Hands-On Lab

CRM Online from PHP

Lab version: 1.0.0

Last updated: 5/11/2011

* 1. 
  2. Contents

[Overview 3](#_Toc292905423)

[Exercise 1: Setting up the PHP project and importing the required libraries 4](#_Toc292905424)

[Task 1 - Create a new Netbeans Project 4](#_Toc292905425)

[Exercise 1 Verification 8](#_Toc292905426)

[Exercise 2: Setting up the environment to Integrate with CRM Online 10](#_Toc292905427)

[Task 1 - Open the solution you created in Exercise 1 in Netbeans 10](#_Toc292905428)

[Task 2 – Add the necessary code to Main.php to connect get WLID tokens 10](#_Toc292905429)

[Task 5 – Token exchange with Windows Live ID 11](#_Toc292905430)

[Exercise 2 Verification 13](#_Toc292905431)

[Exercise 3: Using PHP Libraries to Perform CRUD operations against CRM Online 14](#_Toc292905432)

[Task 1 – Create the functions for Create, Read, Update and Delete 14](#_Toc292905433)

[Task 2 – Call the CRUD functions from Main.php 19](#_Toc292905434)

[Exercise 3 Verification 19](#_Toc292905435)

[Summary 20](#_Toc292905436)

Overview

* 1. Dynamics CRM 2011 has a well published CRUD data and metadata API accessible through WCF and SOAP interfaces. These interfaces are language neutral and accessible from many different programming languages. This makes ISVs that are working on disparate platforms to be able to connect and exchange information with Dynamics CRM securely to integrate their solutions with Dynamics CRM.

# Scenario

* 1. Contoso is an independent software vendor (ISV) that provides data enrichment solutions to companies and already has a server side infrastructure using PHP. Contoso would like to Integrate with Dynamics CRM 2011 to bring their services to millions of users on Dynamics CRM.
  2. Contoso’s data enrichment solution is a server-based application that reaches out into Customer CRM systems and adds missing data to the account records. We have been asked to build an integration from the Contoso server to Microsoft Dynamics CRM Online so that Contoso can create, read, update and delete an account that may be of interest to their customer based on the profile that the customer has entered with Contoso.

# Objectives

* 1. After completing this lab you will have accomplished the following objectives:
  + Understand how to build a PHP application to access CRM Online
  + Understand how to perform CRUD operations using PHP against CRM Online

# System Requirements

* 1. You must have the following items to complete this lab:
  + PHP 5.3.4 or higher
  + Netbeans 7.0 with support for PHP
  + Dynamics CRM 2011 Online Instance

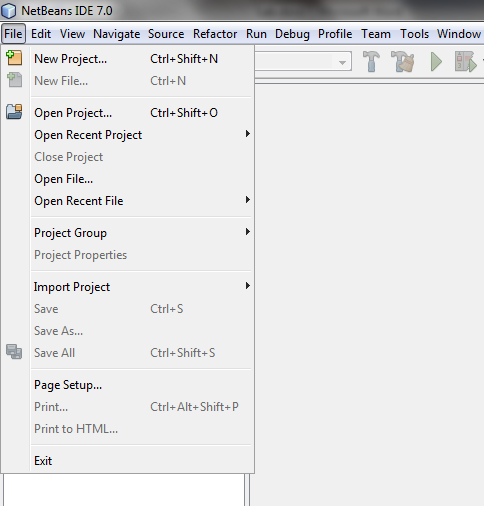
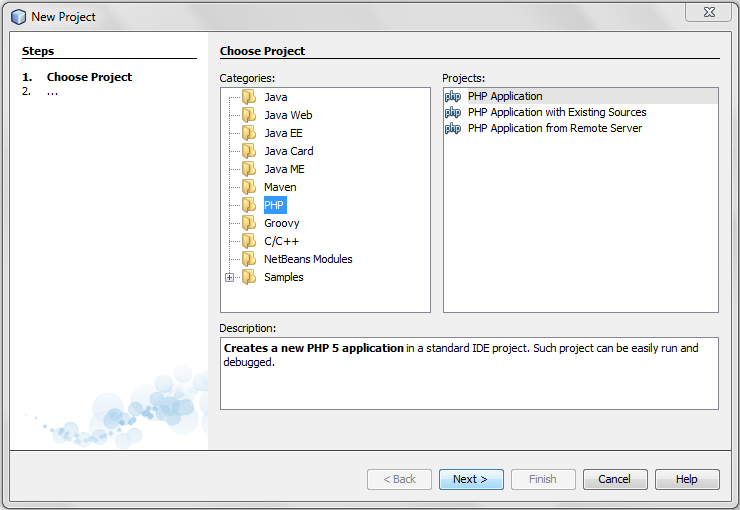
# Exercises

