is the label resource id attribute from the package_res.xml file.

The id attribute is the unique identifier of a UI form of a type in the forms_XX.xml file. For example, a contact form definition is as follows:

```
<form id="OD Contact">
```

4 Update package_res.xml as follows:

```
<str key="lbl_never_email">Never Email</str>
```

Adding Currency Fields to a Form

This topic describes how to display a currency field, using a currency control on an Oracle CRM On Demand Desktop form. Although the currency control has *field* text in its name, it is a currency control.

```
<currency_field>CurrencyCode</currency_field>
```

In this example, you add an Annual Revenues currency field to the UI form. In this procedure, you update the following metadata files:

- od_meta_info.xml
- od_basic_mapping.xml
- forms_12.xml
- package_res.xml

For information on the contents of these files, see About the Metadata Files Updated During Customization on page 33.

To add a currency control field to a form

1 Update the od_meta_info.xml file as follows:

a Add the following two field definitions for the proper type where the currency code and the currency amount are stored:

```
<field Name="CurrencyCode" Label="CurrencyCode" DataType="DTYPE_TEXT"
HasPicklist="yes" PicklistTypeId='CurrencyPL' IsFilterable='yes'/>

<field Name="AnnualRevenues" Label="AnnualRevenues" DataType="DTYPE_NUMBER"
IsFilterable='yes'/>
```

b Add the type definition that requires these fields, for example, for Opportunity:

```
<object TypeId='Opportunity' Label='#obj_opportunity'...>
    <field Name="CurrencyCode".../>
    <field Name="AnnualRevenues".../>
    ...
</object>
```

2 Update the od_basic_mapping.xml file by adding the following XML section to the type definition that requires these fields:

```
<field id="CurrencyCode">
    <reader>
       <mapi_user>
          <user_field id="od CurrencyCode" ol_field_type="1"></user_field>
          <convertor>
             <string/>
          </convertor>
       </mapi_user>
    </reader>
    <writer>
       <outlook_user>
          <user_field id="od CurrencyCode" ol_field_type="1"></user_field>
          <convertor>
             <string/>
          </convertor>
       </outlook_user>
    </writer>
</field>

<field id="AnnualRevenues">
    <reader>
       <mapi_user>
          <user_field id="od AnnualRevenues" ol_field_type="3"></user_field>
          <convertor>
             <double/>
          </convertor>
       </mapi_user>
    </reader>
    <writer>
       <outlook_user>
          <user_field id="od AnnualRevenues" ol_field_type="3"></user_field>
       <convertor>
            <double/>
       </convertor>
       </outlook_user>
    </writer>
</field>
```

For example, for Opportunity, the structure is similar to the following:

```
<type id="Opportunity" display_name="#obj_opportunity_plural"...>
    <field id="CurrencyCode".../>
    <field id="AnnualRevenues".../>
    ...
</type>
```

3 Update the forms_12.xml file by adding the following XML sample code to update the UI layout by adding the label for the currency field:

```
<cell size="22">
    <multi_currency id="AnnualRevenues" tab_order="46">
       <currency_field>CurrencyCode</currency_field>
       <value_field>AnnualRevenues</value_field>
    </multi_currency>
</cell>
```

In this example, <currency_field> and <value_field> are the field names from the od_basic_mapping.xml file.

Add this code to the appropriate object type form, for example:

```
<form id="SBL Opportunity">
    ...
    <cell size="22">
    <multi_currency id="AnnualRevenues" tab_order="46">
    ...
</form>
```

The steps for adding labels are provided in "Changing the XML Code in the forms_12.xml File" on page 40 and are the same as for a text field.

4 Update the package_res.xml file by adding the resources for the label text.

The steps for adding the resources are provided in Changing the XML Code in the package_res.xml File on page 42 and are the same as for a text field.

Adding Date Fields to a Form

This topic describes how to display a date field on an Oracle CRM On Demand Desktop form. In this example, you add Due Date and End Time date and datetime fields to the UI form. In this procedure, you update the following metadata files:

- od_meta_info.xml
- od_basic_mapping.xml
- forms_12.xml
- package_res.xml

For information on the contents of these files, see About the Metadata Files Updated During Customization on page 33.

To add a date field to a form

1 Update the od_meta_info.xml file by adding the following XML elements to define the fields, to the corresponding object type structure:

- For a date only field type, add:

```
<field Name="DueDate" Label="DueDate" DataType="DTYPE_DATE" />
```

- For a datetime field type, add:

```
<field Name="EndTime" Label="EndTime" DataType="DTYPE_UTCDATETIME" />
```

2 Update the od_basic_mapping.xml file by adding the following XML code to the corresponding object type structure:

```
<field id="DueDate">
    <reader>
       <mapi_user>
          <user_field id="od DueDate" ol_field_type="5"></user_field>
          <convertor>
             <datetime/>
          </convertor>
       </mapi_user>
    </reader>
    <writer>
       <outlook_user>
          <user_field id="od DueDate" ol_field_type="5"></user_field>
          <convertor>
             <datetime/>
          </convertor>
       </outlook_user>
    </writer>
</field>
```

NOTE: For both the date and datetime field types, the mappings are equivalent, so only the XML for DueDate is shown.

3 Update the forms_12.xml file to change the layout by adding the label for the following fields:

- To add the date field, use the following XML sample code to the corresponding object type structure:

```
<cell>
    <datetime id="due" tab_order="13" type="date" store_time="false">
       <field>DueDate</field>
    </datetime>
</cell>
```

- To add the datetime field, use the following XML sample code to the corresponding object type structure:

```
<cell>
    <datetime id="end_time" tab_order="18" type="datetime">
       <field>EndTime</field>
    </datetime>
</cell>
```

In the sample XML code, the content of the <field> element is the field name from the od_basic_mapping.xml file.

The steps for adding labels are provided in "Changing the XML Code in the forms_12.xml File" on page 40 and are the same as for a text field.

4 Update the package_res.xml file by adding the resources for the label text.

The steps for adding the resources are provided in Changing the XML Code in the package_res.xml File on page 42 and are the same as for a text field.

Adding Description Fields to a Form

This topic describes how to display a description field, using the Microsoft Outlook control for description fields on an Oracle CRM On Demand Desktop form. In this example, you add a field called Description to the UI form. In this procedure, you update the following metadata files:

- od_meta_info.xml
- od_basic_mapping.xml
- forms_12.xml
- package_res.xml

For information on the contents of these files, see About the Metadata Files Updated During Customization on page 33.

To add a description control field to a form

1 Update the od_meta_info.xml file by adding the following XML element to define the field to the corresponding object type structure:

```
<field Name="Description" Label="Description" DataType="DTYPE_TEXT" />
```

2 Update the od_basic_mapping.xml file by adding the following XML code to the corresponding object type structure:

```
<field id="Description">
    <reader>
       <mapi_std>
          <mapi_tag id="0x10000000"></mapi_tag>
          <convertor>
             <multiline_string/>
          </convertor>
       </mapi_std>
    </reader>
    <writer>
```

```
            <outlook_std>
               <outlook_field id="Body"></outlook_field>
               <convertor>
                  <multiline_string/>
               </convertor>
            </outlook_std>
         </writer>
      </field>
```

3 Update the forms_12.xml file by adding the label for the description control field to update the UI layout.

To add the description control field, add the following XML to the corresponding object type structure:

```
<cell>
    <control id="description" tab_order="37" window_id="0x103f"></control>
</cell>
```

The steps for adding labels are provided in "Changing the XML Code in the forms_12.xml File" on page 40 and are the same as for a text field.

4 Update the package_res.xml file by adding the resources for the label text.

The steps for adding the resources are provided in Changing the XML Code in the package_res.xml File on page 42 and are the same as for a text field.

Adding Numeric Fields to a Form

This topic describes how to add a numeric field, using the edit control on an Oracle CRM On Demand Desktop form. In this example, you add two fields, Revenue and Number of Employees, to the UI form. In this procedure, you update the following metadata files:

- od_meta_info.xml
- od_basic_mapping.xml
- forms_12.xml
- package_res.xml

For information on the contents of these files, see About the Metadata Files Updated During Customization on page 33.

To add a numeric field to a form

1 Update the od_meta_info.xml file by adding the following XML elements to define the fields in the corresponding object type structure:

- For a fraction, use `DataType="DTYPE_NUMBER"`:

```
<field Name="Revenue" Label="Revenue" DataType="DTYPE_NUMBER" />
```

- For an integer, use `DataType="DTYPE_INTEGER"`:

```
<field Name="NumberEmployees" Label="NumberEmployees"
DataType="DTYPE_INTEGER" IsFilterable='no'/>
```

2 Update the od_basic_mapping.xml file by adding the following XML sections to the corresponding object type structure:

- For fractions, use the convertor, <double/>:

```
<field id="Revenue" ver="3">
   <reader>
      <mapi_user>
         <user_field id="od Revenue" ol_field_type="3"></user_field>
         <convertor>
            <double/>
         </convertor>
      </mapi_user>
   </reader>
   <writer>
      <outlook_user>
         <user_field id="od Revenue" ol_field_type="3"></user_field>
         <convertor>
            <double/>
         </convertor>
      </outlook_user>
   </writer>
</field>
```

- For integers, use the convertor, <integer/>:

```
<field id="NumberEmployees">
   <reader>
      <mapi_user>
         <user_field id="od NumberEmployees" ol_field_type="3"></user_field>
         <convertor>
            <integer/>
         </convertor>
      </mapi_user>
   </reader>
   <writer>
      <outlook_user>
         <user_field id="od NumberEmployees" ol_field_type="3"></user_field>
         <convertor>
            <integer/>
         </convertor>
      </outlook_user>
   </writer>
</field>
```

NOTE: If you are adding a field to the DB FACADE storage, then use the following syntax:

```
<field id="NumberEmployees">
    <type>
      <simple type="integer"/>
    </type>
</field>
```

3 Update the forms_12.xml file by adding the label for the numeric field to update the UI layout:

- For a fraction, use <field value="double">:

```
<edit id="Revenue" max_chars="15" tab_order="50">
   <field value="double" precision ="2">Revenue </field>
</edit>
```

  In this example, precision is the precision value; that is, the number of decimal places that is allowed after a comma.

