

# Magic xpi 4.5 with Dynamics CRM Seminar

Self-Paced Tutorial

Book ID: UTLSDYNMI45

Edition: 1.0, June 2016

Course ID: UCLDYNMI45

Magic University Official Courseware

The information in this manual/document is subject to change without prior notice and does not represent a commitment on the part of Magic Software Enterprises Ltd.

Magic Software Enterprises Ltd. makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose.

The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms and conditions of the license agreement. It is against the law to copy the software on any medium except as specifically allowed in the license agreement.

No part of this manual and/or databases may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or information recording and retrieval systems, for any purpose other than the purchaser's personal use, without the prior express written permission of Magic Software Enterprises Ltd.

All references made to third-party trademarks are for informational purposes only regarding compatibility with the products of Magic Software Enterprises Ltd.

Unless otherwise noted, all names of companies, products, street addresses, and persons contained herein are part of a completely fictitious scenario or scenarios and are designed solely to document the use of Magic xpi.

Magic™ is a trademark of Magic Software Enterprises Ltd.

Btrieve® and Pervasive.SQL® are registered trademarks of Pervasive Software Inc.

IBM®, Topview™, System i5/System i™, pSeries®, xSeries®, RISC System/6000®, DB2®, WebSphere®, Domino®, and Lotus Notes® are trademarks or registered trademarks of IBM Corporation.

Microsoft®, FrontPage®, Windows™, WindowsNT™, ActiveX™, Exchange 2007™, Dynamics® CRM, SharePoint®, Excel®, and Word® are trademarks or registered trademarks of Microsoft Corporation.

Oracle®, JD Edwards EnterpriseOne®, JD Edwards World®, and OC4J® are registered trademarks of the Oracle Corporation and/or its affiliates.

Google Calendar™ and Google Docs™ are trademarks of Google Inc.

Salesforce® is a registered trademark of salesforce.com Inc.

SAP® Business One and SAP® R/3® are registered trademarks of SAP AG in Germany and in several other countries.

Dynamics CRM is a trademark of Dynamics CRM in the United States, the European Union and other countries.

Linux® is a registered trademark of Linus Torvalds.

UNIX® is a registered trademark of UNIX System Laboratories.

GLOBETrotter® and FLEXIm® are registered trademarks of Macrovision Corporation.

Solaris™ and Sun ONE™ are trademarks of Sun Microsystems Inc.

HP-UX® is a registered trademark of the Hewlett-Packard Company.

Red Hat® is a registered trademark of Red Hat Inc.

WebLogic® is a registered trademark of BEA Systems.

Interstage® is a registered trademark of the Fujitsu Software Corporation.

JBoss™ is a trademark of JBoss Inc.

GigaSpaces, GigaSpaces eXtreme Application Platform (XAP), GigaSpaces eXtreme Application Platform Enterprise Data Grid (XAP EDG), GigaSpaces Enterprise Application Grid, GigaSpaces Platform, and GigaSpaces, are trademarks or registered trademarks of GigaSpaces Technologies.

Clip art images copyright by Presentation Task Force®, a registered trademark of New Vision Technologies Inc.

This product uses the FreedImage open source image library. See <http://freeimage.sourceforge.net> for details

This product uses icons created by Axialis IconWorkShop™ (<http://www.axialis.com/free/icons>)

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

This product includes software developed by Computing Services at Carnegie Mellon University (<http://www.cmu.edu/computing/>).

Copyright © 1989, 1991, 1992, 2001 Carnegie Mellon University. All rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).

This product includes software that is Copyright © 1998, 1999, 2000 of the Thai Open Source Software Center Ltd. and Clark Cooper.

This product includes software that is Copyright © 2001-2002 of Networks Associates Technology Inc All rights reserved.

This product includes software that is Copyright © 2001-2002 of Cambridge Broadband Ltd. All rights reserved.

This product includes software that is Copyright © 1999-2001 of The OpenLDAP Foundation, Redwood City, California, USA. All Rights Reserved.

All other product names are trademarks or registered trademarks of their respective holders.

### **Magic xpi 4.5 with Dynamics CRM Seminar**

June 2016

Copyright © 2013-2016 by Magic Software Enterprises Ltd. All rights reserved.

## Table of Contents

Introduction .....	5
About the Seminar .....	5
Dynamics CRM Connector .....	7
Magic xpi Architecture with Dynamics CRM Connector .....	8
Connecting to Dynamics CRM .....	8
Creating a Project .....	8
Supported Operations .....	11
Summary .....	12
Querying Dynamics CRM via Magic xpi .....	13
Preview of the Flow .....	14
Triggering the Flow .....	15
Query Operation .....	19
Querying Based on an Operator .....	28
Summary .....	28
Adding an Object .....	29
Adding an Account .....	30
Adding a Contact .....	33
Summary .....	36
Dealing with Products .....	37
Create Products in Dynamics CRM .....	38
Check Whether the Products Exist .....	40
Summary .....	42
Working with Sales Orders .....	43
Query Price List .....	44



Get Price List ID .....	44
Create Sales Order .....	45
Create Each Product.....	47
Summary .....	48

## Introduction

Welcome to Magic Software University's **Magic xpi 4.5 with Dynamics CRM Seminar** self-paced tutorial. We, at Magic Software University, hope that you will find this tutorial informative and that it will assist you in getting started with this exciting product.

### About the Seminar

The seminar is intended for people with a knowledge of Dynamics CRM who want to know how to successfully use Magic Software Enterprises' Magic xpi product, and how to integrate Magic xpi with Dynamics CRM.

During the seminar you will learn about the Magic xpi Dynamics CRM connector and how Magic xpi integrates with Dynamics CRM.

## Course Prerequisites

Before you start with the course there is basic knowledge that you need to have:

Development knowledge	Familiar with Magic xpi 4.5, Magic xpi 4.1 or iBOLT/Magic xpi 3.x
Dynamics CRM	Knowledge of Dynamics CRM

Your computer must also meet some basic requirements:

Hardware	<ul style="list-style-type: none"> <li>• Windows XP Pro and later. The course was tested on Windows 7</li> <li>• Pentium processor 1.8GHz and upwards</li> <li>• 4Gb RAM or greater</li> <li>• At least 1Gb of free space</li> <li>• Screen resolution of at least 1024x768 pixels</li> </ul>
Magic xpi	<p>You will need to install the following:</p> <ul style="list-style-type: none"> <li>• Magic xpi V4.5</li> </ul>
License	For deployment purposes, you need the IBDYCRM license from your Magic Software Enterprises representative. This is not required for development purposes.
Dynamics CRM	This seminar has been designed using the Dynamics CRM installation in Magic Software headquarters. The demonstration data is based on Microsoft Dynamics® CRM 2013 SP1.
.NET framework 4.5.2	This is a prerequisite for the Dynamics CRM connector to work. If it is not already installed on your computer, install it using the following link: <a href="http://www.microsoft.com/en-us/download/details.aspx?id=42637">http://www.microsoft.com/en-us/download/details.aspx?id=42637</a> .

# Lesson 1

## Dynamics CRM Connector

The Magic xpi Dynamics CRM connector enables a work flow between Magic xpi and Dynamics CRM.

Using the Dynamics CRM connector, you can query, create, update, delete, assign and change the state of entities in Dynamics CRM.

You can also trigger a Magic xpi flow when actions such as create, update, or delete are performed in Dynamics CRM.

As was mentioned in the Prerequisites section, for deployment purposes, to work with the Dynamics CRM connector, you need a special Magic xpi license: **IBDYCRM**.

This lesson covers various topics including:

- An introduction to the Dynamics CRM connector
- Installing the newest version of the Dynamics CRM connector
- Creating a Dynamics CRM resource
- Connecting Magic xpi to Dynamics CRM

## Magic xpi Architecture with Dynamics CRM Connector

The Magic xpi Dynamics CRM connector works with Dynamics CRM's REST API.

The Dynamics CRM connector supports Magic xpi's XML interface. The Dynamics CRM connector can create, query, update, and delete data objects as well as change the stage or assign an owner to an entity in Dynamics CRM.



This course uses Dynamics CRM 2013 SP1. Other versions will not be demonstrated in this course. However, most of the functionality is similar.

## Connecting to Dynamics CRM

The Dynamics CRM connector needs to be connected to a specific user in Dynamics CRM. Therefore, before working with the Magic xpi Dynamics CRM connector, you need:

- A valid Dynamics CRM resource/server
- A valid Dynamics CRM user name
- A valid Dynamics CRM password



If you want to connect to a Dynamics CRM 2013 on-premise machine you simply make sure that the **DCRM2013OnPremise** flag is set to **Y**.

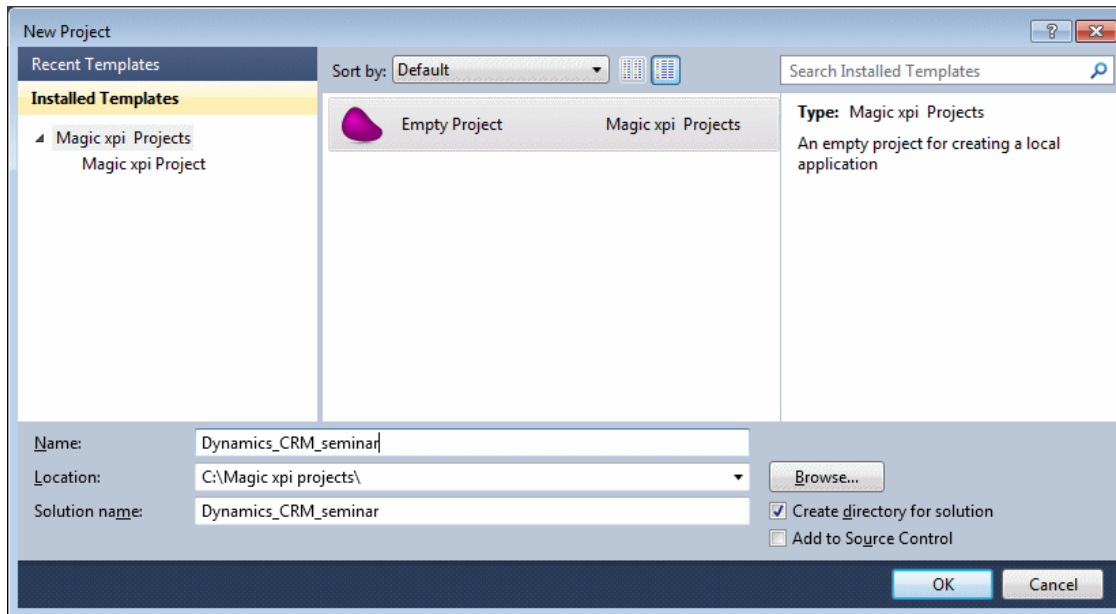
## Creating a Project

As with any development project, the first step is to create a new Magic xpi project.

To create a new Magic xpi project:

1. Open Magic xpi.
2. Click on the **File** menu, and select **New**. The **New Project** dialog box will open.
3. Create a new project called **Dynamics\_CRM\_seminar**.





For the purpose of this course, data has been prepared for you.

4. Copy the **course\_data** folder into the **Dynamics\_CRM\_Seminar\Dynamics\_CRM\_Seminar** folder. This subfolder was created when you created the project.

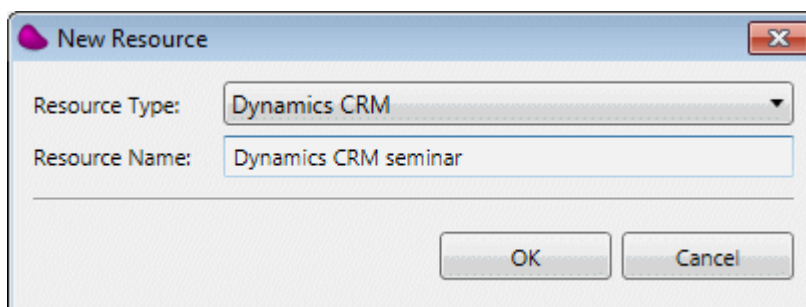


A final version of the project is provided in the **Final\_Dynamics\_CRM\_Project** folder.