* 1. This Hands-On Lab comprises the following exercises:
  2. Setting up the PHP project and importing the required libraries
  3. Setting up the environment to Integrate with CRM Online
  4. Using Soap calls to Perform CRUD operations against CRM Online
  5. Estimated time to complete this lab: **30 minutes**.

Exercise 1: Setting up the PHP project and importing the required libraries

1. Dynamics CRM Online has a very good and well published API that can be accessed using SOAP. We have included a library of PHP classes that will help to make SOAP calls to CRM Online. FetchXML will be used to do the CRUD Operations.

Task 1 - Create a new Netbeans Project

* 1. In this task, you will setup a new Netbeans Project to integrate with CRM Online. If you have built a Netbeans application before much of this task should seem familiar.
  2. Start Netbeans from the Operating System launcher. On Windows from **Start | All Programs | Netbeans | Netbeans IDE 7.0**
  3. Click **File | New | Project** to launch the New Project dialog.
  4. Select PHP Application from the dialog that opens.
     1. 
     2. 
     3. Figure 1
        1. Create a new PHP Application
  5. In the Project Name input area type PHP2CRMOnline and type **C:\Projects\PHP2CRMOnline** in the Sources Folder input area.
     1. **Note:** Create the folders C:\Projects and C:\Projects\PHP2CRMOnline beforehand.
  6. Set the PHP Version to be PHP 5.3 and default encoding to be UTF-8.

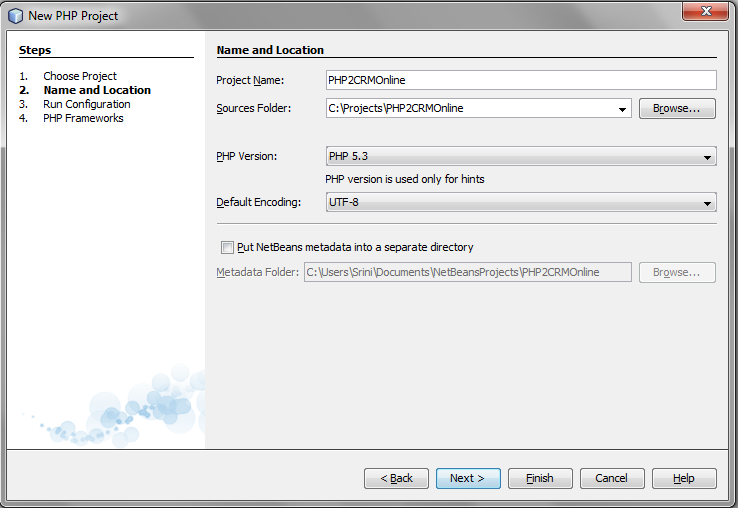


Figure 2

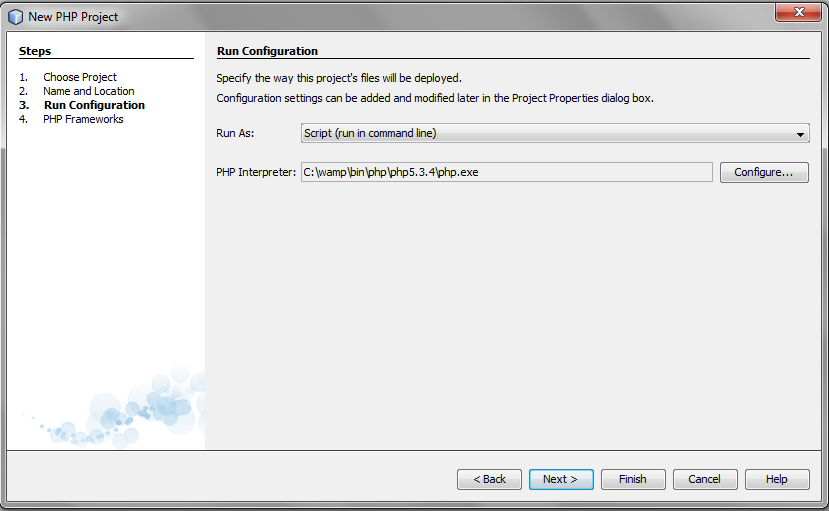
* + - 1. New PHP Project Options
  1. Click **Next** to give more information about your project.
  2. Netbeans will now ask you for additional project information, choose the **Run As:** option as Script (Run in a command line) and let the PHP Interpreter point to the correct location of your PHP install and click **Next**. 