- For an integer, use <field value="integer">:

```
<edit id="NumberEmployees" max_chars="15" tab_order="50">
   <field value="integer">NumberEmployees</field>
</edit>
```

  In this example, the content of the <field> element is the field name from the od_basic_mapping.xml file.

The steps for adding labels are provided in "Changing the XML Code in the forms_12.xml File" on page 40 and are the same as for a text field.

4 Update the package_res.xml file by adding the resources for the label text.

The steps for adding the resources are provided in Changing the XML Code in the package_res.xml File on page 42 and are the same as for a text field.

Adding a Picklist Field in Oracle CRM On Demand Desktop

This topic shows how to add a picklist field to the Contact object type.

Before You Begin

Review the chapter on customization in *Oracle CRM On Demand Desktop Administration Guide*.

In this procedure, you update the following metadata files:

- od_meta_info.xml
- od_basic_mapping.xml
- forms_12.xml

For information on the contents of these files, see About the Metadata Files Updated During Customization on page 33.

To add a picklist to the Contact object type

1 Define the objects and fields to synchronize with Oracle CRM On Demand as follows:

a Use an XML editor to open the od_meta_info.xml file.

b In the od_meta_info.xml file, locate the following element:

```
<object TypeId='Contact' ...>
```

The <object> element contains several child <field> elements. These child elements define the fields in the Contact object.

c Add the following <field> element as a child of the <object TypeId='Contact' ...> element:

```
<field Name="ContactType" Label="Contact Type" DataType="DTYPE_TEXT"
HasPicklist="yes" CRMName="Contact Type" />
```

In this example, ContactType is the exact field name that is used in SOAP queries.

d Save and close the od_meta_info.xml file.

2 Map the picklist field from the Contact object in the Oracle CRM On Demand database to a field in Oracle CRM On Demand Desktop as follows:

a Use an XML editor to open the od_basic_mapping.xml file.

b In the od_basic_mapping.xml file, locate the following element:

```
<type id="Contact" ...>
```

The <object> element contains several child <field> elements. These child elements define the fields in the Contact object.

c Add the following <field> element as a child element of the <object TypeId='Contact' ...> element:

```
<field id="ContactType">
  <reader>
    <mapi_user>
      <user_field id="od ContactType" ol_field_type="1"></user_field>
        <convertor>
          <string/>
        </convertor>
    </mapi_user>
  </reader>
  <writer>
    <outlook_user>
      <user_field id="od ContactType" ol_field_type="1"></user_field>
        <convertor>
          <string/>
        </convertor>
    </outlook_user>
  </writer>
</field>
```

d Create a new object type for the picklist.

To create a new object type for the picklist, you must use the following format for the value of the id attribute of the <type> element:

```
<type id="object_namefield_namePicklist">
```

where:

- *object_name* is the name of the object type in the od_meta_info.xml file.
- *field_name* is the name of the field in the object that you define in object_name.

In this example for the Contact Type field, add the following XML code inside the <database><types>…</types></database> elements:

```
<type id = "ContactContactTypePicklist" icon = "type_image:Generic:16">
 <field id = "Label">
    <type>
      <simple type = "string"/>
    </type>
  </field>
  <field id = "Value">
    <type>
      <simple type = "string"/>
    </type>
  </field>
  <field id = "SortOrder">
    <type>
      <simple type = "integer"/>
    </type>
  </field>
  <field id = "IsDefault">
    <type>
      <simple type = "boolean"/>
    </type>
  </field>
  <field id = "ParentCode">
    <type>
      <simple type = "string"/>
    </type>
  </field>
</type>
```

3 Customize the picklist control on the form layout by adding the following code in forms_12.xml in the vertical stack, as an example:

```
<cell size="22">
  <combobox id="ContactType" tab_order="16">
    <items format=":[:(Label):]" value_column="Value" has_null_item="false">
      <source type="auto" name="ContactContactTypePicklist"/>
        <order_by>
          <order ascend="true">SortOrder</order>
        </order_by>
    </items>
    <field>ContactType</field>
  </combobox>
</cell>
```

where:

- The *attribute* name of <source> is equal to the object type for the for the picklist form in Step d on page 60.
- <*field*> contains the name of the picklist field.

- The *has_null_item* attribute of <items> is set to false to prevent picking an empty value from the picklist.

For more information on a vertical stack, see About Vertical and Horizontal Stacks on page 62.

Adding a Cascading Picklist to Oracle CRM On Demand Desktop

Cascading picklists restrict the values of one picklist, the related picklist, based on the value selected in another picklist, the parent picklist. In this example, a parent picklist for opportunities presents a picklist called Status, and drives the value of a related picklist called ReasonWonLost.

Before You Begin

Ensure you have added both picklists to the od_meta_info.xml, od_basic_mapping.xml, and forms_12.xml metadata files. For more information, see "Adding a Picklist Field in Oracle CRM On Demand Desktop" on page 54. Also ensure that the forms_12.xml metadata file contains the following code:

```
<cell size="22">
  <combobox id="cbx_Status" tab_order="26
  <items format=":[:(Label):]" value_column="Value" has_null_item="true">
  <source type="auto" name="OpportunityStatusPicklist"/>
    <order_by>
      <order ascend="true">SortOrder</order>
    </order_by>
  </items>
    <field>Status</field>
  </combobox>
</cell>
<cell size="22">
  <combobox id="cbx_ReasonWonLost" tab_order="32">
  <items format=":[:(Label):]" value_column="Value" has_null_item="true">
  <source type="auto" name="OpportunityReasonWonLostPicklist"/>
    <order_by>
      <order ascend="true">SortOrder</order>
    </order_by>
  </items>
    <field>ReasonWonLost</field>
  </combobox>
</cell>
```

For information on the contents of the form_12.xml metadate file, see About the Metadata Files Updated During Customization on page 33.

To add a cascading picklist to Oracle CRM On Demand Desktop

1 In the forms.js metadata file, locate the Opportunity form handler function:

```
function od_opportunity_form(ctx)
```

2 Add the following code to the Opportunity form handler:

```
function Status_control_changed()
{
  var status = (form.cbx_Status.value != null) ? form.cbx_Status.value :
form.item.Status;
  update_picklistReasonWonLost_control(status);
}
```

The Opportunity form handler calls this function when the value of the parent picklist is changed. This function obtains the new value of the parent picklist and passes this value to the related picklist.

3 Add the following code to the Opportunity form handler:

```
function update_picklistReasonWonLost_control(parent_value)
{
  if (parent_value == null || parent_valueparent_value = "{C22EE487-1B92-4ba0-BC8D-
E1368E110FFE}";// this line is just to make sure that we created unique value for
the parent value and filtered current picklist correctly
  {
    parent_value = "{C22EE487-1B92-4ba0-BC8D-E1368E110FFE}";// this line is just to
make sure that we created unique value for the parent value and filtered current
picklist correctly
    form.cbx_ReasonWonLost.items.filter = session.create_expression("ParentCode",
"ne", parent_value);
    return;
  }
var pl_filter = ctx.session.create_criteria("or");
pl_filter.add(ctx.session.create_expression("ParentCode", "eq", parent_value));
pl_filter.add(ctx.session.create_expression("ParentCode", "eq", ""));
pl_filter.add(od_helpers.create_expression_or_null_value_filter(ctx.session,
"ParentCode", "eq", null));
form.cbx_ReasonWonLost.items.filter = pl_filter;
}
```

This function checks the value of the parent picklist. If the value of the parent picklist is empty, then all values in the related picklist are visible. If the parent picklist is populated, then the values in the related picklist are filtered, depending on the value in the parent picklist.

4 In the Opportunity form handler function, add the following code to the picklist you want to change:

```
ctx.events.connect(form.cbx_Status, "changed", Status_control_changed);
```

5 In the Opportunity form handler function, add the following code:

```
Status_control_changed();
```

This code enables the filtering of the parent picklist.

6 Save the forms.js metadata file.

Adding Multiselect Picklists to an Oracle CRM On Demand Desktop Form

This topic shows how to add multiselect picklists using an example. It shows you how to add the picklist into the Contact form in the UI. Use this example as a reference to add other multiselect picklist fields, and adjust them for your specific requirements.

Before You Begin

Review the chapter on customization in *Oracle CRM On Demand Desktop Administration Guide*.

The following procedure adds a multiselect picklist to the Contact form. Clicking the **...** button in the multiselect picklist launches a dialog box to select field values.

In this procedure, you update the following metadata files:

- od_meta_info.xml
- od_basic_mapping.xml
- forms_12.xml
- forms.js

For information on the contents of these files, see About the Metadata Files Updated During Customization on page 33.