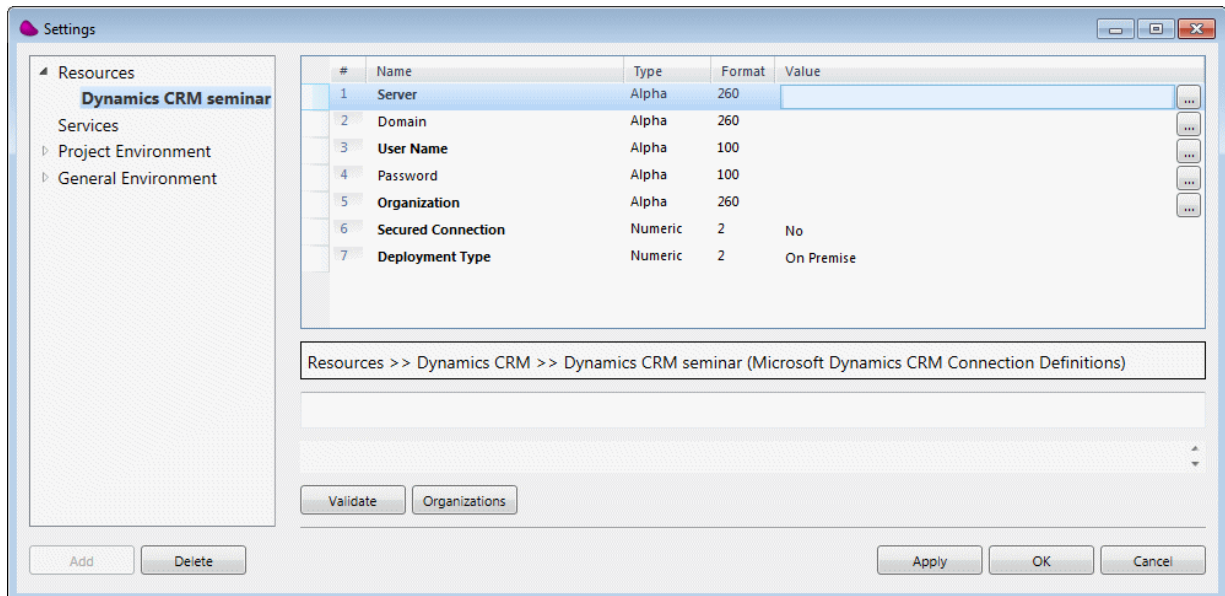
## Defining a Resource

Before using the Dynamics CRM connector in a step, you need to define the **Dynamics CRM** resource.

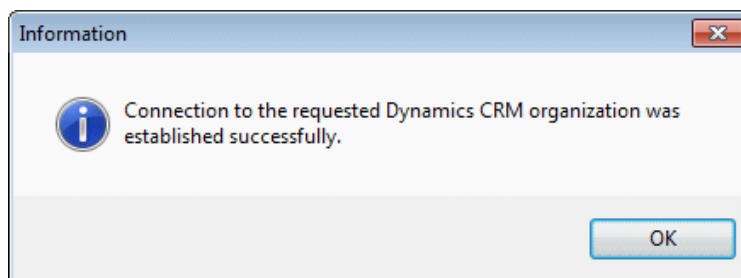
1. From the **Project** menu, select **Settings**.
2. While parked on the **Resources** option, click **Add** to add a resource.
3. From the **Resource Type** field, select **Dynamics CRM**.
4. Name the resource: **Dynamics CRM seminar**.



There are a number of mandatory settings to be defined in the Dynamics CRM resource. These are the settings that appear in bold.



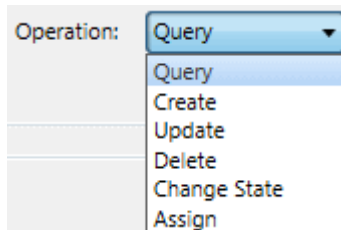
5. Enter your **Server**, **User Name**, and **Password** for the Dynamics CRM server. Note that the password is case sensitive.
6. Set the **Secured Connection** setting to **Yes**.
7. You can leave the **Deployment Type** setting as **On Premise**.
8. Click the **Organizations** button and select the organization that you want to connect to.
9. Click the **Validate** button to check your connection. If all of the settings were entered correctly, you should see the following message:



You have now successfully created a connection from Magic xpi to Dynamics CRM.

## Supported Operations

As was stated at the beginning of this lesson, the Dynamics CRM connector comes with six operations. These include CRUD operations: **Create**, **Query** (Read), **Update** and **Delete**, as well as **Change State** and **Assign**.



Only the relevant operations for each entity and its privileges will be displayed.

### Query

This operation is used to retrieve data from an entity. You can use the Data Mapper to retrieve entities based on specific criteria. For example, you can retrieve all contacts with Title = Professor. If you want to retrieve all available contacts, you do not need to supply any mapping.

### Create

This operation is used to create new entities in your organization's data. You can use the Data Mapper to specify new entity fields' values.

### Update

This operation is used to update a specific entity in your organization's data. When you update an entity, you need to know its ID. For example, you can use the **Query** operation (above) to retrieve an entity ID, and then you can call the **Update** operation. You should use the Data Mapper to set the entity ID, as well as the other fields that you want to update.

### Delete

This operation is used to delete a specific entity from your organization's data. If you want to delete an entity, you need to know its ID. For example, you can use the **Query** operation (above) to retrieve an entity ID, and then you can call the **Delete** operation. You should use the Data Mapper to set the entity ID. This is the only value that you should set.

## Assign

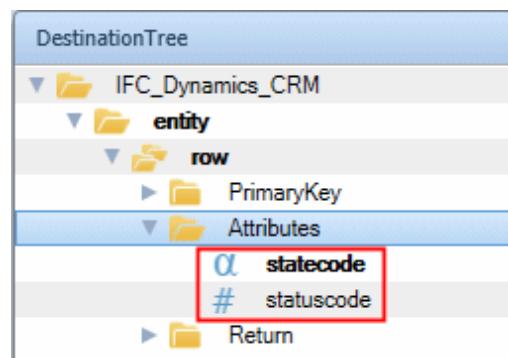
This operation is used to change the owner of a specific entity in your organization. You can assign an entity to a user or to a team, depending on your Dynamics CRM resource's account privileges.

## Change State

This operation is used to change an entity's state. Each entity has a state, such as **Active**, **Inactive** and **Open**, which can be changed with this operation.

When you use this operation, the Data Mapper has two nodes:

- **statecode**, which is the Status
- **statuscode**, which is the Status reason



You should provide **statecodes** as state strings (**Active**, **Inactive**, etc) and not as a number representing the state. You should provide **statuscodes** as numeric values and not as strings.

Dynamics CRM only accepts **statuscodes** that are valid for a particular **statecode**. For example, you can have a Lead entity with a **statecode** of **Open** (meaning 0). This **statecode** can have a **statuscode** of 1 (meaning: New) or 2 (meaning: Contacted).

For a list of the **statecodes** and **statuscodes**, see: [https://technet.microsoft.com/en-us/library/dn531157\(v=crm.7\).aspx](https://technet.microsoft.com/en-us/library/dn531157(v=crm.7).aspx).

## Summary

In this lesson:

- You were introduced to the Magic xpi Dynamics CRM connector.
- You installed the newest version of the Dynamics CRM connector.
- You created a project for the seminar.
- You created a Dynamics CRM resource and connected Magic xpi to Dynamics CRM.

# Lesson 2

## Querying Dynamics CRM via Magic xpi

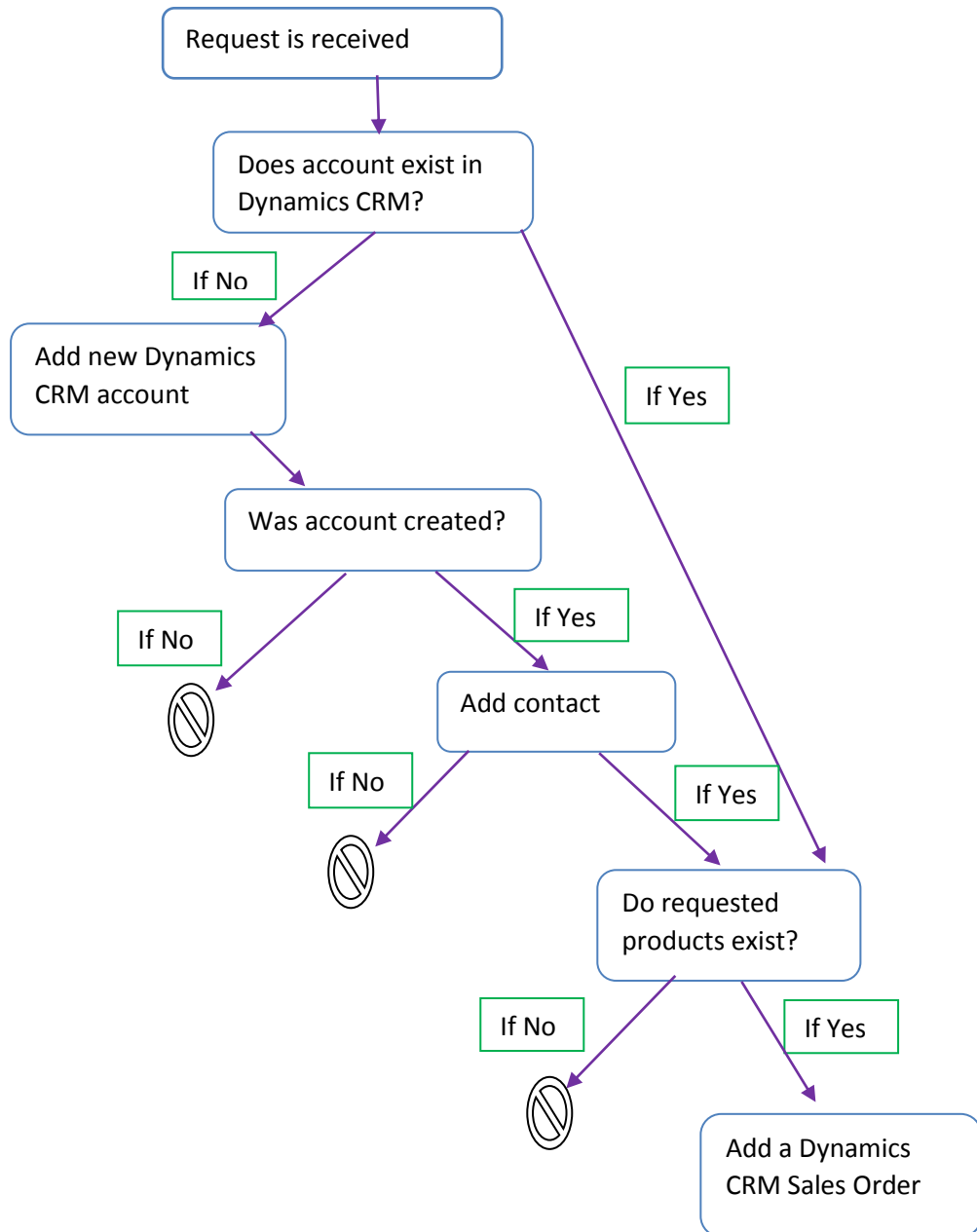
A Dynamics CRM Query operation is used to retrieve data from an object according to specific search criteria.

This lesson covers various topics including:

- A preview of the flow that will be built throughout this course
- Query operation
- Query operators

## Preview of the Flow

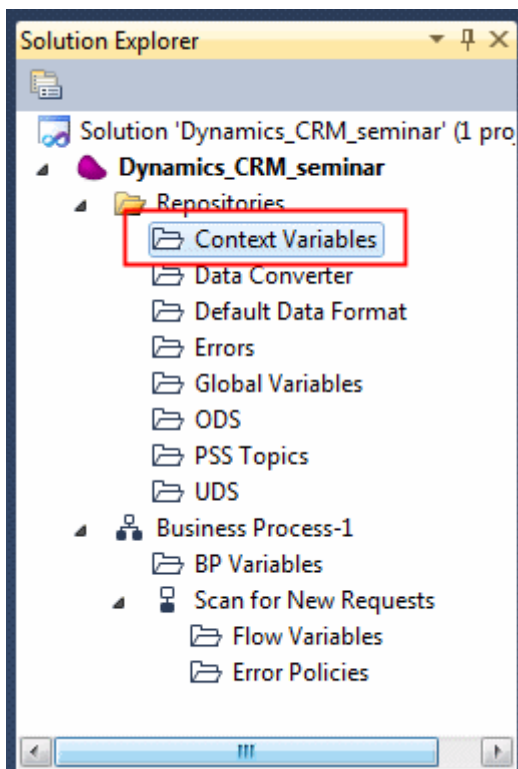
The business process logic of the Magic xpi flow that you will create is as follows:




## Triggering the Flow

You'll use a trigger to activate the flow. Since Microsoft Dynamics CRM does not provide triggering as part of their API, you'll look for new and/or updated objects using Magic xpi's Directory Scanner component.