Figure 3

Additional PHP Project Options

* 1. You will be shown two PHP framework options, do not select any of them and click Finish. This will create your PHP Project.

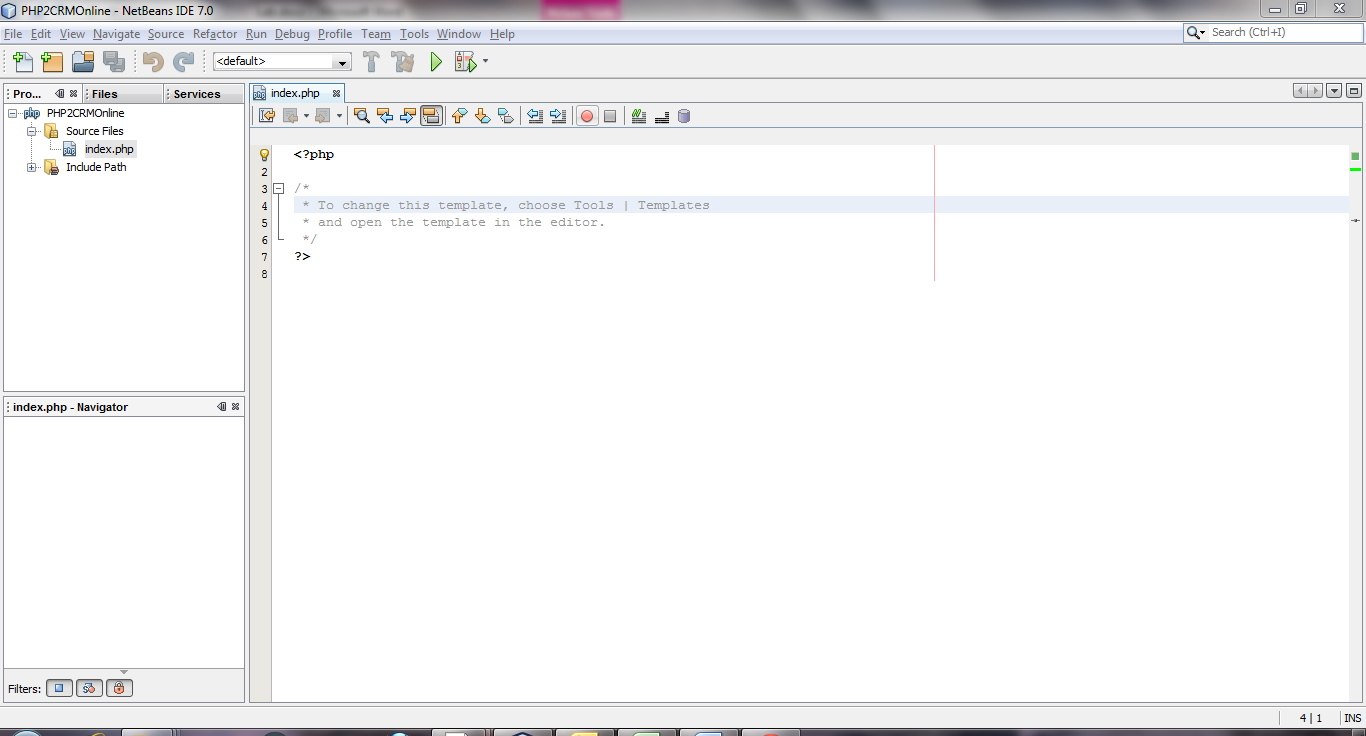


Figure 4

* + - 1. Blank Netbeans PHP Application with index.php
  1. Now that the project has been generated, the first thing we need to do is to add the required libraries that the project shall depend on.
  2. Open the “Setup” folder in Windows Explorer. [ The setup folder is provided as part of this lab ]
  3. Copy all the PHP files found in the library folder to c:\Projects\ PHP2CRMOnline
  4. Rename index.php to Main.php.