To add a multiselect picklist to an Oracle CRM On Demand Desktop form

1. Define the objects and fields to synchronize with Oracle CRM On Demand as follows:
   a. Use an XML editor to open the od_meta_info.xml file.
   b. In the od_meta_info.xml file, locate the following element:

      ```
      <object TypeId='Contact' ...>
      ```

      The <object> element contains several child <field> elements. These child elements define the fields in the Contact object.
   c. Add the following <field> element as a child of the <object TypeId='Contact' ...> element:

      ```
      <field Name="msplTest_MSP" Label="msplTest_MSP" DataType="DTYPE_CSVLIST"
      HasPicklist="yes" IsFilterable='no'/>
      ```

      In this example, msplTest_MSP is the field name that is used in SOAP queries.
   d. Save and close the od_meta_info.xml file.
2. Map the multiselect picklist field from the Contact object in the Oracle CRM On Demand database to a field in Oracle CRM On Demand Desktop as follows:
   a. Use an XML editor to open the od_basic_mapping.xml file.
   b. In the od_basic_mapping.xml file, locate the following element:

```
<type id="Contact" ...>
```

The <object> element contains several child <field> elements. These children define the fields in the Contact object.

c Add the following <field> element as a child element of the <object TypeId='Contact' ...> element:

```
<field id="msplTest_MSP">
  <reader>
    <mapi_user>
      <user_field id="od msplTestMSP" ol_field_type="1">
      </user_field>
        <convertor>
          <multivalue_string/>
        </convertor>
    </mapi_user>
  </reader>
  <writer>
    <outlook_user>
      <user_field id="od msplTestMSP" ol_field_type="1">
      </user_field>
        <convertor>
          <multivalue_string/>
        </convertor>
    </outlook_user>
  </writer>
</field>
```

d Create a new object type for the picklist.

To create a new object type for the picklist, you must use the following format for the value of the id attribute of the <type> element:

```
<type id="object_namefield_namePicklist" predefined_folder="1">
```

where:

- *object_name* is the name of the object type in the od_meta_info.xml file.
- *field_name* is the name of the field that resides in the object that you define in object_name.

In this example, add the following XML code inside the <database><types>…</types></database> elements:

```
<type id="ContactmsplTest_MSPPicklist"
icon="type_image:ContactmsplTest_MSPPicklist:16">
   <field id ="Label">
      <type>
         <simple type="string"/>
      </type>
   </field>
   <field id ="Value">
      <type>
         <simple type="string"/>
```

```
            </type>
          </field>
          <field id ="SortOrder">
            <type>
              <simple type="integer"/>
            </type>
          </field>
          <field id ="IsDefault">
            <type>
              <simple type="boolean"/>
            </type>
          </field>
        </type>
```

3 Customize the multiselect picklist control on the form layout for the multiselect picklist by adding the following code in forms_12.xml in the vertical stack, as an example:

```
<cell size="21">
    <stack layout="horz">
      <cell>
        <scriptable_edit id="msp_edit" multiline="true">
          <field value="string"></field>
        </scriptable_edit>
      </cell>
      <cell size="5"></cell>
      <cell size="22">
        <button id="btn_msp_edit"><text>...</text>
        </button>
      </cell>
    </stack>
</cell>
```

where:

- <*scriptable_edit*> is a text box with values.
- <*button*> is the multiselect picklist button that displays the multiselect picklist dialog box.

The number of items displayed in the multiselect picklist control is determined by the value of the size attribute of the containing <cell> element.

For more information on a vertical stack, see About Vertical and Horizontal Stacks on page 62.

4 Customize a form handler in the forms.js file to set up a multiselect picklist layout control by adding the following JavaScript code to the Contact form handler:

```
var t_msp = new msp_ctrl(ctx, ctx.form.msp_edit, ctx.form.btn_msp_edit,
{"source_type": " ContactmsplTest_MSPPicklist ", "dest_field_name":
"msplTest_MSP"});
```

The msp_ctrl function accepts the following parameters, where:

- *ctx* is a global object that includes many other objects used to display the UI form and to control events, links, and so on.

- *ctx.form.msp_edit* is the <scriptable_edit> control object. The msp_edit portion must be an appropriate scriptable edit form control ID from the forms_*XX*.xml file. This control is defined using the <scriptable_edit id="…"> element that is specific to an example. This control object contains the reference to the scriptable edit control with the msp_edit identifier, defined in Step 3 on page 61. The ctx.form portion is a generic string that is left unchanged.
- *ctx.form.btn_msp_edit* is the <button> control object. The btn_msp_edit portion must be an appropriate button form control ID from the forms_*XX*.xml file; this control is defined using the <button id="…"> element that is specific to an example. This control object contains the reference to the button control with the btn_msp_edit identifier defined in Step 3 on page 61. The ctx.form portion is a generic string that is left unchanged.
- `{"source_type": "ContactmsplTest_MSPPicklist", "dest_field_name": "msplTest_MSP"}`

  This options object has the following properties, where:

  - *source_type* is the name of the type where picklist items are stored. This property is required.
  - *dest_field_name* is the name of the MSP field. This property is required.
  - *ui_delimiter* is the delimiter for selected items that is used on the form. The delimiter is a comma (,) by default. (Optional)
  - *max_selected* is the number of allowed items that can be selected. The default is 10. (Optional)

About Vertical and Horizontal Stacks

A stack is a group of controls that are placed inside the <stack> element vertically or horizontally depending on the type of stack. The structure for a vertical stack is as follows:

```
<stack layout="vertical">
  <cell><button...></button></cell>
  <cell><button...></button></cell>
</stack>
```

In this example, two buttons are placed vertically on the UI form. If you use a horizontal stack, then the buttons are placed horizontally on the UI form.

Setting Up Books for Oracle CRM On Demand Desktop Objects

Oracle CRM On Demand Desktop supports the book record visibility model. With this feature, you can configure Oracle CRM On Demand Desktop to allow users to synchronize records based on either a book, owner, or team visibility. The list of object types for which users can set up visibility modes is available through the Record Set tab of Oracle CRM On Demand Desktop Control Panel. By default, users can see the Record Set tab in the First Run Assistant wizard.

The following topics describe how to configure the various book-related features in the Oracle CRM On Demand Desktop customization package:

For additional book-related setup information, see Process of Adding Book Support to a New Top-Level Object on page 65

Setting Up Books for an Object Type

After you set up book support for a particular object type, that type becomes visible in the Record Set tab of the Oracle CRM On Demand Desktop Control Panel. This visibility allows users to choose and configure the visibility model that they want to use. If book support is not set, then the object type disappears from the Record Set tab, and the visibility for the type is set to the default visibility. The default visibility is owner and team visibility. If the following setting inside a type definition is set to true, then book visibility is set up. Otherwise, book visibility is not set up.

```
<access_mode_config BooksSupported="true" >...</access_mode_config>
```

To set up books for an object type, for example, Opportunity, complete the steps in the following procedure:

To set up books for an object type

1. Open the od_meta_info.xml file, and find the type definition by TypeId, in this example, Opportunity:

   ```
   <object TypeId="Opportunity" ...>
   ```

2. Inside the <object> element, locate the <access_mode_config> element.
3. Update the BooksSupported attribute of the <access_mode_config> element to TRUE.

   If you have set up the object element correctly, then the Opportunity type appears in the Record Set tab in the Synchronization Control Panel window.

Disabling Book Visibility for an Object Type

Complete the following procedure if you want to disable book visibility for an object type.

To disable book visibility for an object type

- Set the BooksSupported attribute of the <access_mode_config> element to FALSE for that object type.

  The object type disappears from the Record Set tab in the Synchronization Control Panel window.

Configuring the Include Sub-Items Option

Complete this procedure to configure an Include Sub-Items option for users in the Select Book window for object types that support books. Users can select this option when they are configuring a book in the Record Set tab in the Synchronization Control Panel window. By default, the Include Sub-Items option is not set up.

To configure the Include Sub-Items option

1. Edit the od_helpers.js file, available in the Oracle CRM On Demand Desktop customization package.
2. At the top of the file, locate the following definition of the `allow_includesubbooks` variable:

```
var allow_includesubbooks =
{
//"Account": true
}
```

3. Add type names within the braces with the value set to true or false, depending on whether you want to allow or disallow the option for the user.

   For example, if you uncomment the line in Step 2 for Account, the Include Sub-Items check box is displayed in the Select Book window for Account on the Record Set tab in the Synchronization Control Panel window.

Record Types That Support Books

Table 3 lists the Oracle CRM On Demand record types that support book functionality:

Table 3. Record Types That Support Books

Account	Exam	Opportunity
Accreditation	Financial Account	Partner
Allocation	Financial Plan	Policy
Application	Fund	Portfolio
Appointment	HCP Contact Allocation	Program
Business Plan	Household	Sample Lot
Campaign	Inventory Audit Report	Sample Transaction
Certification	Inventory Period	Service Request
Contact	Lead	Smart Call
Course	MDF Request	Solution
Custom Objects	MedEd Event	Special Pricing Request

Table 3. Record Types That Support Books

Deal Registration	Messaging Plan	Task
Dealer	Objective	Vehicle

For more information on Oracle CRM On Demand book support, see *Oracle CRM On Demand Online Help*

Process of Adding Book Support to a New Top-Level Object

When you add a new top-level object to Oracle CRM On Demand Desktop, perform the following procedures to set up book support for that object:

1. Configuring the Access Mode for the Object Type on page 65
2. Adding a BookId Field to an Object Type on page 66
3. Adding Book Selector Controls to Form UI on page 66
4. Prefilling the Default Book on New Records on page 70

Configuring the Access Mode for the Object Type

If you have added a new top-level object that supports books to the customization package, then you must configure book support in the od_meta_info.xml file. To view a list of top-level objects that support books refer to "Record Types That Support Books" on page 64.

This task is a step in Process of Adding Book Support to a New Top-Level Object on page 65.

To configure the access mode for the object type

1. Edit the od_meta_info.xml file, available in the Oracle CRM On Demand Desktop customization package.
2. Locate the type definition; that is, the <object> element, and add the following <access_mode_config> element, within the <object> element:

```
<object TypeId="TypeName">
 <access_mode_config BooksSupported="true" >
   <involved_fields>
     <field Name="Owner" />
     <field Name="BookName" />
   </involved_fields>
   <scopes>
     <scope Id="Dedup" BookId="" />
     <scope Id="ONLKP" BookId="" />
   </scopes>
```

```
  </access_mode_config>
  ...
</object>
```

You can update the BooksSupported attribute of the <access_mode_config> element to turn off book support at any time. For more information, see "Setting Up Books for an Object Type" on page 63.