1. Rename the default flow and call it **Scan for New Requests**.
2. From the **Solution Explorer**, double-click on the **Context Variables** folder.



3. From the **Context Variables** tab, click **Add** and add the following variable:
  - **C.RequestXML**, a BLOB variable
4. Click the  **Save** button.
5. From the Solution Explorer, double-click on the **Flow Variables** folder (under the new **Scan for New Requests** flow).

6. From the **Flow Variables** tab, click **Add** and add the flow variables listed below. When you are asked to use these variables, an explanation about them will be provided.

- **F.AccountXML**, a BLOB variable
- **F.RequestFileName**, an Alpha variable of size 255
- **F.ContactXML**, a BLOB variable
- **F.AccountExists**, a Logical variable with a default value of 'FALSE'LOG. The **F.AccountExists** variable will be used if a query returned a user record from Dynamics CRM.
- **F.AccountId**, an Alpha variable of size 36

7. Click the  **Save** button.

IDs in the Dynamics CRM connector are GUIDs, meaning that they are 36 characters long in the following format: 8-4-4-4-12. For example:



46e58060-c617-e411-b4e7-002219a2d1e6

Therefore, the ID variables used in this course are set to 36.



If you are using an ID in a database's SELECT statement, the ID is surrounded by curly brackets and, therefore, is 38 characters long. To use it from the database in the connector, you need to strip the ID of the brackets.

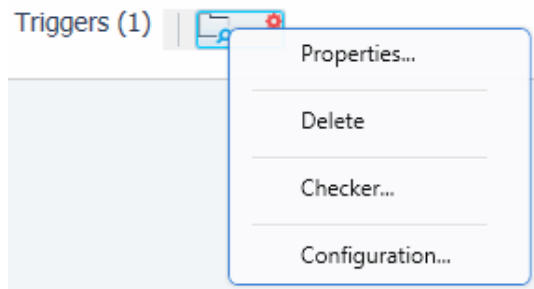
You will receive the request using the Directory Scanner component. There are two ways of using the Directory Scanner component: Trigger or Step. In this example you will use the Trigger mode.

The purpose of this flow is to scan the **in** folder to see if a new XML request is there.

8. From the **Toolbox** (in the **FileManagement** section), drag a **Directory Scanner** component to the Trigger area.
9. In the trigger's **Properties** pane, set the **Trigger Name** property to: **Wait for File**.



10. Right-click on the component and select **Configuration**.



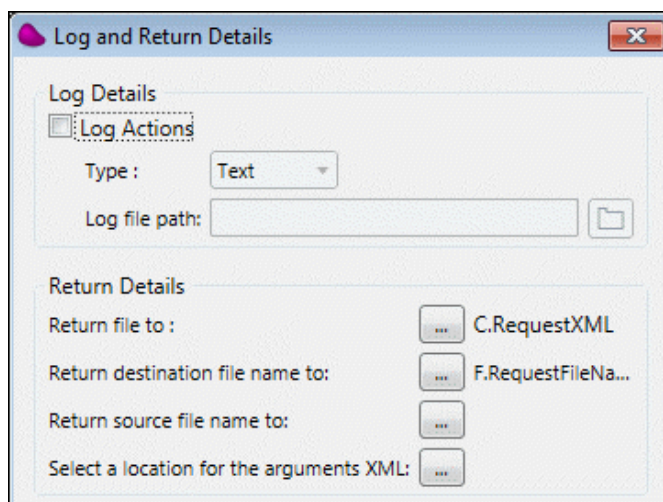
11. From the **Component Configuration: Directory Scanner** dialog box, click **New** to define the trigger.

12. Define the following:

- a. Leave the **Source** as **LAN**.
- b. Set the **Directory** to: `EnvVal ('currentprojectdir')&'course_data\in\'`. The `currentprojectdir` environment variable contains the path to the directory where the current project resides.
- c. In the **Filter** property leave the default of `*.*`.
- d. Leave the **Action** as **Move**.
- e. Set the destination **Directory** to `EnvVal ('currentprojectdir')&'course_data\out\'`.

13. Click the **Advanced** button.

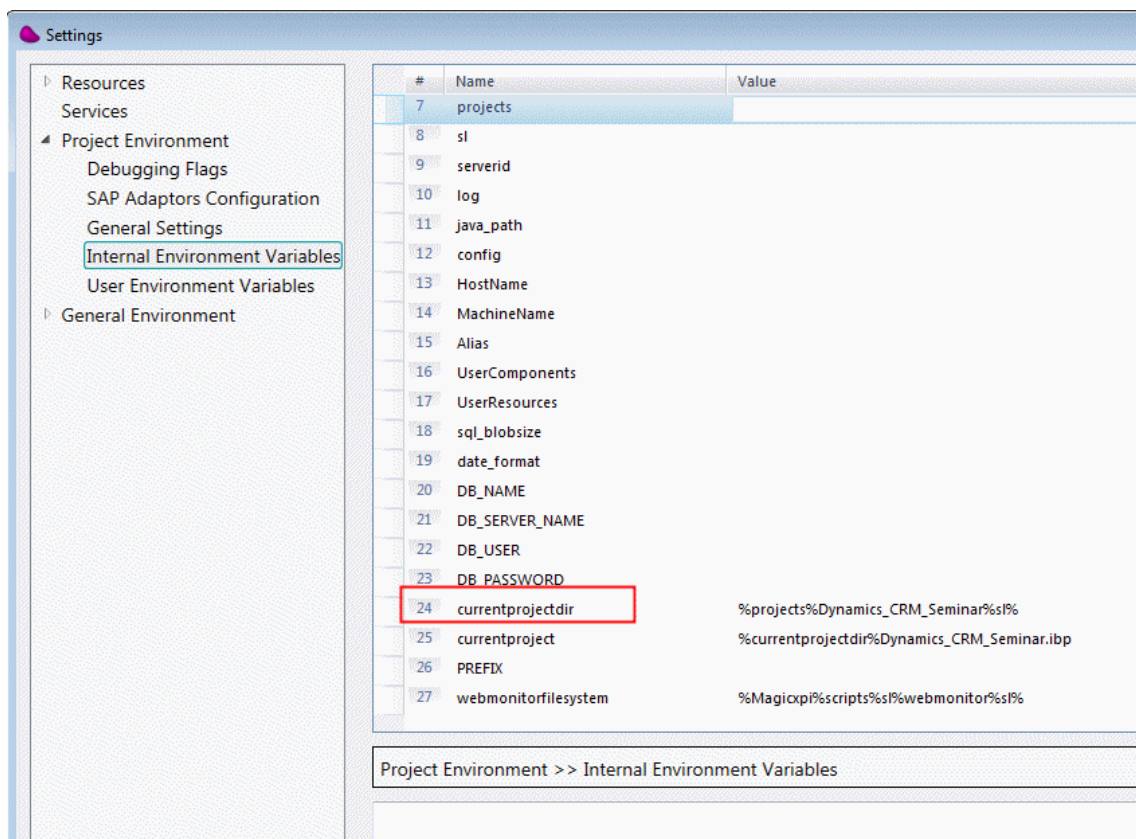
- a. Set the **Return file to** property to `C.RequestXML`. This is the variable that the Directory Scanner will return the content of the file to.
- b. Set the **Return destination file name to** property to `F.RequestFileName`. This is the name of the variable that the Directory Scanner will return the name of the file to.



14. Click **OK**.

15. From the **Project** menu, select **Settings**.

16. Go to the **Project Environment** section and click the **Internal Environment Variables** option.
17. Check that the **currentprojectdir** environment variable is pointing to the correct location. We used this environment variable in the Directory Scanner component, so we need to check this environment variable so that the trigger will know where to take the files from.



You have finished defining the trigger.

## Query Operation


You will now check whether the customer exists as an account in Dynamics CRM.

In Dynamics CRM:



- An *account* is a company.
- A *contact* is a person.
- A *customer* is either a person or a company.

### Query for Account

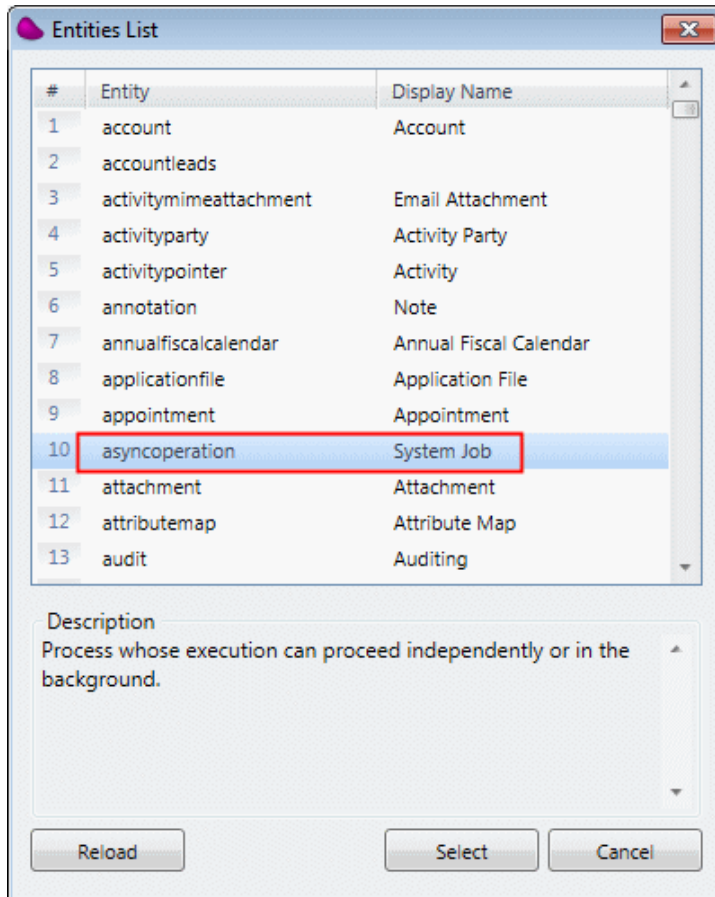
1. Drag a Dynamics CRM connector  as the first step in the **Scan for New Requests** flow and name the step: **Query for Account**.
2. Right-click on the connector and select **Configuration**. The **Dynamics CRM Configuration** dialog box opens.

In the **Resource Name** field, you'll see the **Dynamics CRM seminar** resource. Since this is, currently, the only Dynamics CRM resource, it is selected automatically by Magic xpi.

3. From the **Entity** field, click the selection button .

This loads the information from the Dynamics CRM server. Magic xpi needs to fetch the objects exposed by the Dynamics CRM API before accessing them. Magic xpi connects directly to the Dynamics CRM server, retrieves the available entities, and displays them in a list.

In the Entities List you can see that there is a **Display Name** column. This lets you know what the entity represents in the Dynamics CRM interface. Some are self-explanatory, but for some you'll find the **Display Name** column helpful. For example, you'll see the **asyncoperation** entity represents the **System Job** entity in Dynamics CRM.