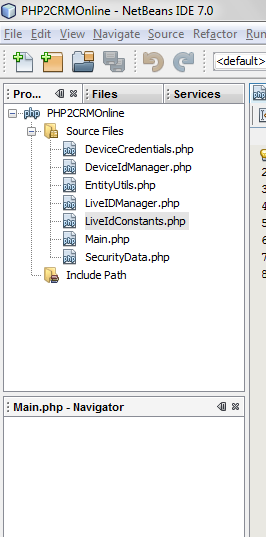
Exercise 1 Verification

* 1. In order to verify that you have correctly performed all steps of Exercise 1, proceed as follows:

#### Verification 1

In this verification, you will ensure that the project is setup and that the library PHP classes have successfully been added to the project.

* 1. Navigate to the PHP2CRMOnline project in Netbeans
  2. In the Project Navigator, Left-Click to expand the project name PHP2CRMOnline.
  3. Check for the existence of two folders called as Source Files and Include Path.
  4. Expand the Source Files folder to confirm the following files exist.
     1. DeviceCredentials.php
     2. DeviceIdManager.php
     3. EntityUtils.php
     4. LiveIDManager.php
     5. LiveIdConstants.php
     6. SecurityData.php
  5. Ensure that the file Main.php also exists and that index.php do not exist.



* + 1. Figure 5
       1. List of PHP Library files included in the project
    2. **Note:** The application is not functional yet, as we have not added any code to Main.php.

Exercise 2: Setting up the environment to Integrate with CRM Online

**Important:** In this Exercise, we will work with the solution that we started building in Exercise 1

1. Microsoft Dynamics CRM Online uses Windows Live ID as the authentication mechanism and for our integration to work we need to perform the authentication against Windows Live ID and retrieve the tokens that will later be presented to CRM Online. In this exercise you will use the PHP Library files we copied in Exercise 1 for device id generation and the Windows Live ID token exchange for you. For this you will need the windows live ID credentials for the CRM Online system handy.

Task 1 - Open the solution you created in Exercise 1 in Netbeans

* 1. In this task, you are going to open the solution you created in Exercise 1. This exercise is dependent on Exercise 1 so you must have completed Exercise 1 to proceed.

Task 2 – Add the necessary code to Main.php to connect get WLID tokens

* 1. In this task you will add a code snippet to the Main.php to perform the token exchange with Windows Live ID.
  2. Navigate to the Projects view and expand on the PHP2CRMOnline project.
  3. Open Source Files folder and double click on Main.php and it should open in the code view window on the right.
  4. Add the following include statements. These include statements will enable us to call the appropriate libraries needed to get WLID tokens
     1. PHP
     2. include\_once "LiveIdManager.php";
     3. include\_once "EntityUtils.php";
  5. *LiveIdManager.php* is the utility class for doing authentication against Windows Live. Entity *Utils.php* is needed to do all calls to Dynamics CRM entities.
  6. This completes the project setup required to make the integration work.

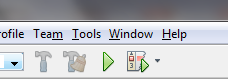
Task 5 – Token exchange with Windows Live ID

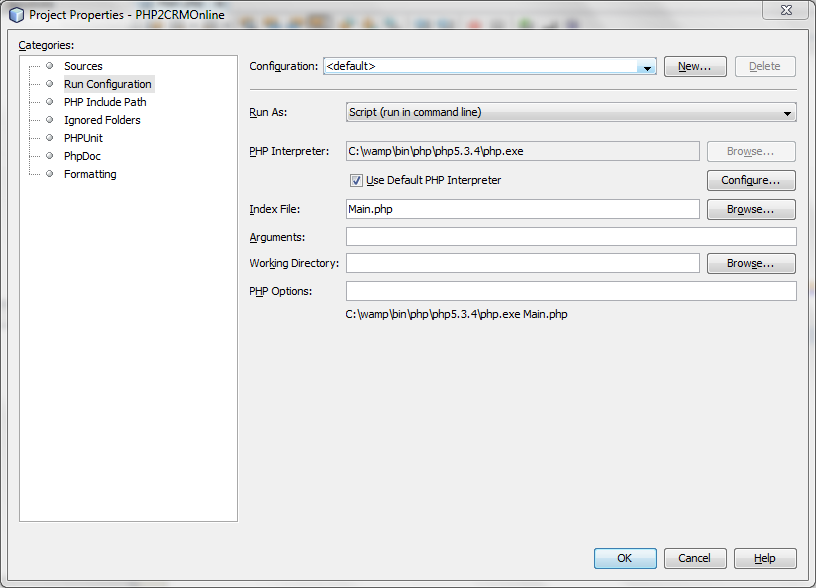
* 1. In this task, you will add the necessary code to perform a token exchange with Windows Live ID. This is the same token set that shall be presented to CRM Online
  2. Navigate to *Main.php.*
  3. Insert the following code to sign onto Windows Live ID and get the keys and tokens.
     + 1. PHP
     1. $liveIDUseranme = "yourCRMliveIDUsername";
     2. $liveIDPassword = "yourCRMliveIDPassword";
     3. $organizationServiceURL = "https://yourcrmorg.crm.dynamics.com/XRMServices/2011/Organization.svc";