3 Save the od_meta_info.xml file.

Adding a BookId Field to an Object Type

After you add a new object type that supports books, you must add a BookId field to the type definition in od_meta_info.xml and od_basic_mapping.xml. The type of field is string.

For guidelines on how to add a new field to an object, see Adding Custom Fields to Oracle CRM On Demand Desktop on page 34. To view a list of top-level objects that support books refer to "Record Types That Support Books" on page 64.

This task is a step in Process of Adding Book Support to a New Top-Level Object on page 65.

Adding Book Selector Controls to Form UI

The following topics describe how to add book selector controls to forms:

- Adding UI Controls for Books on page 66
- Registering the UI Controls in the Form Handler on page 68

To view a list of top-level objects that support books refer to "Record Types That Support Books" on page 64.

This task is a step in Process of Adding Book Support to a New Top-Level Object on page 65.

Adding UI Controls for Books

This procedure shows you how to add the UI controls to display a record's book on a form UI scriptable autocomplete control, which shows the books tree dialog that is used. This procedure shows how the control is set up on an Opportunity form.

The tree view of a book is displayed on the UI with a scriptable autocomplete control and a button control. These controls display a dialog box, showing the book's tree. The scriptable autocomplete control is an expanded version of an autocomplete control. Both autocomplete and scriptable autocomplete controls are used to display associated records, but scriptable autocomplete control can be implemented with additional logic for the value that the control displays. For more information on control elements used in the forms_XX.xml file, see Changing the XML Code in the forms_12.xml File on page 40.

To add a UI control for a book

1. Edit the forms_12.xml file, available in the Oracle CRM On Demand Desktop customization package.
2. Locate the record's Team control in forms_12.xml, add the Book UI control on the form UI underneath this control, and update head_opportunity_book_info for your new type label.

   In the following XML code:

   - The first cell, <cell size="13">...</cell>, contains two controls:
     - <static id="head_opportunity_book_info">, which is a label acting as a header.
     - <edge id="book_separator"/>, which is a horizontal ruler that underlines the header.
   - The second cell, <cell size="22">...</cell>, contains a static control label under the <!-- right side captions --> text and a scriptable autocomplete control under <!-- right side fields -->.

   Each control has a unique identifier (ID), which is used to refer to the control within the script. For example, for the button control <button id="btn_book_select">, the ID is btn_book_select. The value for all names that are prefixed with the number sign (#) as in the name #btn_book_select are specified in the package_res.xml file.

   The Book UI control XML code is as follows:

```
<cell size="13">
 <stack layout="horz">
    <cell size="110">
       <static id="head_opportunity_book_info">
          <text>#head_opportunity_book_info</text>
       </static>
    </cell>
    <cell>
       <stack layout="vert">
          <cell size="7"></cell>
          <cell size="1">
             <edge id="book_separator"/></cell>
       </stack>
    </cell>
 </stack>
</cell>
<cell size="22">
 <stack layout="horz" spacing="5" padding="5">
    <!-- right side captions -->
    <cell size="122">
       <stack spacing="5" layout="vert">
          <cell size="22">
             <static id="lbl_book_of_business">
                <text>#lbl_book_of_business</text>
             </static>
          </cell>
       </stack>
    </cell>
    <!-- right side fields -->
    <cell>
```

```
            <stack spacing="5" layout="vert">
               <cell size="22">
                  <stack layout="horz">
                     <cell>
                        <scriptable_edit id="book_of_business" tab_order="39">
                           <field value="string"></field>
                        </scriptable_edit>
                     </cell>
                     <cell size="5"></cell>
                     <cell size="22">
                        <button id="btn_book_select" tab_order="40">
                           <text>#btn_book_select</text>
                        </button>
                     </cell>
                  </stack>
               </cell>
            </stack>
         </cell>
      </stack>
   </cell>
```

NOTE: You must set the scriptable edit control Id and the book selector button control Id to btn_book_select. Otherwise, the UI controls will not function.

3 Save the forms_12.xml file.

Registering the UI Controls in the Form Handler

After the UI controls have been added to the forms_12.xml file, register the controls in the form handler by completing the following procedure.

To register the UI controls in the form handler

1 Edit the forms.js file, and locate the form handler function.

For example, for an Opportunity record, the function is od_opportunity_form().

2 Add the following function call inside the form handler function:

```
register_book_control(ctx, team_control_id);
```

In this function call, *team_control_id* is the Id of Team autocomplete_list control on a form. For example,:

3 Locate the Id of the autocomplete_list control that displays the record's team, for example:

```
<autocomplete_list id="TeamToOpportunity" tab_order="36">
```

4 For the above autocomplete_list control example Id, define the function register_book_control() as follows:

```
register_book_control(ctx, "TeamToOpportunity");
```

The team_control_id argument is not required if no team is associated with the object. For example, for an Opportunity object, the function looks like the following:

```
register_book_control(ctx, "TeamToOpportunity");
```

NOTE: Add the register_book_control() function call anywhere after the ctx.form_links_manager.init_new(); line. Otherwise, book prefilling will not work properly when a new record is created. For more information on book prefilling, see "To prefill the default book on new records" on page 70.

5 Save the forms.js file.

The form is populated with a control for the selected book name, and the button that brings up a dialog box displaying a book tree. Using this dialog box, a user can select different book.

About Book Prefilling

Prefilling is the process of automatically assigning predefined values to certain fields when a new record is created. Default values or functions that calculate default values must be linked to the fields that users want to have prefilled when they create a new record. This linking is specified in the business_logic.js file. In this example, the Book Name field is automatically prefilled with the value Book_12... after an Opportunity form is opened. For instructions to configure Oracle CRM On Demand Desktop so that when a user creates a new record, the default book is automatically added to the book for the field displaying the record, see "Setting Up Books for Oracle CRM On Demand Desktop Objects" on page 62.

The following line links the BookId field with the prefill_book function that returns the ID of the book to prefill the field.

```
scheme.objects.get_object("TypeName").get_field("BookId")["initial_value_fn"] =
prefill_book;
```

The algorithm implemented in the function is as follows:

1 Get the book that a user has selected on the Record Set tab on the Synchronization Control Panel.

2 If the user did not select a book, then get the book that is set as the default for the object type from the ::DefaultsTypeDefaultBookPicklist picklist. This picklist sets the default books for object types.

3 If the book is not available; that is, there is no default book for the record, then get the book that is set as the default for the current user regardless of the object type from the ::DefaultsBookPicklist picklist. This picklist sets the default book for users.

   If the default book for the user is not available, then do not return a value, and the Book Name field is not prefilled.

4 If the book ID is obtained from any of the preceding steps, then check whether the book can contain values or whether the book is only a container for other books. Return the book's ID if the book can contain values, or do not return any ID if the book is a container.

To set up default book prefilling, see Prefilling the Default Book on New Records on page 70.

Prefilling the Default Book on New Records

To set up default book prefilling on the form when a user creates a new record, complete the following procedure. For more information on book prefilling, see About Book Prefilling on page 69. To view a list of top-level objects that support books refer to Record Types That Support Books on page 64.

After you complete this procedure, a default book for the object type is populated on the form when a user creates a new item.

This task is a step in Process of Adding Book Support to a New Top-Level Object on page 65.

To prefill the default book on new records

1 Edit the business_logic.js file, and locate the create_ondemand_meta_scheme2() function.

2 Insert the following code anywhere inside that function:

```
scheme.objects.get_object("TypeName").get_field("BookId")["initial_value_fn"] =
prefill_book;
```

In this code, *TypeName* is the name of the object type for which you are setting up book prefilling, for example, Opportunity. After you add this code, a default book for the object type is populated on the form when a user creates a new item.

You can specify any top-level type name, such as Account, Contact, or Opportunity for *TypeName.* However, if you are defining book prefilling for an Activity type, then you must link the prefill_activity_book function and not the prefill_book function as follows:

```
scheme.objects.get_object("Activity").get_field("BookId")["initial_value_fn"] =
prefill_activity_book;
```

3 Add the prefill_book function and prefill_activity_owner functions inside the create_ondemand_meta_scheme2() function under the //Defaulting comment:

```
function prefill_book(ctx, type_id)
 {
    type_id = type_id || ctx.item_ex.get_type();
    var ownership_mode = od_helpers.get_ownership_mode(ctx, type_id);
    switch (ownership_mode)
    {
       case "book":
          var user_selected_book = od_helpers.get_ol_storage_rs_book(ctx,
type_id);
          var book_id = user_selected_book != null ? user_selected_book.BookId :
"";
          if (!book_id) {
             var default_book =
ctx.session.find_item("::DefaultsTypeDefaultBookPicklist",
ctx.session.create_expression("Value", "eq", type_id));
             book_id = default_book != null ? default_book.BookId : "";
          }
          if(!book_id)
          {
             var defaults = helpers.get_defaults(ctx.session),
                user_default_book_name = defaults != null ?
```

```
defaults.DefaultBookName : "",
                user_default_book =
ctx.session.find_item("::DefaultsBookPicklist",
ctx.session.create_expression("Label", "eq", user_default_book_name));
             book_id = user_default_book != null ? user_default_book.Value :
book_id;
          }
          if(od_helpers.book_can_contain_data(ctx, book_id))
             return book_id;
       break;
       default:
          return "";
       break;
    }
  }
  function prefill_activity_owner(ctx)
  {
     return prefill_owner(ctx, "Activity");
  }
```

4 Save the updated business_logic.js file.

.

4 Adding Custom Objects to Oracle CRM On Demand Desktop

This chapter describes how to add a custom object to Oracle CRM On Demand Desktop, using an example. It shows you how to add a Service Request custom object. Although Service Request is actually a default type, it is used as an example to better show the customization process using a default object using existing code in the default package. Use this example as a reference for adding other custom objects for your specific requirements. This chapter includes the following topics:

Adding Custom Objects to Oracle CRM On Demand Desktop

This procedure describes how to add the Service Request custom object to Oracle CRM On Demand Desktop. In this procedure, you make sure the custom object is available through Oracle CRM On Demand Web services, and then you update the following metadata files:

- od_meta_info.xml
- od_basic_mapping.xml
- connector_configuration.xml
- forms_12.xml
- business_logic.js
- forms.js
- package_res.xml
- views.xml

For information on the contents of these files, see About the Metadata Files Updated During Customization on page 33.