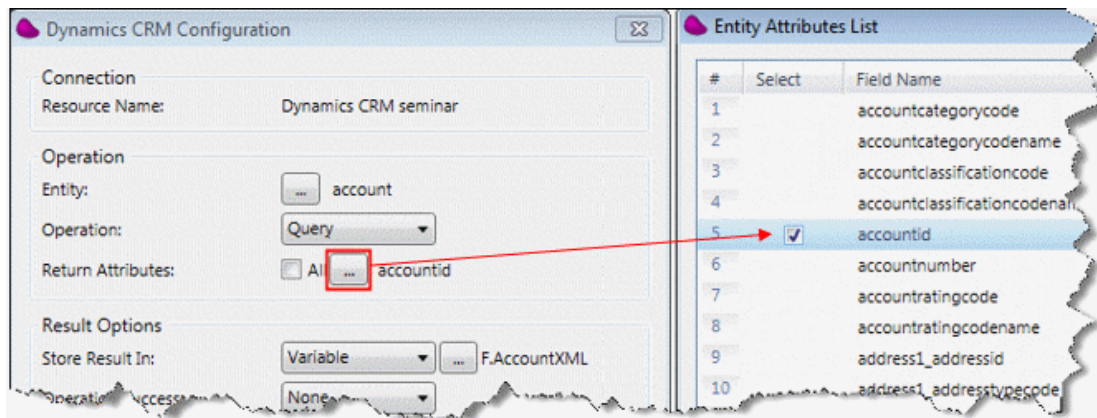


4. Select **account**.

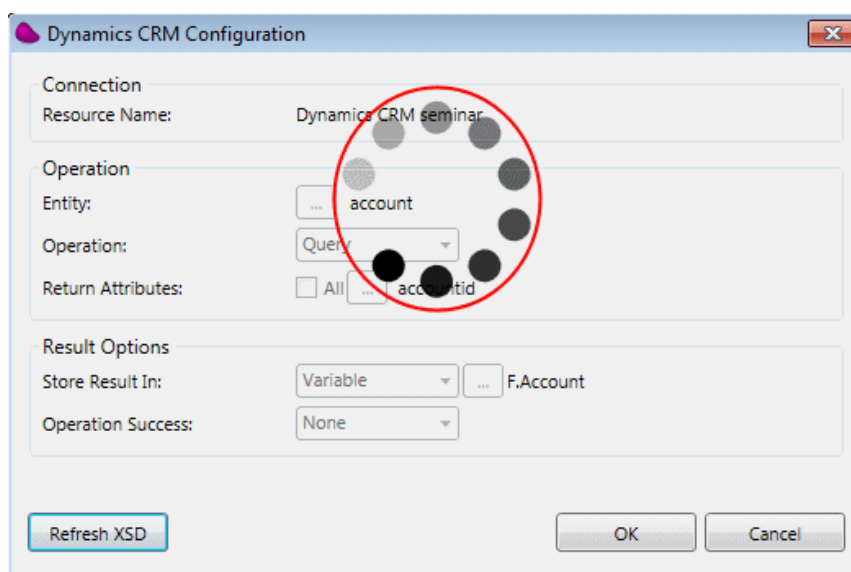
Since you're simply just browsing to see if an account exists in Dynamics CRM, you'll perform a Query. The Dynamics CRM Query operation is used to retrieve data from an entity. The rest of the operations will be discussed later on in this seminar.

5. From the **Operation** field, select **Query**.

- The **Return Attributes** option enables you to define which fields will be returned in the result XML. Clear the **Return Attributes** check box and select **accountid**. In general, this is recommended, because it reduces your result set, the size of the fields that are returned.



- The **Store result in** field will hold the XML retrieved from Dynamics CRM. You can select either a file or a variable. Select **Variable**, and then select the **F.AccountXML** variable that you defined earlier.
- It's a good idea to click the **Refresh XSD** button to make sure that you're using the latest module metadata. Changes that are made in the Dynamics CRM environment – customizations and so forth – are all pulled into the integration environment so that when you do the data mapping, it's all there and available to you. The following in progress image will appear, showing that Magic xpi is retrieving data from the Dynamics CRM server.



- Click **OK**.

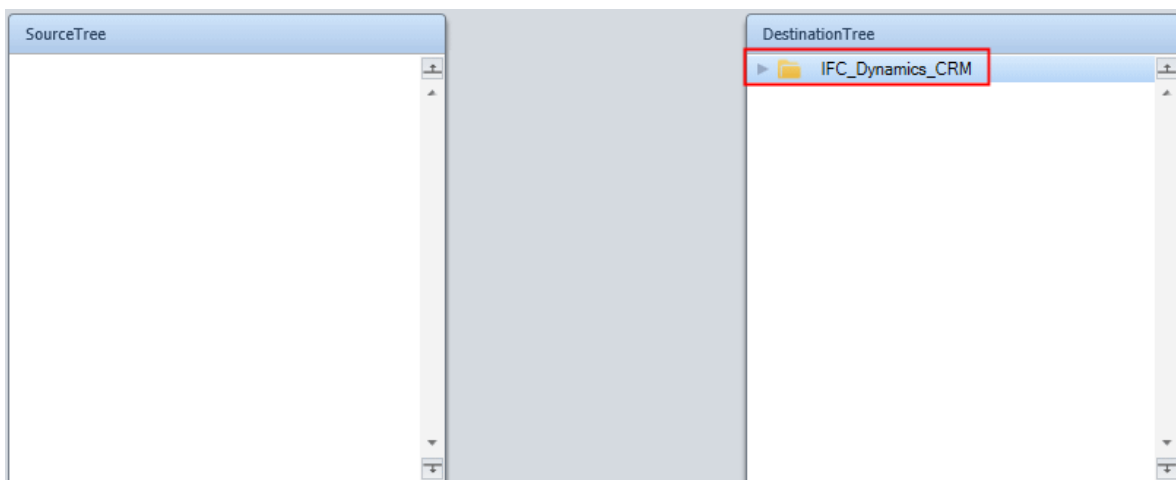


The Magic xpi Dynamics CRM connector saves the XML Schema, the XSD, in the following directory:

[project dir]\[project name]\DynamicsCRM\XSD\[project name]

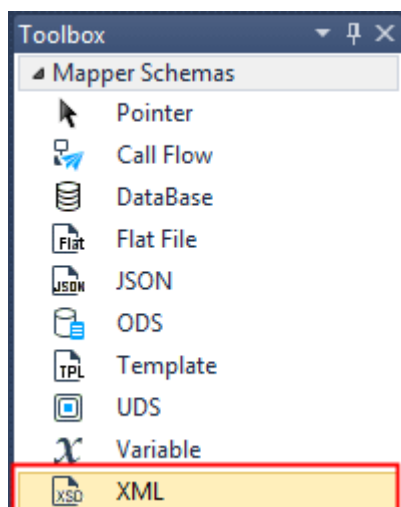
Since the Dynamics CRM connector uses the XML interface, you will use the Data Mapper to configure it.

After defining the properties for the Dynamics CRM connector, a new **IFC Model** entry was created in the **Destination** section: **IFC\_Dynamics CRM**.

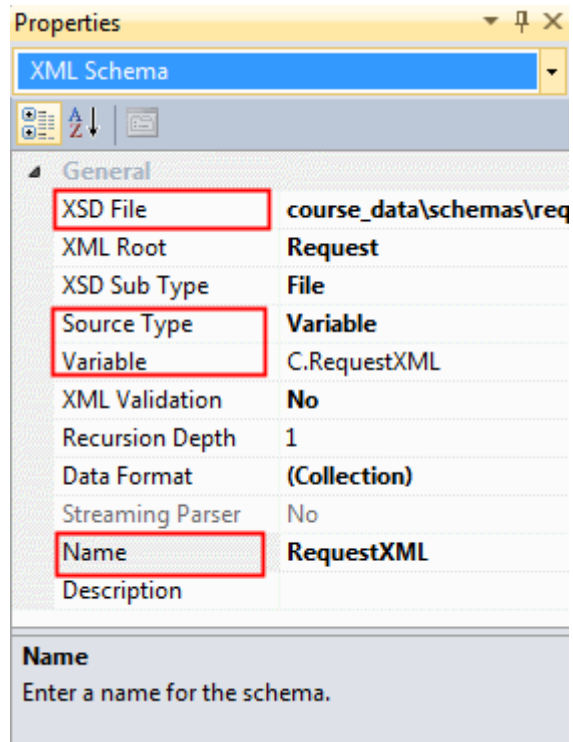


You need to use the request XML that was retrieved by the Directory Scanner to check whether the account exists in Dynamics CRM. Therefore, you need to have XML as the source.

1. From the **Toolbox**, drag an **XML** entry to the Source pane of the Data Mapper.



2. Go to the **Properties** pane.
  - a. Set the **Name** property to **RequestXML**.
  - b. In the **XSD File** property, select the following schema:  
**course\_data\schemas\request.xsd**
  - c. Set the **Source Type** to **Variable** and select the **C.RequestXML** variable.



3. Click the save icon.

The next stage is to map.

You need to send the customer name to Dynamics CRM to query its existence. Therefore, in the destination, you'll use the **name** node in the Dynamics CRM **account** entity that you previously configured.



To expand all of the nodes, park on the top node of the Source or Destination side right-click and select **Expand all**.

4. In the Source pane, expand the **RequestXML > CustomerDetail** node.
5. In the Destination pane, expand the **account > row > Attributes** node.
6. Connect the **AccountName** node to the **name** node.




- Click the save icon and go back to the Flow area.



Sometimes there are many options listed in a node. To easily search for one, press **Ctrl+Shift+F** from one of the panes or go to the **Edit** menu, select **Find** and then **Find Text**. In the **Find Text** dialog box, you can narrow your search. Once the cursor is on one of the items that meets your search criteria, you can press **F3** to go to the next item.

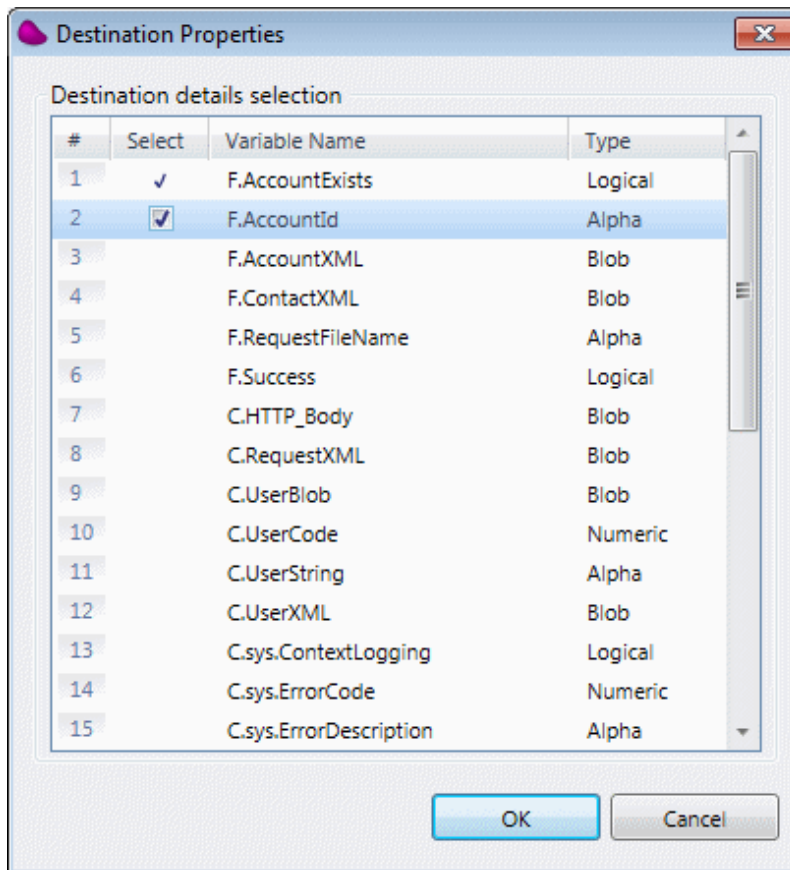
## Check If Account Exists

Now you'll check if the variable received a True value, meaning that the account exists.