1. $liveIDManager = new LiveIDManager();
2. $securityData =

$liveIDManager->authenticateWithLiveID( $organizationServiceURL, $liveIDUseranme, $liveIDPassword );

1. //Print out the token received from WLID
2. if($securityData!=null && isset($securityData)){
3. echo ("\nKey Identifier:" . $securityData->getKeyIdentifier());
4. echo ("\nSecurity Token 1:" . $securityData->getSecurityToken0());
5. echo ("\nSecurity Token 2:" . $securityData->getSecurityToken1());
6. }else{
7. echo "Unable to authenticate LiveId.";
8. return;
9. }
   1. To run Main.php, click on the green play button in Netbeans.



* 1. In the Project Properties window, change the **Index File** option to Main.php
     1. 
     2. Figure 6
        1. List of PHP Library files included in the project

Exercise 2 Verification

* 1. In order to verify that you have correctly performed all steps of Exercise 2, proceed as follows:

#### Verification 1

In this verification, you will confirm that the tokens are being received from Windows Live ID.

* 1. Run the application
  2. Look at the debug window in Netbeans.
  3. Confirm that you see three strings; these are the token strings from Windows Live ID.

Exercise 3: Using PHP Libraries to Perform CRUD operations against CRM Online

1. **Important:** In this Exercise navigate to <https://yourcrmorg.crm.dynamics.com>. If the CRM instance you are connecting to is in EMEA or APAC the URL you use to access it shall be <https://yourcrmorg.crm4.dynamics.com> or <https://yourcrmorg.crm5.dynamics.com>
2. In this exercise you will create the functions necessary to perform CRUD operations on CRM Online.