Before You Begin

Review the chapter on customization in *Oracle CRM On Demand Desktop Administration Guide*.

NOTE: This procedure adds the Service Request custom object to Microsoft Outlook storage and not DB FACADE storage.

For more information on Microsoft Outlook and DB FACADE storage, see About Microsoft Outlook and DB FACADE Storage on page 37.

To add a custom object to Oracle CRM On Demand Desktop

1 Make sure that the custom object is available through Oracle CRM On Demand Web services.

For information on using Web services, see *Oracle Web Services On Demand Guide*.

2 To add a custom object to Oracle CRM On Demand Desktop, you must update at least three files, od_meta_info.xml, od_basic_mapping.xml and connector_configuration.xml, as described in the following steps:

a Update the od_meta_info.xml file.

Add the custom object definitions to the od_meta_info.xml file. This file provides information about the user's environment to Oracle CRM On Demand Desktop, such as the objects available for Oracle CRM On Demand Desktop, the fields for the objects, and so on.

To add a custom object definition you must add an <object> element with a set of <field> elements that together define a set of fields that are available for Oracle CRM On Demand Desktop, as in the following sample outline:

```
<object ... >
   <field ... />
   <field ... />
   <field ... />
</object>
```

The <object> element can contain many different attributes, but the minimum required attributes are described in the following table.

Attribute Name	Attribute Value	Attribute Description
TypeId	Service Request	The unique identifier of this object type in Oracle CRM On Demand Desktop. This attribute can be any value that identifies this object in the Oracle CRM On Demand Desktop configuration. For child objects, use the following naming convention: *Parent_Name.Child_Name*
Label	Service Request	The label that Oracle CRM On Demand Desktop uses for this type of object in the user interface, in particular, for a filter view on a control panel.
LabelPlural	Service Requests	The plural label that Oracle CRM On Demand Desktop uses for this type of object in the user interface, in particular, for a filter view on a control panel.
ViewMode	Sales Rep	The view mode for this type of object. The view mode controls the object's visibility and data volume synchronized with Microsoft Outlook. This attribute can be set to any applicable view mode.
IntObjName	Service Request	The name of the integration object that provides access to the object you want to add to Microsoft Outlook.
XmlElemName	ServiceRequest	The name of the XML element that Oracle CRM On Demand Desktop uses in requests to the Oracle CRM On Demand server.

Attribute Name	Attribute Value	Attribute Description
XmlCollectionElemName	ListOfServiceRequest	The name of the XML collection element that Oracle CRM On Demand Desktop uses in requests to the Oracle CRM On Demand server.

To obtain the values for the object's IntObjName, XmlElemName and XmlCollectionElemName attributes, see *Oracle Web Services On Demand Guide*.

The Service Request object in Oracle CRM On Demand contains many fields, but this example uses only a few, as shown in the following sample XML for the Service Request custom object.

NOTE: The following example contains lines with inadvertent breaks due to their width. If you directly copy and paste this example, ensure that you fix any errors that result from these line breaks.

```
<object TypeId="Service Request" Label="Service Request" LabelPlural="Service
Requests" ViewMode="Sales Rep" IntObjName="Service Request"
XmlElemName="ServiceRequest" XmlCollectionElemName="ListOfServiceRequest">
    <open_with_url_tmpl>
       <![CDATA[
       :[:(protocol):]://:[:(hostname):]::[:(port):]/OnDemand/user/
       AccountDetail?ServiceRequestDetailForm.Id=:[:(own_id):]
       ]]>
    </open_with_url_tmpl>

    <extra_command_options>
     <option Name="UseDefaultViewMode" Value="true" Scopes="Dedup;ONLKP;QBID"/>
    </extra_command_options>

    <viewmodes>
       <viewmode Name="None" Scopes="Dedup;ONLKP;QBID" />
    </viewmodes>

    <access_mode_config BooksSupported="true" TypeOrderNumber="6">
       <involved_fields>
          <field Name="Owner" />
          <field Name="BookName" />
       </involved_fields>
       <scopes>
          <scope Id="Dedup" BookId="" />
          <scope Id="ONLKP" BookId="" />
          <scope Id="QBID" BookId="" />
       </scopes>
    </access_mode_config>
    <field Name="SRNumber" Label="SRNumber" DataType="DTYPE_TEXT"
CRMName="SRNumber" BackUpd="all">
       <suppress_on Upstream="true"/>
    </field>
    <field Name="Subject" Label="Subject" DataType="DTYPE_TEXT"
```

```
CRMName="Subject"/>
    <field Name="Description" Label="Description" DataType="DTYPE_TEXT"
CRMName="Description"/>

    <field Name="Id" Label="Id" DataType="DTYPE_ID" IsPrimaryKey="yes"
IsFilterable='no'/>
    <field Name="ModId" Label="ModId" DataType="DTYPE_INTEGER" IsTimestamp="yes"
IsFilterable='no' IsHidden="yes"/>
    <field Name="OwnerId" Label="OwnerId" DataType="DTYPE_ID" IsRefObjId="yes"
RefObjTypeId="User" IsFilterable='no'/>
 </object>
```

You must also add the BackUpd="all" attribute on the SRNumber field because its value should be generated by server and must be synchronized from the Oracle CRM On Demand server side while adding an object on the server. For example:

```
<field Name="SRNumber" Label="SRNumber" DataType="DTYPE_TEXT" CRMName="SRNumber"
BackUpd="all">
```

b Update the od_basic_mapping.xml file.

Add Service Request type to the od_basic_mapping.xml file, which describes the data structure that must be created in Microsoft Outlook by Oracle CRM On Demand Desktop. There are two types of storage that exist in Oracle CRM On Demand Desktop:

- Microsoft Outlook
- DB FACADE

For more information on Microsoft Outlook and DB FACADE storage, see About Microsoft Outlook and DB FACADE Storage on page 37.

NOTE: If an object does not have to be displayed on the Microsoft Outlook folder list view, then you must add this type of object to DB FACADE storage. Objects added to DB FACADE storage cannot be displayed on Outlook forms. Instead, dialog boxes are used to represent the data for those objects.

The syntax for adding an object to storage depends on what the type of storage to which you are adding the object. For adding an object to Outlook storage, the syntax is as follows:

```
<type ... >
   <form ... />
   <field  ... />
   <field  ... />
   <field  ... />
</type>
```

Make the following updates to the od_basic_mapping.xml file that results in the final XML code for the Service Request type in Appendix A, “XML Code for Service Request Types”:

- You must define the attributes of the <type> element described in the following table in the od_basic_mapping.xml file.

Attribute Name	Attribute Value	Description
id	Service Request	This attribute defines the unique ID of this object type in Oracle CRM On Demand Desktop. This value must be the same as the value specified for the TypeId attribute of the <object> element in od_meta_info.xml in Step 1 on page 74.
display_name	Service Requests	This attribute defines the label that Oracle CRM On Demand Desktop uses for this type of object in the user interface. In this example, the label defines the name of a folder in Microsoft Outlook storage, where objects of this type are stored. This value must be unique among all types in the od_basic_mapping.xml file.
folder_type	10	This attribute specifies that objects of this type are stored in a folder of the same type as Microsoft Outlook Contacts.

- Add the <form> element which is required in the definition of the new object type. This element defines some important UI attributes, such as form definition used to display records of this type, the icon for this type, the display name for the form, and other attributes. In the example, define the attributes described in the following table.

Attribute Name	Attribute Value	Attribute Description
message_class	IPM.Contact.SBL. ServiceRequest	This attribute defines the type for new objects by using a subclass of the Microsoft Outlook message class. The new object is based on the Contact object, but you can extend the set of fields for this type. Use the following recommended naming convention for this attribute value: IPM.Contact.SBL.*Object_type*
icon	type_image: Service_Request:16	This attribute defines the icon that is displayed for this type of object, for example, on a form caption in Microsoft Outlook views, and so on. The value of this attribute is the resource key of an image file, for example, a PNG file, 16 x 16 pixels, to be used and defined in any Oracle CRM On Demand Desktop resource file.
large_icon	type_image: Service_Request:32	This attribute defines the icon to be displayed for this type of object on a specific Microsoft Outlook view (icon type). The value of this attribute is the resource key of an image file, for example, a PNG file, 32 x 32 pixels, to be used and defined in any Oracle CRM On Demand Desktop resource file.
display_name	Service Request	This attribute defines the caption of the Microsoft Outlook form if the object is displayed on the Microsoft Outlook form.

The <form> element must also contain the name of the custom form definition as defined in the forms_12.xml file. For this example, leave it as `OnDemand ServiceRequest`, see the following sample XML:

```
<form message_class="IPM.Contact.SBL.ServiceRequest"
display_name="#obj_service_request" icon="type_image:Service_Request:16"
large_icon="type_image:Service_Request:32">OnDemand ServiceRequest</form>
```

- Define the fields of the new object. The field mapping definitions depend on the field type and, in some cases, the decision to map the field to either a custom field or to one of Microsoft Outlook fields. For more information, see Chapter 3, "Adding Custom Fields to Oracle CRM On Demand Desktop."

  In this example, you map the following fields of Service Request object:

  SRNumber
  OwnerId
  Subject
  Description

  You also map the following Service Request fields that you might decide to use later:

  ObjectState
  SuppressFileAs

The final XML code for the Service Request type in the example is provided in "XML Code for the Service Request Type Example" on page 91.

c Update the forms_12.xml file.