- Drag a Data Mapper component as a child of the **Query for Account** step and name the new step: **Check If Account Exists**.
- Right-click on the new step and select **Configuration**.
- From the **Toolbox**, drag an **XML** entry to the Source pane of the Data Mapper.
- Go to the **Properties** pane.
  - In the **XSD File** property, select the following schema:  
`dynamicscrm\XSD\Dynamics CRM seminar\account.xsd`
  - Set the **Source Type** to **Variable** and select the **F.AccountXML** variable.
- From the **Toolbox**, drag a **Variable** entry to the Destination pane.
- Go to the **Properties** pane.
- From the **Variables** property, click the selection button 




8. Select both the **F.AccountExists** and **F.AccountId** variables.



9. Click **OK**.
10. On the Source pane, open the following node: **account > row > Attributes**.
11. On the Destination pane, open the **Instance** node if it's not yet open.
12. Connect the **accountid** node to the **F.AccountExists** node.
13. While the cursor is on the **F.AccountExists** node, go to the **Properties** pane.
14. In the **Calculated Value** field, enter the following expression:

```
ISNULL (Src.S1/account/row/Attributes/accountid ) OR NOT (
Src.S1/account/row/Attributes/accountid = '' )
```

In the expression above, the path **Src.S1/account/row/Attributes/accountid** is entered by clicking the **Source Nodes**  icon at the top of the Expression Editor.

This expression returns True if there is a value. This means that if the **accountid** field is not null or empty, then the account exists.

15. Also connect **accountid** to **F.AccountId**. This will update the **F.AccountId** variable with the value of the Dynamics CRM **accountid** fields. This connection will be used in a later step.
16. Save and return to the Flow area.


## Testing Your Project

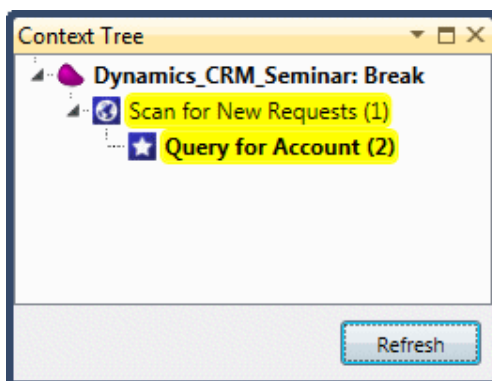
You will want to test your flow to make sure it works.

1. Right-click on the **Query for Account** step and select **Breakpoint**. A red dot will appear next to the step. A breakpoint means that processing will halt at that point.
2. In the **course\_data/out** folder, you'll find XML requests. Copy the **request001.xml** file from the **out** folder to the **in** folder. This XML file includes an account that does not yet exist in Dynamics CRM.



If you want to create a log file for the Dynamics CRM connector, set the **DebugMDCRMComponent** flag to **Y**. You'll find the flag in the **ifs.ini** file or you can configure it from the **Project > IFS Settings** menu option. A log file called **MDCRM\_Debug.log** will be created in the **logs** directory if Magic xpi executes a Dynamics CRM step using the server.

3. From the toolbar, click the Start Debugging  icon (or from the **Debug xpi** menu, select **Start Debugging**). You can also press **F5** to start the Debugger. Magic xpi checks the project for any syntax errors. If there are syntax errors, you will not be able to continue. There are various types of syntax errors, such as a mandatory property that was not defined or was incorrectly defined. When the breakpoint is reached (which can sometimes take a few seconds), the **Toolbox** will become the **Context Tree**.

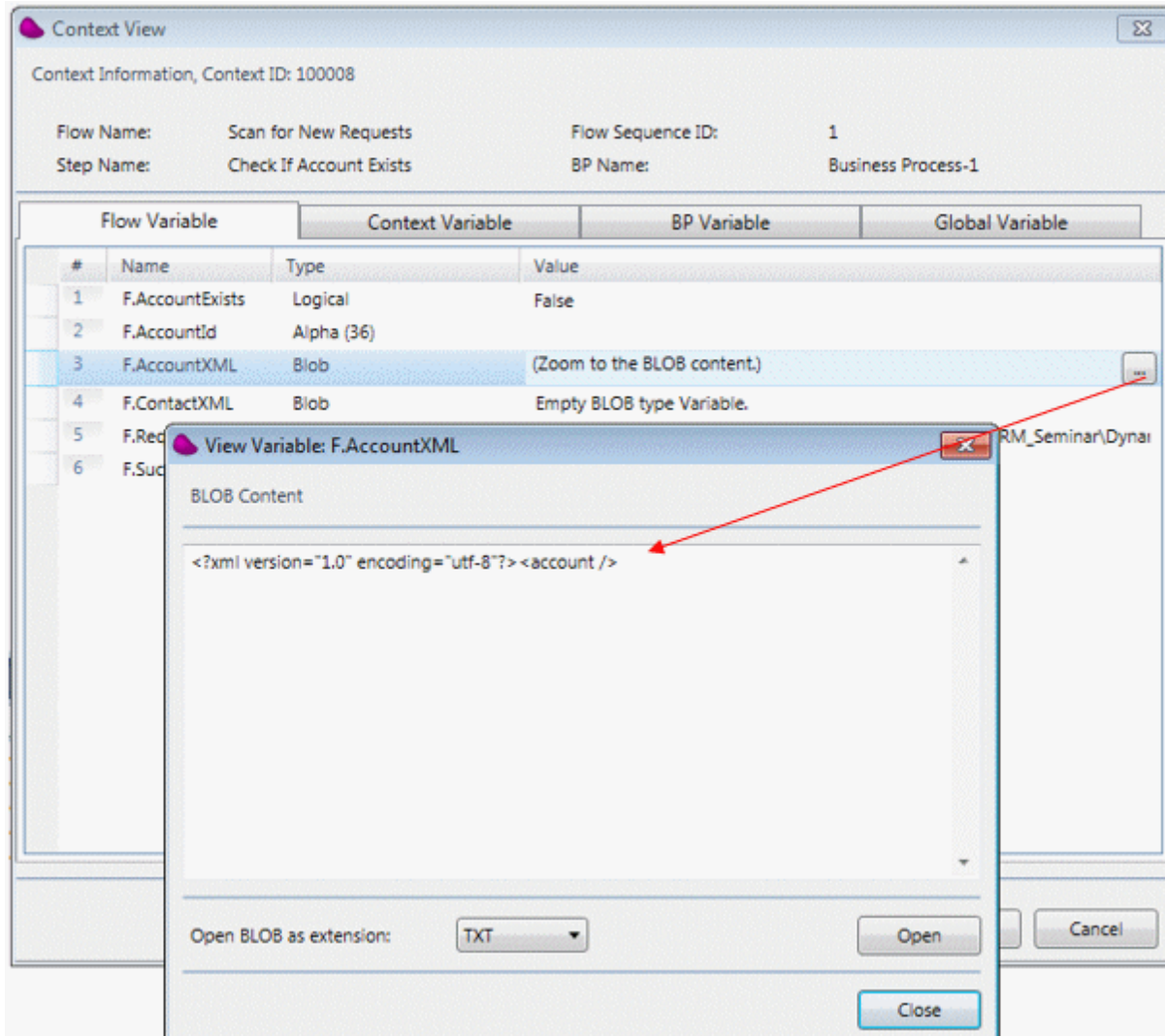


4. From the Context Tree, right-click on the **Query for Account** option and select **Step**. This will run the second step.

You now want to look at the **F.AccountXML** variable, which is the variable that you selected in the **Store result in** field.

5. In the **Context Tree**, right-click on the **Check If Account Exists** option and select **Context View** (or select it from the **Debug xpi** menu).

6. Find the **F.AccountXML** variable and notice that it says **(Zoom to the BLOB content)** in the **Value** column. If it says, **Empty BLOB type Variable**, then the flow did not successfully execute.
7. Click the zoom button and you'll see that the variable was filled in. However, you can also see that the account does not exist. Later on in the seminar, you'll see how to handle this.



Context View

Context Information, Context ID: 100008

Flow Name: Scan for New Requests      Flow Sequence ID: 1  
Step Name: Check If Account Exists      BP Name: Business Process-1

	Flow Variable	Context Variable	BP Variable	Global Variable
#	Name	Type	Value	
1	F.AccountExists	Logical	False	
2	F.AccountId	Alpha (36)		
3	F.AccountXML	Blob	(Zoom to the BLOB content.)	
4	F.ContactXML	Blob	Empty BLOB type Variable.	
5	F.Rec			
6	F.Suc			

View Variable: F.AccountXML

BLOB Content

```
<?xml version="1.0" encoding="utf-8"?> <account />
```

Open BLOB as extension: TXT

Open      Cancel

Close

8. **Close** to go back to development mode.

## Querying Based on an Operator

Queries are used to retrieve information from entities. You can use the Data Mapper to retrieve entities based on specific criteria. For example, you can retrieve all contacts with Title = Professor. If you want to retrieve all available contacts, you do not need to supply any mapping. Currently supported comparison operators for this operation, are listed here.

Note that these operators can only be used on String fields.

Condition	Operator	Value
Equals x	=	x
Does not equal x	!=	x
Is greater than x	>	x
Is greater than or equal to x	>=	x
Is less than x	<	x
Is less than or equal to x	<=	x
Begins with x	LIKE	x%
Does not begin with x	NOTLIKE	x%
Ends with x	LIKE	%x
Does not end with x	NOTLIKE	%x
Contains x	LIKE	%x%
Does not contain x	NOTLIKE	%x%

## Summary

In this lesson:

- You saw a preview of the flow that will be built throughout this course.
- You created a trigger for the flow using the Directory Scanner component.
- You learned about the Query operation.
- You also were shown the Query operators that can be used.

# Lesson 3

## Adding an Object

In the previous lessons you learned how to fetch information from Dynamics CRM.

Querying a database is not the only operation needed in a project. It is often necessary to add an object to the database.

In this lesson, you'll see how Magic xpi enables you to add an entry to the Dynamics CRM database.

You'll also learn about using entries in Dynamics CRM selection lists.

## Adding an Account

The steps needed to add an object are very similar to the steps required to query an object.

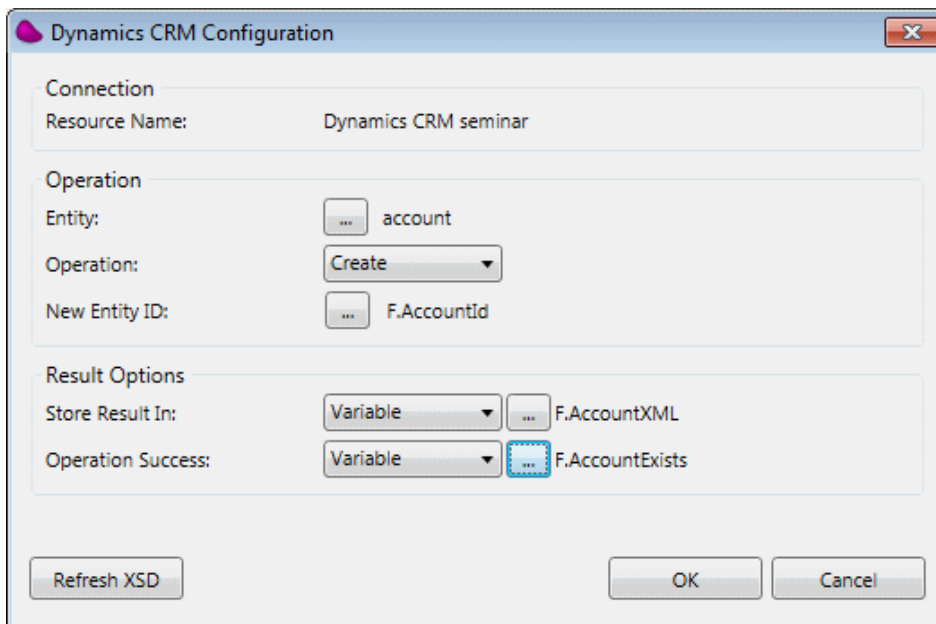
Now, you'll add an account if the account does not exist. In other words, if the **Check If Account Exists** step returns false, you'll add the account to Dynamics CRM.

1. Park on the Scan for new requests flow.
2. Add a Dynamics CRM connector as a child of the **Check If Account Exists** step, name it **Add Account**.
3. Right-click and select **Configuration**.
4. From the **Entity** property, select **account**.
5. Set the **Operation** field to **Create**.
6. From the **New Entity ID** property, select **F.AccountId**. When an object is added, Dynamics CRM returns the object ID of the newly created object into this variable.