Task 1 – Create the functions for Create, Read, Update and Delete

* 1. In this task, you will create the functions necessary to perform the CRUD functions against CRM Online.
  2. Navigate to the Netbeans project and open Main.php.
  3. All the functions we create in this exercise will be associated with the ACCOUNT entity.
  4. We will start with the Create Account with PHP. We create a function called *createAccount* by passing the OrganizationUrl and SecurityData we obtained after LiveId authentication.
  5. Insert the following code for *createAccount* into the Main.php
     + 1. PHP
     1. function createAccount($CRMURL,$securityData) {
     2. $domainname = substr($CRMURL,8,-1);
     3. $pos = strpos($domainname, "/");
     4. $domainname = substr($domainname,0,$pos);
     6. $accountsRequest = EntityUtils::getCreateCRMSoapHeader($CRMURL, $securityData).
     7. '
     8. <s:Body>
     9. <Create xmlns="http://schemas.microsoft.com/xrm/2011/Contracts/Services">
     10. <entity xmlns:b="http://schemas.microsoft.com/xrm/2011/Contracts" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
     11. <b:Attributes xmlns:c="http://schemas.datacontract.org/2004/07/System.Collections.Generic">
     12. <b:KeyValuePairOfstringanyType>
     13. <c:key>name</c:key>
     14. <c:value i:type="d:string" xmlns:d="http://www.w3.org/2001/XMLSchema">Newer Corporation</c:value>
     15. </b:KeyValuePairOfstringanyType>
     16. </b:Attributes>
     17. <b:EntityState i:nil="true"/>
     18. <b:FormattedValues xmlns:c="http://schemas.datacontract.org/2004/07/System.Collections.Generic"/>
     19. <b:Id>00000000-0000-0000-0000-000000000000</b:Id>
     20. <b:LogicalName>account</b:LogicalName>
     21. <b:RelatedEntities xmlns:c="http://schemas.datacontract.org/2004/07/System.Collections.Generic"/>
     22. </entity>
     23. </Create>
     24. </s:Body>
     25. </s:Envelope>
     26. ';
     27. $response = LiveIDManager::GetSOAPResponse("/Organization.svc", $domainname, $CRMURL, $accountsRequest);
     29. $createResult ="";
     30. if($response!=null && $response!=""){
     31. preg\_match('/<CreateResult>(.\*)<\/CreateResult>/', $response, $matches);
     32. $createResult = $matches[1];
     33. }
     35. return $createResult;
     36. }
  6. The *createAcount* function calls *LiveIdManager* with the security tokens to create a SOAP header for our call. Once we get the SOAP header, we use Dynamics CRM’s SOAP calls to create our account.
  7. In the above function we are creating an account with the name “Newer Corporation”.
  8. We get a SOAP response for the createAccount which also holds the new account’s accountId.
  9. Next we write the *readAccount* function to retrieve account data from Dynamics CRM.
  10. The following function uses FetchXML to retrieve the account information. We pass the accountId for Newer Corporation and the OrganizationUrl, SecurityData to retrieve the data.
      + 1. PHP
      1. function readAccount($accountId,$CRMURL,$securityData){
      3. $domainname = substr($CRMURL,8,-1);
      5. $pos = strpos($domainname, "/");
      6. $domainname = substr($domainname,0,$pos);
      8. $accountsRequest = EntityUtils::getCRMSoapHeader($CRMURL, $securityData) .
      9. '
      10. <s:Body>
      11. <Execute xmlns="http://schemas.microsoft.com/xrm/2011/Contracts/Services">
      12. <request i:type="b:RetrieveMultipleRequest" xmlns:b="http://schemas.microsoft.com/xrm/2011/Contracts" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
      13. <b:Parameters xmlns:c="http://schemas.datacontract.org/2004/07/System.Collections.Generic">
      14. <b:KeyValuePairOfstringanyType>
      15. <c:key>Query</c:key>
      16. <c:value i:type="b:FetchExpression">
      17. <b:Query>&lt;fetch mapping="logical" count="50" version="1.0"&gt;&#xD;
      18. &lt;entity name="account"&gt;&#xD;
      19. &lt;attribute name="name" /&gt;&#xD;
      20. &lt;attribute name="address1\_city" /&gt;&#xD;
      21. &lt;attribute name="telephone1" /&gt;&#xD;
      22. &lt;filter type="and"&gt;
      23. &lt;condition attribute="accountid" operator="eq" value="'.$accountId.'" /&gt;
      24. &lt;/filter&gt;
      25. &lt;/entity&gt;&#xD;
      26. &lt;/fetch&gt;
      27. </b:Query>
      28. </c:value>
      29. </b:KeyValuePairOfstringanyType>
      30. </b:Parameters>
      31. <b:RequestId i:nil="true"/><b:RequestName>RetrieveMultiple</b:RequestName>
      32. </request>
      33. </Execute>
      34. </s:Body>
      35. </s:Envelope>
      36. ';
      37. $response = LiveIDManager::GetSOAPResponse("/Organization.svc", $domainname, $CRMURL, $accountsRequest);
      38. $accountsArray = array();
      39. if($response!=null && $response!=""){
      41. $responsedom = new DomDocument();
      42. $responsedom->loadXML($response);
      43. $entities = $responsedom->getElementsbyTagName("Entity");
      44. foreach($entities as $entity){
      45. $account = array();
      46. $kvptypes = $entity->getElementsbyTagName("KeyValuePairOfstringanyType");
      47. foreach($kvptypes as $kvp){
      48. $key = $kvp->getElementsbyTagName("key")->item(0)->textContent;
      49. $value = $kvp->getElementsbyTagName("value")->item(0)->textContent;
      50. if($key == 'accountid'){ $account['accountId'] = $value; }
      51. if($key == 'name'){ $account['name'] = $value; }
      52. if($key == 'telephone1'){ $account['telephone'] = $value; }
      53. if($key == 'address1\_city'){ $account['address'] = $value; }
      54. }
      55. $accountsArray[] = $account;
      56. }
      57. }
      58. return $accountsArray;
      59. }
  11. In the next function we will update the name of the account we created to “Updated Newer”
  12. Insert the code below in the Main.php.
      + 1. PHP
      1. function updateAccount($accountId,$CRMURL,$securityData) {
      3. $domainname = substr($CRMURL,8,-1);
      5. $pos = strpos($domainname, "/");
      6. $domainname = substr($domainname,0,$pos);
      8. $accountsRequest = EntityUtils::getUpdateCRMSoapHeader($CRMURL, $securityData).
      9. '<s:Body><Update xmlns="http://schemas.microsoft.com/xrm/2011/Contracts/Services">
      10. <entity xmlns:b="http://schemas.microsoft.com/xrm/2011/Contracts" xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
      11. <b:Attributes xmlns:c="http://schemas.datacontract.org/2004/07/System.Collections.Generic">
      12. <b:KeyValuePairOfstringanyType>
      13. <c:key>name</c:key>
      14. <c:value i:type="d:string" xmlns:d="http://www.w3.org/2001/XMLSchema">Updated Newer</c:value>
      15. </b:KeyValuePairOfstringanyType>
      16. </b:Attributes>
      17. <b:EntityState i:nil="true"/>
      18. <b:FormattedValues xmlns:c="http://schemas.datacontract.org/2004/07/System.Collections.Generic"/>
      19. <b:Id>'.$accountId.'</b:Id>
      20. <b:LogicalName>account</b:LogicalName>
      21. <b:RelatedEntities xmlns:c="http://schemas.datacontract.org/2004/07/System.Collections.Generic"/>
      22. </entity></Update>
      23. </s:Body>
      24. </s:Envelope>';
      26. $response = LiveIDManager::GetSOAPResponse("/Organization.svc", $domainname, $CRMURL, $accountsRequest);
      28. return $response;
      29. }
  13. Next, we will write a function “deleteAccount”. This will be used to delete the account.
      + 1. PHP
      1. function deleteAccount($accountId,$CRMURL,$securityData) {
      3. $domainname = substr($CRMURL,8,-1);
      4. $pos = strpos($domainname, "/");
      5. $domainname = substr($domainname,0,$pos);
      6. $accountsRequest = EntityUtils::getDeleteCRMSoapHeader($CRMURL, $securityData).
      7. '<s:Body>
      8. <Delete xmlns="http://schemas.microsoft.com/xrm/2011/Contracts/Services">
      9. <entityName>account</entityName>
      10. <id>'.$accountId.'</id>
      11. </Delete>
      12. </s:Body>
      13. </s:Envelope>';
      14. $response = LiveIDManager::GetSOAPResponse("/Organization.svc", $domainname, $CRMURL, $accountsRequest);
      15. }