To be able to see custom object fields values, views, linked objects, and so on, you must define the custom object form layout in the forms_12.xml file. The outline of the the custom object form layout is in the example provided in "XML Code for the Custom Objects Form Layout Example" on page 94.

d Update the business_logic.js file.

To avoid receiving a Microsoft Outlook dialog with the message that field File As was not filled, you need to add the following code in function `create_ondemand_meta_scheme2` , before `// Activity processing and related set`:

```
scheme.objects.get_object("Service
Request").get_field("SuppressFileAs")["initial_value_res"] =
"lang_general_initial_fileas";
```

e Update the forms.js file.

Customer objects requires their own handler. This handler should be specified in the forms.js file by adding in the following function:

```
function od_sr_form(ctx)

{
 if (!od_helpers.check_first_sync(ctx))
 {
    od_helpers.lock_form_before_first_sync(ctx)
    return;
 }

 ctx.form_links_manager.init_new();
```

```
 var form = ctx.form;
 form.SRNumber.enabled = false;
}
```

For the Service Request custom object, the following line disables the SRNumber field because its value is generated by the server:

```
"form.SRNumber.enabled = false;"
```

Consequently, you must specify the following handler in object form in the script section in the forms_12.xml file:

```
var current_form = new
include.forms.od_sr_form(include.forms.create_form_ctx(ctx));
```

f Update the connector_configuration.xml file.

The changes made in the od_meta_info.xml and od_basic_mapping.xml describe only the new object in a data structure, either in Microsoft Outlook or on Oracle CRM On Demand. However, to synchronize items of this object type between Oracle CRM On Demand and Microsoft Outlook, you must add the new object definition to a connection_configuration.xml file.

To add the object definition, you add the following structure to the <types> element in the connection_configuration.xml file:

```
<type ... >
   <view ... />
   <synchronizer ... />
      <links>
         <link ... />
         ...
      </links>
      <natural_keys>
         <natural_key>
            <field ... />
            ...
         </natural_key>
         ...
      </natural_keys>
   </synchronizer>
</type>
```

After completing adding the object definition for this example, the resulting XML code is as follows:

```
<type id="Service Request" state_field="ObjectState">
   <view label="Service Request" label_plural="Service Requests"
small_icon="type_image:Service_Request:16"
normal_icon="type_image:Service_Request:24"
large_icon="type_image:Service_Request:48"></view>
      <synchronizer name_format=":[:(SRNumber):]" threshold="3">
         <links>
            <link>AccountId</link>
            <link>ContactId</link>
```

```
            <link>OwnerId</link>
         </links>
         <natural_keys>
            <natural_key>
               <field>Subject</field>
            </natural_key>
         </natural_keys>
      </synchronizer>
</type>
```

- Define the attribute on the <type> element as described in the following table.

Attribute Name	Attribute Value	Description
id	Service Request	The unique ID of this object type in Oracle CRM On Demand Desktop. Set this value to the same value specified for the TypeId attribute of the <object> element in od_meta_info.xml and for the id attribute of the <type> element in the od_basic_mapping.xml file.

- Define the attributes on the <view> element as described in the following table. The view element defines some of the user interface settings.

Attribute Name	Attribute Value	Attribute Description
label	Service Request	This attribute defines the label that Oracle CRM On Demand Desktop uses for this type of object in the user interface. In particular, Oracle CRM On Demand Desktop uses this label for synchronization issues to resolve duplicates, resolve conflicts, and synchronize the confirmation tabs of the Control Panel.
label_plural	Service Requests	This attribute defines the plural label that Oracle CRM On Demand Desktop uses for this type of object in the user interface. In particular, Oracle CRM On Demand Desktop uses this label for synchronization issues to resolve duplicates, resolve conflicts, and synchronize the confirmation tabs of the Control Panel.

Attribute Name	Attribute Value	Attribute Description
small_icon	type_image:Service_Request:16	This attribute defines the icon to be displayed for this type of object on a Control Panel as a child element icon on a filter tree in various lists of the Control Panel. The value of this attribute is a resource key of an image file, for example, a PNG file, 16 x 16 pixels, to be used and defined in any Oracle CRM On Demand Desktop resource file.
normal_icon	type_image:Service_Request:24	This attribute defines the icon that is displayed for this type of object on a Control Panel, as the top-level element icon on a filter tree. Set the value of this attribute as a resource key of an image file, for example, a PNG file, 24 x 24 pixels, to use and define in any Oracle CRM On Demand Desktop resource file.
large_icon	type_image:Service_Request:48	This attribute defines the icon that is displayed for this type of object, The icon is on a synchronization dialog box. Set the value of this attribute as a resource key of an image file, for example, a PNG file, 48 x 48 pixels, to use and define in any Oracle CRM On Demand Desktop resource file.

- Define the attributes on the <synchronizer> element as described in the following table. The <synchronizer> element defines the synchronization settings for the object, such as the link fields, natural keys, how the object is displayed if it appears in any Microsoft Windows Control Panel list, and so on.

  .

Attribute Name	Attribute Value	Attribute Description
name_format	:[:(SRNumber):]	This attribute's value defines a mask that is used to build the object identifier in any list of the Microsoft Windows Control Panel (synchronization issues, duplicates, and so on). In this example, the Control Panel displays the value of the Name field.

  The <synchronizer> element also contains the <links> and <natural_keys> elements.

  The <links> element defines a set of link fields for the object. Oracle CRM On Demand Desktop requires these fields to build an object's dependency tree during synchronization. Oracle CRM On Demand Desktop must also determine which fields contain Ids of other objects, so that their values can be converted from Oracle CRM On Demand Id format to Oracle CRM On Demand Desktop Id format and from Oracle CRM On Demand Desktop Id format to Oracle CRM On Demand format.

- The <natural_keys> element contains a set of field sets denoted by <natural_key>, which is used to identify duplicated records during synchronization. Records are identified as a duplicate if there are two matching records in Oracle CRM On Demand and Oracle CRM On Demand Desktop, and if the match is defined by the fields that are specified in natural keys. In the example, the field is: Subject.

g Update the package_res.xml file.

The package_res.xml file contains descriptions of every resource in package, for example, labels, images, and so on. Any customer object also requires resources to be defined, as follows:

- For the Service Request object, add the following label of list view to resources:

```
<str key="view_for_service_requests">CRM On Demand Service Requests</str>
```

- Define the Service Request object icon:

```
<img key="type_image:ServiceRequest:16">IMAGE_IN_BASE64</img>
<img key="type_image:ServiceRequest:24">IMAGE_IN_BASE64</img>
<img key="type_image:ServiceRequest:32">IMAGE_IN_BASE64</img>
<img key="type_image:ServiceRequest:48">IMAGE_IN_BASE64</img>
```

NOTE: Replace IMAGE_IN_BASE64 text with valid image in Base64 encoding.

h Update the views.xml file.

To expose object field values on a list view, you must define the object custom view with the corresponding columns. You can apply the custom view to the folder by adding its reference to the custom_views tag in the basic mapping.

The following is an example of a custom view.

NOTE: The following example contains lines with inadvertent breaks due to their width. If you directly copy and paste this example, ensure that you fix any errors that result from these line breaks.

```
 <str key="all_sr">
 <![CDATA[<?xml version="1.0"?>
 <view type="table">
 <viewname>$view_for_service_requests$</viewname>
 <viewstyle>table-layout:fixed;width:100%;font-family:Segoe UI;font-
style:normal;font-weight:normal;font-size:8pt;color:Black;font-charset:0</
viewstyle>
 <viewtime>214524766</viewtime>
 <linecolor>8421504</linecolor>
 <linestyle>3</linestyle>
 <gridlines>1</gridlines>
 <autosizing>0</autosizing>
 <newitemrow>0</newitemrow>
 <incelledit>0</incelledit>
 <usequickflags>0</usequickflags>
 <collapsestate/>
 <rowstyle>background-color:window;color:windowtext</rowstyle>
 <headerstyle>background-color:#D3D3D3</headerstyle>
 <previewstyle>color:Blue</previewstyle>
 <filter>&quot;http://schemas.microsoft.com/exchange/extensionattribute1&quot;
&lt;&gt; 'true'</filter>
 <arrangement>
    <autogroup>0</autogroup>
    <collapseclient/>
 </arrangement>
 <column>
    <name>HREF</name>
    <prop>DAV:href</prop>
    <checkbox>1</checkbox>
 </column>
 <column>
    <maxrows>1701576704</maxrows>
    <editable>0</editable>
    <heading>Icon</heading>
    <prop>http://schemas.microsoft.com/mapi/proptag/0x0fff0102</prop>
    <bitmap>1</bitmap>
    <width>50</width>
    <style>padding-left:3px;;text-align:center</style>
 </column>
 <column>
    <heading>Service Request Number</heading>
    <prop>http://schemas.microsoft.com/mapi/string/{00020329-0000-0000-C000-
000000000046}/od%20SRNumber</prop>
```

```
    <type>string</type>
    <width>200</width>
    <style>padding-left:3px;;text-align:left</style>
    <editable>1</editable>
    <userheading>Service Request Number</userheading>
 </column>
 <column>
    <heading>Service Request Subject</heading>
    <prop>http://schemas.microsoft.com/mapi/string/{00020329-0000-0000-C000-
000000000046}/od%20Subject</prop>
    <type>string</type>
    <width>200</width>
    <style>padding-left:3px;;text-align:left</style>
    <editable>1</editable>
    <userheading>Service Request Subject</userheading>
 </column>
 <column>
    <heading>Service Request Description</heading>
    <prop>http://schemas.microsoft.com/mapi/string/{00020329-0000-0000-C000-
000000000046}/od%20Description</prop>
    <type>string</type>
    <width>200</width>
    <style>padding-left:3px;;text-align:left</style>
    <editable>1</editable>
    <userheading>Service Request Description</userheading>
 </column>
 </view>]]>
 </str
```

In the preceding custom view:

- `<str key="all_sr">` is the view Id defined in the basic_mapping.xml file.
- `<prop>http://schemas.microsoft.com/mapi/string/{00020329-0000-0000-C000-000000000046}/od%20SRNumber</prop>` is the custom object field "od SRNumber" property value

Adding Custom Objects to DB FACADE Storage

If you are adding an object to DB FACADE storage, then the syntax to add the object is simpler than the syntax for Microsoft Outlook storage because you do not require the <form> element, and the number of <field> child elements is less as shown in the following template.