Dynamics CRM returns the ID of the last object created. If your step is adding or updating multiple records or objects, make sure to take the IDs from the result XML.

7. From the **Store Result In** field, select the **F.AccountXML** variable.
8. From the **Operation Success** field, select the **F.AccountExists** variable.



9. Click **OK**.

You need to use the request XML that was retrieved by the Directory Scanner. This contains the customer information. Therefore, you need to have an XML as the source.

1. Add an **XML** source.
2. Go to the XML source's **Properties** pane:
3. Set the **Name** property to **RequestXML**.
4. In the **XSD File** property, select the following schema:  
**course\_data\schemas\request.xsd**
5. Set the **Data Source** to **Variable** and select the **C.RequestXML** variable.

You are now ready to map.

1. From the Source pane, open the following node: **RequestXML > CustomerDetail**.
2. From the Destination pane, open **account > row > Attributes**.
3. Connect the following nodes:

Source	Destination
AccountName	name
Street	address1_line1
City	address1_city
ZipCode	address1_postalcode
Country	address1_country

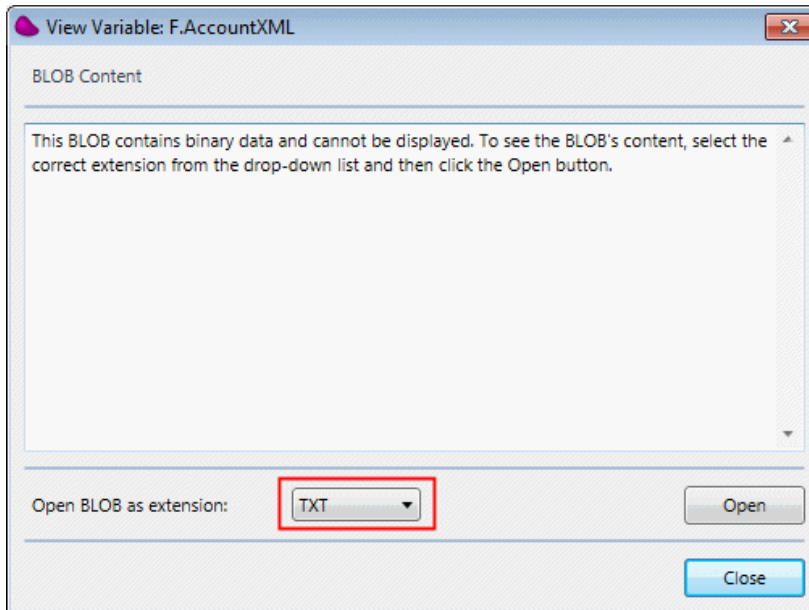
4. Click the save icon.

You only want this step to be executed if the customer does not exist; in other words, the **Check If Account Exists** step's result was unsuccessful.

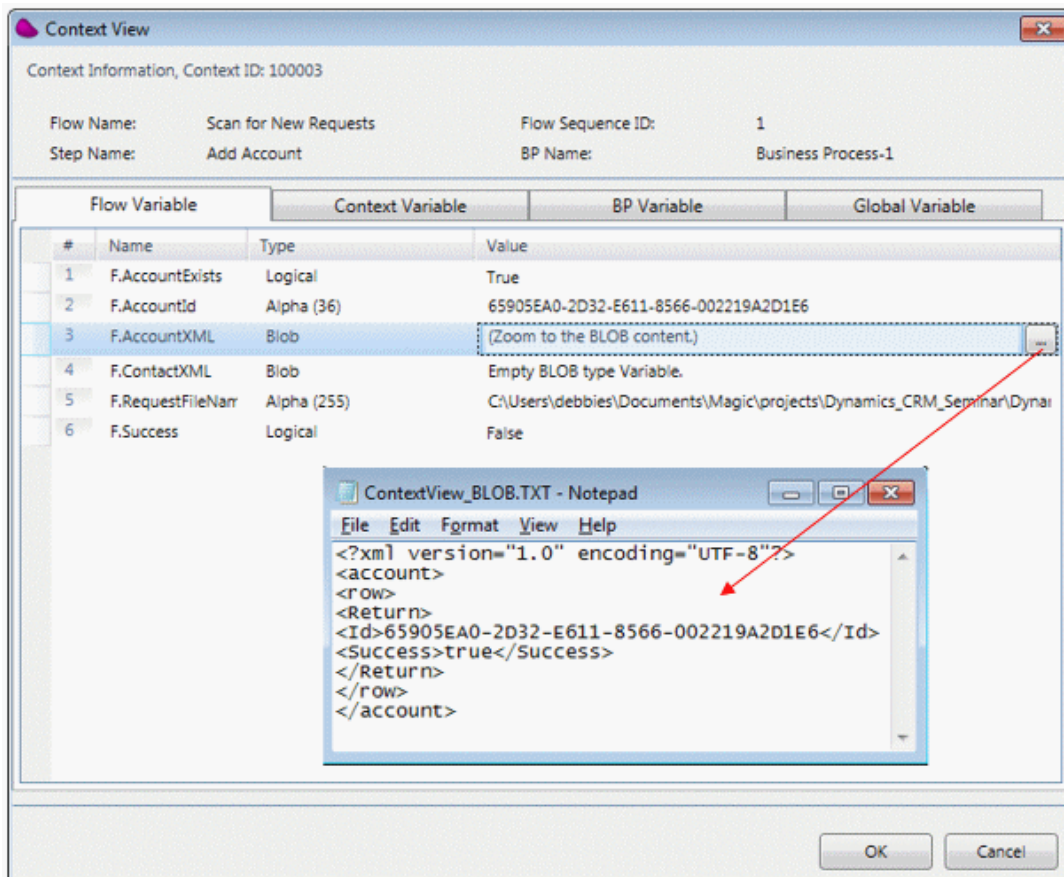
1. Park on the **Add Account** step.
2. Right-click and set the following condition: **NOT (F.AccountExists)**.
3. Once again, copy the **Result001.xml** file from the **course\_data/out** folder to the **in** folder.
4. Run the Debugger for this flow with a breakpoint on the **Add Account** step. The result for the **Create** operation is stored in the **Store result in** variable, which in the **Add Account** step is the **F.AccountXML** variable.
5. When the Debugger reaches the **Add Account** step, click the **Step** option.
6. Once the Debugger stops running, open the **Context View**.
7. Zoom from the **F.AccountXML** variable.



- In the **View Variable** dialog box, go to the **Open BLOB as extension:** field and select **TXT**.



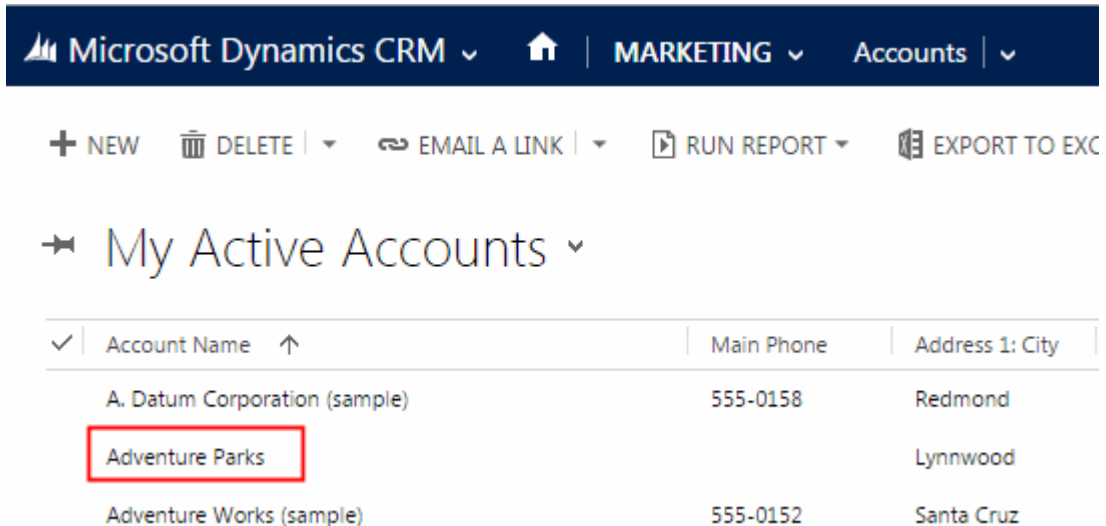
- Click **Open** and you'll see the content of the variable. For every **Create** operation, the returned XML contains a success or failure indication. In the image below, you can see that the step was successful.





If there is an error, you will see the error in the returned XML.

- In addition, open Dynamics CRM and if the process worked correctly, you should see the new account in Dynamics CRM.



Account Name	Main Phone	Address 1: City
A. Datum Corporation (sample)	555-0158	Redmond
<b>Adventure Parks</b>		Lynnwood
Adventure Works (sample)	555-0152	Santa Cruz

Although the account has been added, the contact has not yet been added.



Contacts are usually related to an account, but it is not mandatory.

## Adding a Contact

- Drop a **Dynamics CRM** connector as a child step of the **Add Account** step. Name the step **Add Contact**.
- You'll only want a contact to be added if the account exists. So, right-click on the new step and add the following condition: **F.AccountExists**.
- Right-click again and select **Configuration**.
- From the **Entity** property, select the **contact** entity.
- Set the **Operation** field to **Create**.
- Store the result in the **F.ContactXML** variable.
- Click **OK**.
- From the **Toolbox**, drag an **XML** onto the **Source** pane and name it **FetchContactFromRequest**.
- In the **XML Schema Properties** pane, go to the **XSD File** property and select the following: **course\_data\schemas\request.xsd**.
- From the **XML Root** property, select **Request**.

11. Make sure the **Source Type** property is set to **Variable** and from the **Variable** property, select the **C.RequestXML** variable.

You are now ready to map.

1. From the Source pane, open the following node: **Request > CustomerDetail**.
2. From the Destination pane, open **contact > row > Attributes**.
3. Connect **Contact\_Name** to the **firstname** and the **lastname** nodes.

In order to have the first name and last name appear together as the customer name, you'll use expressions.

4. Right-click on the **firstname** node, click the **Show Properties** option and in the **Calculated Value** field, enter the following expression:

```
StrToken (RepStr ( Trim ( Src.S1/Request/CustomerDetail/Contact_Name ), ' ', '_xpi_' )
, 1 , '_xpi_')
```

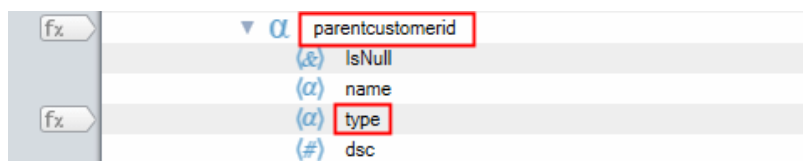
The expression first replaces the separating space with a unique separator. This is because a space cannot be a token delimiter. In our example, we've used **\_xpi\_** to ensure that the separator is unique. The expression then fetches the first token.

Now you'll do the same for the last name.

5. Right-click on the **lastname** node, click the **Properties** button and in the **Calculated Value** field, enter the following expression:

```
StrToken (RepStr ( Trim ( Src.S1/Request/CustomerDetail/Contact_Name ), ' ', '_xpi_' )
, 2 , '_xpi_')
```

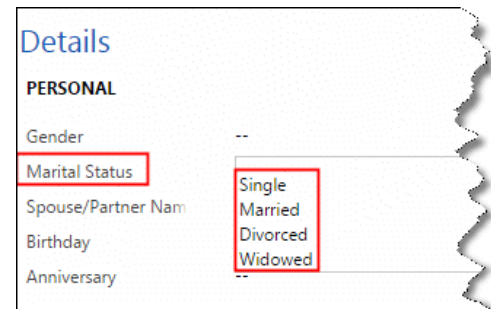
6. Connect the **E-mail\_Address** node to the **emailaddress1** node.
7. In the **Destination** pane, park on the **parentcustomerid** node and from the **Calculated Value** field, select **F. AccountId**, the ID returned by the **Add Account** step.
8. In the **parentcustomerid** node's **type** attribute, go the **Calculated Value** field and type in: **'account'**. You're telling Dynamics CRM what type of parent to assign to your contact.