Task 2 – Call the CRUD functions from Main.php

* 1. In this task, you will invoke the CRUD functions from Main.php.
  2. Navigate to Main.php and find the echo calls we added in Exercise 3.
  3. Insert the code below after the echo function calls.
     + 1. PHP
     1. $accountId = createAccount($organizationServiceURL, $securityData);
     2. print\_r(readAccount($accountId, $organizationServiceURL, $securityData));
     3. updateAccount($accountId, $organizationServiceURL, $securityData);
     4. print\_r(readAccount($accountId, $organizationServiceURL, $securityData));
     5. //deleteAccount($accountId, $organizationServiceURL, $securityData);
  4. You will notice that the deleteAccount function is commented out for the verification step. This is deliberately done so that the verification can be done using CRM Online.
  5. From the Run Menu select ***Run Project(PHP2CRMOnline)***.

Exercise 3 Verification

* 1. In order to verify that you have correctly performed all steps of Exercise 4, proceed as follows:

#### Verification 1

In this verification, you will confirm that the integration with CRM Online is working.

* 1. Run the application
  2. Navigate to <https://yourcrmorg.crm.dynamics.com>
  3. Navigate over to the Accounts list and click on all accounts.
  4. Confirm that **Updated Newer** exists as an account.

Summary

* 1. In this lab you have experienced using PHP to connect to Microsoft Dynamics CRM Online. If you organization uses PHP as its server side technology, now you can use a similar method to connect with CRM Online from your organization.