For more information on Microsoft Outlook and DB FACADE storage, see About Microsoft Outlook and DB FACADE Storage on page 37.

Before You Begin

Review the chapter on customization in *Oracle CRM On Demand Desktop Administration Guide*.

Example Template for Adding an Object to DB FACADE Storage

See the following template:

```
<type ... >
     <field  ... />
     <field  ... />
     <field  ... />
</type>
```

Completing this procedure results in the following XML code in the od_basic_mapping.xml file:

```
<type id="Internal_Product" icon="type_image:Service Request:16">
  <field id ="SRNumber">
     <type>
        <simple type="string"/>
     </type>
  </field>
  <field id="AccountId">
     <type>
        <simple type="binary"/>
     </type>
  </field>
  <field id="AccountName">
     <type>
        <simple type="string"/>
     </type>
  </field>
     <field id="ContactId">
     <type>
        <simple type="binary"/>
     </type>
  </field>
  <field id="Area">
     <type>
        <simple type="string"/>
     </type>
  </field>
  <field id="Cause">
     <type>
        <simple type="string"/>
     </type>
  </field>
  <field id="Type">
     <type>
        <simple type="string"/>
     </type>
  </field>
  <field id="Source">
     <type>
        <simple type="string"/>
     </type>
  </field>
  <field id="CurrencyCode">
     <type>
```

```
            <simple type="string"/>
         </type>
      </field>
      <field id="Priority">
         <type>
            <simple type="string"/>
         </type>
      </field>
      <field id="Status">
         <type>
            <simple type="string"/>
         </type>
      </field>
      <field id="OwnerId">
         <type>
            <simple type="binary"/>
         </type>
      </field>
      <field id="Subject">
         <type>
            <simple type="string"/>
         </type></field>
      <field id="StatusContact">
         <type>
            <simple type="string"/>
         </type>
      </field>
      <field id="StatusAccount">
         <type>
            <simple type="string"/>
         </type>
      </field>
   </type>
```

NOTE: DB FACADE storage objects cannot have <multiwriter> functionality, which is why some fields are omitted in the sample code. Objects stored in DB FACADE storage do not require the ObjectState and SuppressFileAs fields.

To add a custom object to DB FACADE storage

1 Define the attributes of the <type> element in the od_basic_mapping.xml file as described in the following table.

Attribute Name	Attribute Value	Attribute Description
id	Service Request	The unique ID of this object type in Oracle CRM On Demand Desktop. Set this attribute to the same value specified for the TypeId attribute of the <object> element in od_meta_info.xml. **NOTE:** The name you assign to an object in od_meta_info.xml must be used for the object's name in the od_basic_mapping.xml file.
icon	type_image:Service Request:16	This attribute defines the icon that is displayed for this type of object, for example, on a form caption, in Microsoft Outlook views, and so on. Set the value of this attribute to a resource key of an image file, for example, a PNG file, 16 x 16 pixels, to be used and defined in any Oracle CRM On Demand Desktop resource file.

2 Define the attribute of the <field> element in od_basic_mapping.xml as described in the following table.

Attribute Name	Attribute Value	Attribute Description
id	Service Request	The unique ID of this field within an object type.

3 Define the attribute of the <simple> element in the od_basic_mapping.xml file as described in the following table.

Attribute Name	Attribute Value	Attribute Description
type	■ string ■ binary ■ integer ■ double ■ foreign_key ■ boolean	The unique ID of this field within an object type.

A XML Code for Service Request Types

This appendix contains one topic: XML Code for the Service Request Type Example

The topic lists the final XML code for the Service Request type example in Adding Custom Objects to Oracle CRM On Demand Desktop on page 73.

NOTE: The following examples in this appendix contain lines with inadvertent breaks due to their width. If you directly copy and paste these examples, ensure that you fix any errors that result from these line breaks.

XML Code for the Service Request Type Example

The XML code is as follows:

```
<type id="Service Request" folder_type="10" display_name="Service Requests">
        <form message_class="IPM.Contact.SBL.ServiceRequest" display_name="Service
Request" icon="type_image:ServiceRequest:16"
large_icon="type_image:ServiceRequest:32">OnDemand ServiceRequest</form>
        <custom_views default_name="#view_for_service_requests">
           <view id="all_sr" name="#view_for_service_requests"/>
        </custom_views>
        <field id="SRNumber">
           <reader>
              <mapi_user>
                 <user_field id="od SRNumber" ol_field_type="1"/>
                 <convertor>
                    <string/>
                 </convertor>
              </mapi_user>
           </reader>
           <writer>
              <outlook_user>
                 <user_field id="od SRNumber" ol_field_type="1"/>
                 <convertor>
                    <string/>
                 </convertor>
              </outlook_user>
           </writer>
        </field>
        <field id="Subject">
           <reader>
              <mapi_user>
                 <user_field id="od Subject" ol_field_type="1"/>
                 <convertor>
                    <string/>
```

```
            </convertor>
         </mapi_user>
      </reader>
      <writer>
         <outlook_user>
            <user_field id="od Subject" ol_field_type="1"/>
            <convertor>
               <string/>
            </convertor>
         </outlook_user>
      </writer>
   </field>
   <field id="OwnerId" ver="2">
      <reader>
         <mapi_user>
            <user_field id="od OwnerId" ol_field_type="1"/>
            <convertor>
               <binary_hexstring/>
            </convertor>
         </mapi_user>
      </reader>
      <writer>
         <multiwriter>
            <outlook_user>
               <user_field id="od OwnerId" ol_field_type="1"/>
               <convertor>
                  <binary_hexstring/>
               </convertor>
            </outlook_user>
            <link_fields>
               <field from="Alias" to="Owner"/>
            </link_fields>
         </multiwriter>
      </writer>
   </field>
   <field id="ObjectState">
      <reader>
         <mapi_user>
            <user_field id="od ObjectState" ol_field_type="3"/>
            <convertor>
               <integer/>
            </convertor>
         </mapi_user>
      </reader>
      <writer>
         <multiwriter>
            <outlook_user>
               <user_field id="od ObjectState" ol_field_type="3"/>
               <convertor>
                  <integer/>
               </convertor>
            </outlook_user>
            <outlook_std>
               <outlook_field id="User1"/>
```

```
            <convertor>
               <bitmask2string>
                  <rule mask="134217728" result="134217728" value=""/>
                  <rule mask="1" result="1" value="true"/>
                  <rule mask="1073741824" result="1073741824" value="true"/>
               </bitmask2string>
            </convertor>
         </outlook_std>
         <outlook_user>
            <user_field id="od IndirectlyVisible" ol_field_type="6"/>
            <convertor>
               <bitmask2bool>
                  <condition mask="1" result="1" eq="true"/>
               </bitmask2bool>
            </convertor>
         </outlook_user>
      </multiwriter>
   </writer>
</field>
<field id="Description">
   <reader>
      <mapi_user>
         <user_field id="od Description" ol_field_type="1"/>
         <convertor>
            <string/>
         </convertor>
      </mapi_user>
   </reader>
   <writer>
      <outlook_user>
         <user_field id="od Description" ol_field_type="1"/>
         <convertor>
            <string/>
         </convertor>
      </outlook_user>
   </writer>
</field>
<field id="Owner">
   <reader>
      <mapi_user>
         <user_field id="od Owner" ol_field_type="1"/>
         <convertor>
            <string/>
         </convertor>
      </mapi_user>
   </reader>
   <writer>
      <outlook_user>
         <user_field id="od Owner" ol_field_type="1"/>
         <convertor>
            <string/>
         </convertor>
      </outlook_user>
   </writer>
```

```
        </field>
        <field id="SuppressFileAs">
           <reader>
              <mapi_std>
                 <mapi_tag id="0x3A16001F"/>
                 <convertor>
                    <string/>
                 </convertor>
              </mapi_std>
           </reader>
           <writer>
              <outlook_std>
                 <outlook_field id="CompanyName"/>
                 <convertor>
                    <string/>
                 </convertor>
              </outlook_std>
           </writer>
        </field>
     </type>
```