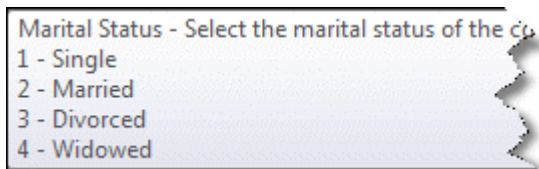
When adding a new object to Dynamics CRM from your Internet browser, a dropdown list provides a predefined list of available values.

For example, for a contact in Dynamics CRM, a dropdown list enables you to select whether the contact is Single, Married, Divorced or Widowed.

These values are provided internally by Dynamics CRM.



1. In the **Destination** pane, park on the **familystatuscode** node and go to the **Properties** pane.
2. In the **Additional XML Properties** section of the properties, hover over the **Documentation** field and you will see the available options as defined by Dynamics CRM.



3. In the **Calculated Value** property, manually enter **1** as the value.

You have now finished adding the contact.

Now you'll check the flow.

4. Run the Debugger using the **result002.xml** file with a breakpoint on the **Add Contact** step. This file contains an account and contact that does not yet exist in Dynamics CRM.
5. Look in Dynamics CRM to make sure that a new account and contact were added.



6. Also look at the contact's status. We set the **familystatuscodename** node to **Single** and you'll see in Dynamics CRM, that the **Marital Status** is set to **Single**.

## Details

### PERSONAL

Gender	--
Marital Status	<b>Single</b>
Spouse/Partner Name	--
Birthday	--
Anniversary	--



As with the **Query** operation, the Magic xpi Dynamics CRM connector saves the XML Schema, the XSD, in the following directory:

`[project dir]\[project name]\DynamicsCRM\XSD\[resource name]`

## Summary

In this lesson you learned about:

- Adding an object to the database.
- How Magic xpi enables you to add an entry to the Dynamics CRM database.
- Using entries in Dynamics CRM selection lists.

# Lesson 4

## Dealing with Products

Microsoft recommends setting up your product catalog in the following order:

- Discount lists
- Unit groups
- Price lists
- Products

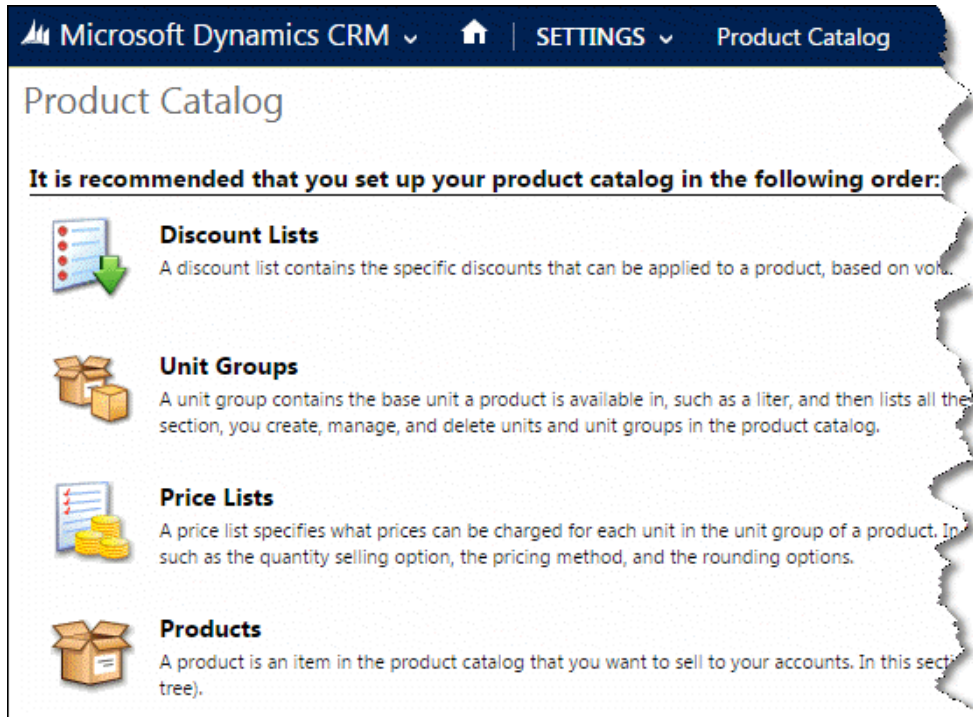
In this seminar, we'll cover the last three and then show you how to manage this all via Magic xpi.

Once you have determined that an account exists, you will then check whether the products in the result XML are valid Dynamics CRM products.

## Create Products in Dynamics CRM

The first thing we'll do here is add products directly to Dynamics CRM so that we have data to work with.

1. Go to the **Settings** menu and open the **Product Catalog**.



The screenshot shows the Microsoft Dynamics CRM interface. The top navigation bar includes 'Microsoft Dynamics CRM', a home icon, 'SETTINGS', and 'Product Catalog'. The main heading is 'Product Catalog'. Below this, a recommendation states: 'It is recommended that you set up your product catalog in the following order:'. The recommended order is listed as follows:

- Discount Lists**: A discount list contains the specific discounts that can be applied to a product, based on volume.
- Unit Groups**: A unit group contains the base unit a product is available in, such as a liter, and then lists all the units in the section, you create, manage, and delete units and unit groups in the product catalog.
- Price Lists**: A price list specifies what prices can be charged for each unit in the unit group of a product. It includes options such as the quantity selling option, the pricing method, and the rounding options.
- Products**: A product is an item in the product catalog that you want to sell to your accounts. In this section, you create, manage, and delete products (in the product tree).

2. Click on **Unit Groups** and then click **+ NEW**.
3. In the **Name** field, type in **Travel Gadgets**.
4. In the **Primary Unit** field, type in **1**.
5. Click **OK** and then **Save & Close**.
6. Back in the **Product Catalog**, click on **Price Lists** and then click **+ NEW**.
7. In the **Name** field, type in **Gadgets** and then click **Save & Close**.
8. Back in the **Product Catalog**, click on **Products**.

9. Click **+ NEW** and add two products with the following values for the mandatory fields:

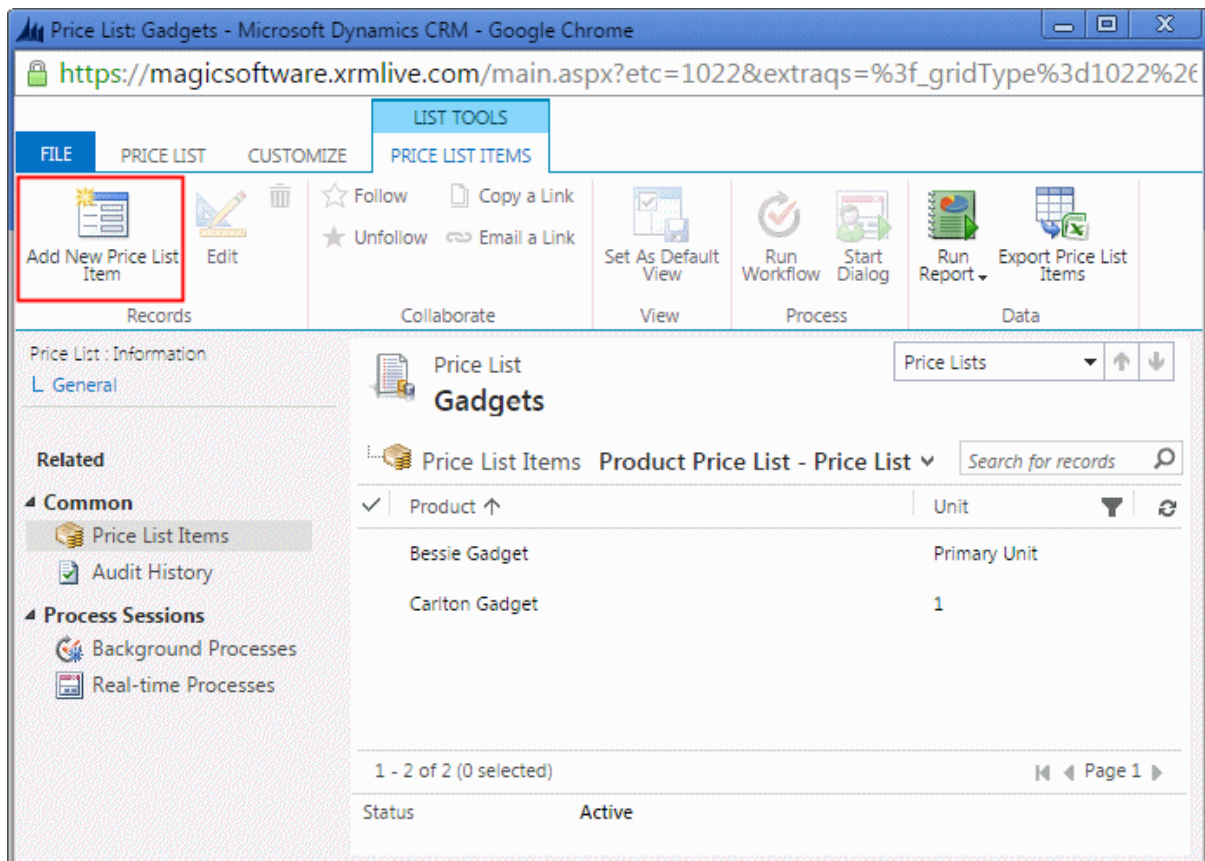
	Product #1	Product #2
ID	GAD014	GAD023
Product Name	Bessie Gadget	Carlton Gadget
Unit Group	Default Unit	Travel Gadgets
Default Unit	Primary Unit	1
Decimals Supported	0	0

Ignore the Default Price List warning message.

10. Back in the **Product Catalog**, go to the **Price Lists**.

11. In the **Gadgets** price list, click the **Add New Price List Item** option and add the two products that you just defined above with the following values:

	Product #1	Product #2
Product	Bessie Gadget	Carlton Gadget
Unit	Primary Unit	1
Amount	10	12



Price List: Gadgets - Microsoft Dynamics CRM - Google Chrome

https://magicsoftware.xrmlive.com/main.aspx?etc=1022&extraqs=%3f\_gridType%3d1022%26

FILE PRICE LIST CUSTOMIZE LIST TOOLS PRICE LIST ITEMS

Add New Price List Item Edit Follow Copy a Link Unfollow Email a Link Set As Default View Run Workflow Start Dialog Run Report Export Price List Items

Records Collaborate View Process Data

Price List: Information  
L General

Price List Gadgets

Price List Items Product Price List - Price List Search for records

Product	Unit
Bessie Gadget	Primary Unit
Carlton Gadget	1

1 - 2 of 2 (0 selected) Page 1

Status Active

## Check Whether the Products Exist

Now you'll check whether the products in the result XML are valid Dynamics CRM products. To perform this you will need a separate flow that will check each product.

1. Create a flow named **Check Products**.
2. Add the following context variable:
  - **C.All\_Products\_Exist**, a Logical variable with the default value set to 'TRUE'LOG.
3. Add the following flow variables:
  - **F.Products**, a BLOB variable. This will hold the returned data from the Dynamics CRM query.
  - **F.ProductAvailable**, a Logical variable.
  - **F.RequestedPrice**, a Numeric variable with a size of 6.2.
  - **F.ProductName**, an Alpha variable with a size of 100

Now you will query the **product** entity.

1. Drop a Dynamics CRM connector as the first step of the **Check Products** flow. Name it **Query Products**.
2. Right-click on the step and select **Configuration**.
3. In the **Entity** property, select the **product** entity.
4. Set the **Operation** to **Query**.
5. Set the **Store result in** property to **F.Products**.
6. Click **OK**.

The next stage is to map.

1. In the **Destination** pane, expand **product > row > Attributes**.
2. Park on the **name** node and in the **Calculated Value** property, zoom to the Expression Editor and type in: **Trim( F.ProductName )**.
3. Park on the **statuscode** node and in the **Calculated Value** property, type **1**, which stands for **Active**. In other words, you only want to access the products that are defined in Dynamics CRM as active.