XML Code for the Custom Objects Form Layout Example

The XML code is as follows:

```
<form id="OnDemand ServiceRequest">
   <script>
      include("forms.js", "forms");
      var ctx = {
         "application": application,
         "ui": application.ui,
         "application_script": application_script,
         "form": form
      };
      var current_form = new
include.forms.od_sr_form(include.forms.create_form_ctx(ctx));
   </script>
   <page id="General" min_height="800" min_width="900">
      <cell>
         <stack layout="horz" padding="5">
            <cell>
               <stack layout="vert" padding="5">
                  <cell size="5"/>
                     <cell size="150">
                        <stack layout="horz" spacing="2">
                           <cell>
                              <stack layout="horz" spacing="5">
                                 <cell size="110">
                                    <stack spacing="5" layout="vert" padding="4">
                                       <cell size="22">
                                          <static id="lbl_srnumber" tab_order="1">
```

```
                                    <text>SR Number</text>
                                 </static>
                              </cell>
                              <cell size="22">
                                 <static id="lbl_srsubject" tab_order="8">
                                    <text>Subject</text>
                                 </static>
                              </cell>
                              <cell size="35">
                                 <static id="lbl_srdescription" tab_order="10">
                                    <text>Description</text>
                                 </static>
                              </cell>
                           </stack>
                        </cell>
                        <cell>
                           <stack layout="vert" spacing="5">
                              <cell size="22">
                                 <edit id="SRNumber" max_chars="250"
tab_order="12" multiline="true">
                                    <field value="string">SRNumber</field>
                                 </edit>
                              </cell>
                              <cell size="22">
                                 <edit id="Subject" max_chars="250" tab_order="13"
multiline="true">
                                    <field value="string">Subject</field>
                                 </edit>
                              </cell>
                              <cell size="22">
                                 <edit id="Description" max_chars="250"
tab_order="14" multiline="true">
                                    <field value="string">Description</field>
                                 </edit>
                              </cell>
                           </stack>
                        </cell>
                     </stack>
                  </cell>
               </stack>
            </cell>
            <!-- hidden section -->
            <cell size="0">
               <stack layout="vert">
                  <cell>
                     <control id="btn Full Name" window_id="0x6f2a"/>
                  </cell>
                  <cell>
                     <control id="lbl Job title" window_id="0x11a3"/>
                  </cell>
                  <cell>
                     <control id="lbl Company" window_id="0x11a2"/>
                  </cell>
                  <cell>
```

```
      <control id="lbl File as" window_id="0x11a4"/>
   </cell>
   <cell>
      <control id="edit Full Name" window_id="0x1000"/>
   </cell>
   <cell>
      <control id="edit Job title" window_id="0x1180"/>
   </cell>
   <cell>
      <control id="edit Company" window_id="0x1181"/>
   </cell>
   <cell>
      <control id="cb File as" window_id="0x1182"/>
   </cell>
   <cell>
      <control id="Image" window_id="0x1108"/>
   </cell>
   <cell>
      <control id="lbl Internet" window_id="0x11bb"/>
   </cell>
   <cell>
      <control id="edge Internet" window_id="0x11c5"/>
   </cell>
   <cell>
      <control id="btn E-mail" window_id="0x1101"/>
   </cell>
   <cell>
      <control id="dd E-mail" window_id="0x111b"/>
   </cell>
   <cell>
      <control id="lbl Display as" window_id="0x11c6"/>
   </cell>
   <cell>
      <control id="lbl Web page" window_id="0x11aa"/>
   </cell>
   <cell>
      <control id="lbl IM" window_id="0x11ba"/>
   </cell>
   <cell>
      <control id="edit E-mail" window_id="0x1018"/>
   </cell>
   <cell>
      <control id="edit Display as" window_id="0x101c"/>
   </cell>
   <cell>
      <control id="edit Web" window_id="0x11a9"/>
   </cell>
   <cell>
      <control id="edit Im address" window_id="0x1016"/>
   </cell>
   <cell>
      <control id="lbl Phone numbers" window_id="0x11a5"/>
   </cell>
   <cell>
```

```
   <control id="edge Phone numbers" window_id="0x11a1"/>
</cell>
<cell>
   <control id="btn Phone 1" window_id="0x1113"/>
</cell>
<cell>
   <control id="dd Phone 1" window_id="0x110a"/>
</cell>
<cell>
   <control id="btn Phone 2" window_id="0x1114"/>
</cell>
<cell>
   <control id="dd Phone 2" window_id="0x110b"/>
</cell>
<cell>
   <control id="btn Phone 3" window_id="0x1115"/>
</cell>
<cell>
   <control id="dd Phone 3" window_id="0x110c"/>
</cell>
<cell>
   <control id="btn Phone 4" window_id="0x1116"/>
</cell>
<cell>
   <control id="dd Phone 4" window_id="0x110d"/>
</cell>
<cell>
   <control id="edit Phone 1" window_id="0x1001"/>
</cell>
<cell>
   <control id="edit Phone 2" window_id="0x1002"/>
</cell>
<cell>
   <control id="edit Phone 3" window_id="0x1003"/>
</cell>
<cell>
   <control id="edit Phone 4" window_id="0x1004"/>
</cell>
<cell>
   <control id="lbl Addresses" window_id="0x11a7"/>
</cell>
<cell>
   <control id="edge Addresses" window_id="0x11a8"/>
</cell>
<cell>
   <control id="btn Address" window_id="0x6f2b"/>
</cell>
<cell>
   <control id="dd Address" window_id="0x1109"/>
</cell>
<cell>
   <control id="chb Address" window_id="0x1080"/>
</cell>
<cell>
```

```
      <control id="edit Address" window_id="0x1017"/>
   </cell>
   <cell>
      <control id="btn Map It" window_id="0x111c"/>
   </cell>
   <cell>
      <control id="Business card" window_id="0x11c7"/>
   </cell>
   <cell>
      <control id="lbl Notes" window_id="0x11a6"/>
   </cell>
   <cell>
      <control id="edge Notes" window_id="0x11b7"/>
   </cell>
   <cell>
      <control id="edit Notes" window_id="0x103f"/>
   </cell>
   <cell>
      <control id="Static In Folder:" window_id="0x200"/>
   </cell>
   <cell>
      <control id="Static In folder" window_id="0x201"/>
   </cell>
   <cell>
      <control id="Edit In folder" window_id="0x202"/>
   </cell>
   <cell>
      <control id="Static In shared folder" window_id="0x206"/>
   </cell>
   <cell>
      <control id="Modified by_1" window_id="0x204"/>
   </cell>
   <cell>
      <control id="Modified by_2" window_id="0x402"/>
   </cell>
   <cell>
      <control id="Static Full Name" window_id="0x1301"/>
   </cell>
   <cell>
      <control id="Static Phone Number Type_1" window_id="0x1304"/>
   </cell>
   <cell>
      <control id="Static Phone 1" window_id="0x1305"/>
   </cell>
   <cell>
      <control id="Static Phone Number Type_2" window_id="0x1306"/>
   </cell>
   <cell>
      <control id="Static Phone 2" window_id="0x1307"/>
   </cell>
   <cell>
      <control id="Static Phone Number Type_3" window_id="0x1308"/>
   </cell>
   <cell>
```

```
      <control id="Static Phone 3" window_id="0x130e"/>
   </cell>
   <cell>
      <control id="Static Phone Number Type_4" window_id="0x1310"/>
   </cell>
   <cell>
      <control id="Static Phone 4" window_id="0x1311"/>
   </cell>
   <cell>
      <control id="Static Address" window_id="0x1312"/>
   </cell>
   <cell>
      <control id="Static E-mail_1" window_id="0x1303"/>
   </cell>
   <cell>
      <control id="Static E-mail_2" window_id="0x1314"/>
   </cell>
   <cell>
      <control id="Static Contacts" window_id="0x1313"/>
   </cell>
   <cell>
      <control id="Static Categories" window_id="0x1104"/>
   </cell>
   <cell>
      <control id="Button Edit Yomi(V)..." window_id="0x1105"/>
   </cell>
   <cell>
      <control id="edit Furigana First" window_id="0x1011"/>
   </cell>
   <cell>
      <control id="edit Furigana Last" window_id="0x1012"/>
   </cell>
   <cell>
      <control id="Static Last(G)" window_id="0x11ab"/>
   </cell>
   <cell>
      <control id="Static /_1" window_id="0x11ad"/>
   </cell>
   <cell>
      <control id="edit Last" window_id="0x1014"/>
   </cell>
   <cell>
      <control id="Static First(M)" window_id="0x11b1"/>
   </cell>
   <cell>
      <control id="edit First" window_id="0x100a"/>
   </cell>
   <cell>
      <control id="RichEdit20WPT_5" window_id="0x1013"/>
   </cell>
   <cell>
      <control id="Static Department:" window_id="0x11be"/>
   </cell>
   <cell>
```

```
                        <control id="RichEdit20WPT_6" window_id="0x103e"/>
                     </cell>
                     <cell>
                        <control id="Static Contact Photograph_2"
window_id="0x1302"/>
                     </cell>
                     <cell>
                        <control id="rctrl_renwnd32" window_id="0x1023"/>
                     </cell>
                     <cell>
                        <control id="chb Address hidden" window_id="0x1085"/>
                     </cell>
                     <cell>
                        <control id="Static Postal code(U)" window_id="0x11b2"/>
                     </cell>
                     <cell>
                        <control id="Static /_2" window_id="0x11c9"/>
                     </cell>
                     <cell>
                        <control id="edit Postal Code" window_id="0x100b"/>
                     </cell>
                     <cell>
                        <control id="Static State(D):" window_id="0x11b3"/>
                     </cell>
                     <cell>
                        <control id="edit State" window_id="0x100c"/>
                     </cell>
                     <cell>
                        <control id="Static City(Q):" window_id="0x11b4"/>
                     </cell>
                     <cell>
                        <control id="edit City" window_id="0x100d"/>
                     </cell>
                     <cell>
                        <control id="Static Street(B):" window_id="0x11b5"/>
                     </cell>
                     <cell>
                        <control id="edit Street" window_id="0x100e"/>
                     </cell>
                     <cell>
                        <control id="Static Country/Region:" window_id="0x11b6"/>
                     </cell>
                     <cell>
                        <control id="REComboBox20W Country/Region:"
window_id="0x118b"/>
                     </cell>
                     <cell>
                        <control id="Button Contacts..." window_id="0x10a3"/>
                     </cell>
                     <cell>
                        <control id="RichEdit20WPT_11" window_id="0x10a4"/>
                     </cell>
                     <cell>
```

```
                        <control id="NativeFontCtl" window_id="0x1309"/>
                      </cell>
                      <cell>
                        <control id="SysAnimate32" window_id="0x10fe"/>
                      </cell>
                      <cell>
                        <control id="Static" window_id="0x11c3"/>
                      </cell>
                      <cell>
                        <control id="Static /_4" window_id="0x11ca"/>
                      </cell>
                      <cell>
                        <control id="Static /_3" window_id="0x11c8"/>
                      </cell>
                    </stack>          </cell>
                </stack>
              </cell>
            </stack>
          </cell>
    </page>
</form>
```

Index