In the next step you'll update the **F.ProductAvailable** and **C.All\_Products\_Exist** variables. The flow variable is to check each product and the context variable is to check all of the products.

1. Drop a **Flow Data** step as a child step of the **Query Products** step and name it **Update Variables**.
2. Open the **Flow Data Configuration** dialog box and click **Add**.
3. Set the **Action** property to **Update**.
4. Set the **Type** to **Flow**.



5. From the **Name** column, select the **F.ProductAvailable** variable.
6. Set the **Update Expression** to 'TRUE'LOG.
7. Set the following condition for this step: **InStr (F.Products , 'productid')>0**. Here we are checking if the **F.Products** variable contains a product ID, to determine if our product exists in Dynamics CRM.
8. Click **Add**.
9. Set the **Action** property to **Update**.
10. Set the **Type** to **Context**.
11. From the **Name** column, select the **C.All\_Products\_Exist** variable.
12. Set the **Update Expression** to 'FALSE'LOG.
13. Set the following condition for this step: **NOT (F.ProductAvailable)**. This expression means that if the product is not available, the order cannot be filled and we must set the **C.All\_Products\_Exist** variable to **False**.

If any of the previous runs of the **Check Products** flow found a product that doesn't exist, no further check should be performed. You can prevent the **Check Products** flow from running by conditioning the first step as follows:

14. Right-click on the **Query Products** step and set the following condition:  
**C.All\_Products\_Exist**.

Now you are ready to call the new flow.

1. Go back to the **Scan for New Requests** flow.
2. Drop a **Data Mapper** component as a child step of the **Check If Account Exists** step.
3. Name it **Check Products**.
4. Double click on the step. The Data Mapper screen opens.

You need to use the request XML that was retrieved by the Directory Scanner to retrieve the request products. Therefore, you need to have an XML as the source.

1. Add an **XML** entry to the **Source** pane of the Data Mapper.
2. Set the **Name** property to **RequestXML**.
3. In the **XSD File** property, select the following schema:  
**course\_data\schemas\request.xsd**
4. Set the **XML Root** property to **Request**.
5. Set the **Source Type** to **Variable** and select the **C.RequestXML** variable.

You now need to call the new flow.

1. Add a **Call Flow** entry to the **Destination** pane of the Data Mapper.
2. Set the **Name** property to **CheckProductsFlow**.
3. In the **Flow Name** property, select the **Check Products** flow.

The next stage is to map.

4. In the Data Mapper screen, connect **Product\_Name** (which you can find here: **Request > items > item**) to **F.ProductName**.

If a customer exists, the flow goes directly to the **Check Products** step. If the customer does not exist, the flow will first create the customer and then the contact. You then want the flow to move to the **Check Products** step.

5. Right-click on the **Add Contact** step, select **GoTo** and click on the **Check Products** step.

You are now ready to test.

1. For testing purposes, add a **NOP** step under the **Check Products** step.
2. Put a breakpoint on the **NOP** step and remove any other breakpoints that are set.
3. Place the **request003.xml** file in the **in** folder. This file includes one product that you defined in Dynamics CRM and one that you did not define in Dynamics CRM.
4. Run the Debugger.
5. When the Debugger stops on the **NOP** step, open the Context View. You'll see that the **C.All\_Products\_Exist** variable has a value of **False**.
6. Stop the Debugger.
7. Place the **request002.xml** file in the **in** folder.
8. Run the Debugger.
9. When the process is finished, check that the **C.All\_Products\_Exist** variable is set to **True**. If the variable returns **True**, the project is working as expected.
10. Delete the **NOP** step.

## Summary

In this lesson, you saw how to:

- Create products in Dynamics CRM, including the Unit groups, Price lists, and Products.
- Manage the products via Magic xpi.
- Check whether the products in the result XML are valid Dynamics CRM products.

# Lesson 5

## Working with Sales Orders

Check to see if the products in the request are valid, meaning that they exist. If the products are valid, add the request as a Dynamics CRM **sales order**.

The steps for the flow order in the project are:

1. Query price list
2. Get price list ID
3. Create sales order
4. Create each product

## Query Price List

Sales orders in Dynamics CRM need to be assigned to a price list.

First you'll add some variables.

1. Add a new context variable: **C.NewSO\_GUID**, an Alpha variable of size 36.
2. Add the following flow variables to the **Scan for New Requests** flow:
  - **F.SalesOrder**, a BLOB variable
  - **F.NameOfSO**, an Alpha variable of size 30
  - **F.PriceLevel**, a BLOB variable
  - **F.PriceListID**, an Alpha variable of size 36

Now you'll add a new step that will query the price list.

1. Add a **Dynamics CRM** connector as a child step of the **Check Products** step and name the step: **Query Price List**.
2. Open the **Dynamics CRM Configuration** dialog box.
3. From the **Entity** property, select **pricelevel**.
4. From the **Operation** property, select **Query**.
5. Set the **Return Attributes** property to **pricelevelid**.
6. From the **Store result in** property, select **F.PriceLevel** variable.
7. Open the Mapper.
8. In the **Destination** pane's park on the **name** node and in the **Calculated Value** property, enter: '**Gadgets**'.

## Get Price List ID

In this step, you'll be getting the ID of the default price list.

1. Add a Data Mapper step under the **Query Price List** step and call it: **Get Default PriceLevel ID**.
2. On the Source side, create an XML named **PriceLevel** and select the **pricelevel.xsd** file.
3. From the **Variable** property, select the **F.PriceLevel** variable.
4. On the Destination side, create a **Variable** destination and name it: **PriceLevelID**.
5. From the **Variables** property, select the **F.PriceListID** variable.
6. Click **OK**.
7. On the Data Mapper's **Source** side, open: **pricelevel > row > Attributes**.
8. Expand the **Destination** side.
9. Map the **pricelevelid** node to the **F.PriceListID** node.

## Create Sales Order

Now you want to create a sales order in Dynamics CRM.

1. Add a **Dynamics CRM** connector as a child step of the **Get Default PriceLevel ID** step and name the step: **Create Sales Order**.
2. Open the **Dynamics CRM Configuration** dialog box.
3. From the **Entity** property, select **salesorder**.
4. From the **Operation** property, select **Create**.
5. From the **New Entity ID** property, select **C.NewSO\_GUID**.
6. From the **Store result in** property, select the **F.SalesOrder** variable.
7. Click **OK**.

You need to connect the sales order to a specific account. The **Accountid** is part of the XML returned by the **Account Query** operation. Therefore you can use this as the source.

1. Create an **XML** source and name it **AccountInfo**.
2. From the **XSD File** property, select: **dynamicscrm\XSD\Dynamics CRM seminar\account.xsd**.
3. From the **Variable** property, select **F.AccountXML**.

You are now ready to map. In the **Data Mapper** screen:

1. In the Source pane, open: **account > row > Attributes**.
2. In the Destination pane, open: **salesorder > row > Attributes**.
3. Connect the **accountid** to the **customerid**.
4. Expand the **customerid** and in the **type** element's **Calculated Value** property, enter: **'account'**.
5. Connect the following nodes:

Source	Destination
address1_line_1	shipto_line1
address1_line_2	shipto_line2
address1_line_3	shipto_line3
address1_name	shipto_name

6. Park on the **Destination** pane's **name** node and from the **Calculated Value** property, select the **F.NameOfSO** variable.
7. Park on the **Destination** pane's **pricelevelid** node and set the **Calculated Value** property to the **F.PriceListID** variable.
8. Go to the **F.NameOfSO** variable (in the Flow Variables repository) and in the **Default Value** column, type in a name, such as: **Seminar Sales Order**. This will be the name of the sales order in Dynamics CRM.

You now want to add the products to the sales order. You first have to retrieve the products.

1. Create a new flow and name it: **Sales Order Products**.
2. Add the following flow variables:
  - **F.SODetail\_GUID**, Alpha 36
  - **F.SODetail**, Blob
  - **F.Success**, Logical 1
  - **F.ProductName**, Alpha 100
  - **F.Product**, Blob
  - **F.Qty**, Numeric 8.2
  - **F.PriceList**, Blob
3. Drag a Dynamics CRM connector onto the flow and name it: **Get Product Info**.
4. Open the **Dynamics CRM Configuration** dialog box and from the **Entity** property, select the **product** entity.
5. Set the **Operation** to **Query**.
6. Set the **Store Result In** property to **F.Product**.
7. Click **OK**.
8. From the **name** node's **Calculated Value** property, select the **F.ProductName** variable.

You will now add the products to the sales order.

1. Drag another Dynamics CRM connector under the **Get Product Info** step and name it: **Add Product to Sales Order**.
2. Double click on the step and from the **Entity** property, select the **salesorderdetail** entity. This represents a line item in a sales order in Dynamics CRM.
3. Set the **Operation** to **Create**.
4. From the **New Entity ID** property, select the **F.SODetail\_GUID** variable.
5. From the **Store Result In** property, select the **F.SODetail** variable.
6. From the **Operation Success** property, select the **F.Success** variable.
7. Click **OK**.
8. In the Data Mapper screen, create an XML source and name it: **Product**.
9. From the **XSD File** property, select: **dynamicscrm\XSD\Dynamics CRM seminar\product.xsd**.
10. From the **Variable** property, select the **F.Product** variable.
11. Map the following nodes:
  - **defaultuomid** to **uomid**
  - **price** to **priceperunit**
  - **productid** to **productid**
12. On the destination side, from the **quantity** node's **Calculated Value** property, select the **F.Qty** variable.
13. From the **salesorderid** node's **Calculated Value** property, select the **C.NewSO\_GUID** variable.

15. Right-click on the **Add Product to Sales Order** step and add the following condition:  
**InStr (F.Product,'productid') > 0**. Here we are checking if the **F.Product** variable contains a product ID, to determine if our product exists in Dynamics CRM.

## Create Each Product

Now you'll call this flow from the main flow.

1. Go back to the **Scan for requests** flow.
2. Add a Data Mapper step under the **Create Sales Order** step and call it: **Create Each Product**.
3. On the Source side, create an XML named **RequestXML** and select the **request.xsd** file.
4. Set the **XML Root** property to **Request**.
5. Set the **Variable** property to **C.RequestXML**.
6. On the Destination side, create a **Call Flow** destination and set the **Name** property to: **SOProducts**.
7. From the **Flow Name** property, select the **Sales Order Products** flow.
8. From the Data Mapper's Source side, open: **RequestXML > Request > items > item**.
9. Expand the Destination side.
10. Map the **Product\_Name** node to the **F.ProductName** node.
11. Map the **Qty** node to the **F.Qty** node.
12. Add a NOP service after the **Create Each Product** step for debugging purposes.

You are ready to test.

1. Once again, place the **request002.xml** file in the **in** folder.
2. Set a breakpoint on the NOP step.
3. Debug the project.
4. When the process is finished, you should see a screen in Dynamics CRM similar to the one below.

+ NEW 🗑️ DELETE 📄 CREATE INVOICE 📦 FULFILL ORDER 🚫 CANCEL ORDER 🧮 RECALCULATE 👤 GET PRODUCTS ⋮

ORDER

## Seminar Sales Order

**Summary**

Order ID *	🔒 <b>ORD-01035-R7H6Y7</b>	<b>PRODUCTS</b>				🔒
Name *	<b>Seminar Sales Order</b>	Product Name	Price Per Unit	Quantity	Discount	Extended Amount
Currency *	🔒 <b>US Dollar</b>	🔒 <b>Bessie Gadget</b>	🔒 <b>\$10.00</b>	<b>1.00000</b>	<b>\$0.00</b>	🔒 <b>\$10.00</b>
Price List *	<b>Gadgets</b>	🔒 <b>Carlton Gadget</b>	🔒 <b>\$12.00</b>	<b>3.00000</b>	<b>\$0.00</b>	🔒 <b>\$36.00</b>
Prices Locked *	🔒 <b>No</b>					



Dynamics CRM automatically assigns an order ID when an order is created. In our example above, it is: **ORD-01035-R7H6Y7**.

## Summary

In this lesson, you:

- Learned how to add an object to the Dynamics CRM database.
- Added an account and a contact for that account.
- Added a new sales order and its details