



Integrating Salesforce with SharePoint 2007 via the Business Data Catalog

SalesForce CRM is a popular tool that allows you to manage your Customer Relation Management in the cloud through a web based system. This is particularly useful for a company that does not have a huge sales team - but needs all the functionality of a CRM system without bringing huge costs.

This can quite easily become another island of data issue. People working in your accounts department may need to tag documents with customer metadata. With the business data catalog this is easy to do as long as you can integrate successfully with the Salesforce - Force.com web services. This document will walk you through an example of how we have done this, so you can use our solution or base yours from it.

Force.com web services

SalesForce CRM offers a web service that you can use to pull your CRM data from and even push data back to. As you may know from working with the Business Data Catalog, it is much easier to integrate a web service with the BDC if the data is presented in a way that mimics the BDC methods such as Finder, SpecificFinder and IdEnumerator. To help us integrate with the BDC we will write a custom web service wrapper around the Salesforce web services.

By default the API/web service access is only available in Enterprise or Unlimited editions of Salesforce, so if you want to integrate with the BDC you'll need to either upgrade or see if you can get the API access bolted on.

This example will walk through using a Developer account with some sample Salesforce data. Once you have this up and running you can look at how to get the web service calling your live Salesforce data. You can sign up for your developer account at <http://developer.force.com/join>

First thing we need to do is get access to the Salesforce wsdl file so we can generate the necessary proxy classes and objects. Do this by

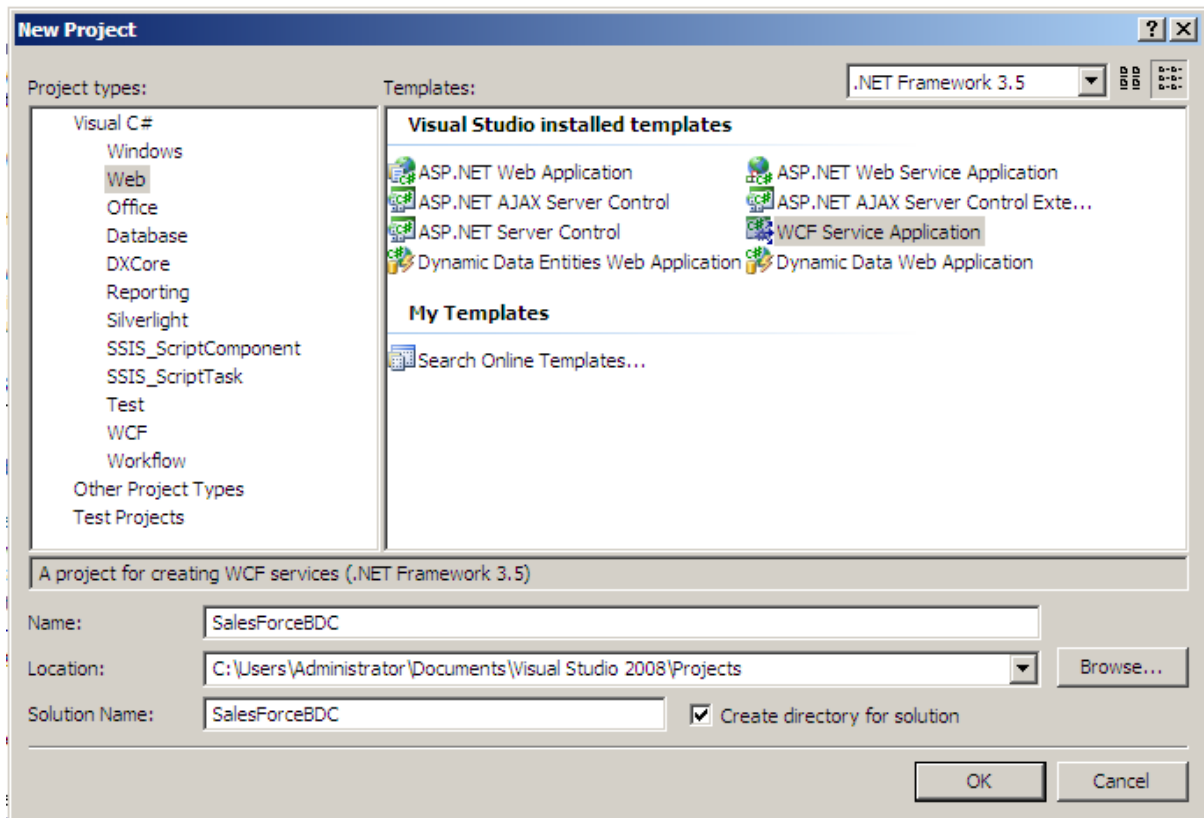
- 1, logging on via <http://developer.force.com>
- 2, Click on Setup in the top right hand corner
- 3, Expand the Developer node in the left hand menu
- 4, Click on the API link

We are now presented with a number of different options to download. As we are looking to integrate with our own companies Salesforce data we can select to 'Generate Enterprise WSDL', right click on the link and choose to save the resulting file as Enterprise.wsdl.xml.

We can now create a Visual Studio project to make use of this wsdl file.

5, Open up Visual Studio 2008 and go File -> New -> Project

6, From the project types select Web, and WCF Service Application from the templates window. Give the project a name such as SalesforceBDC



7, Visual Studio 2008 will now go ahead and create the necessary project structure you need to use. It also creates some default files for you such as IService1.cs and Service1.svc. Lets rename these files to just IService.cs and Service.svc.

8, Also we have Service1 class name remaining in Service.svc.cs. Double click on the file and rename it:

```
] namespace SalesforceBDC
{
    // NOTE: If you change the class name "I
] public class Service : IService
{
}
```

9, As you'll see there is a notice that if you change this class name you'll also need to change the reference in web.config. So double click on web.config and scroll to the end of the file to the `<system.serviceModel>` section. We simply need to update the

```
<service behaviorConfiguration="SalesForceBDC.Service1Behavior"
name="SalesForceBDC.Service1">
```

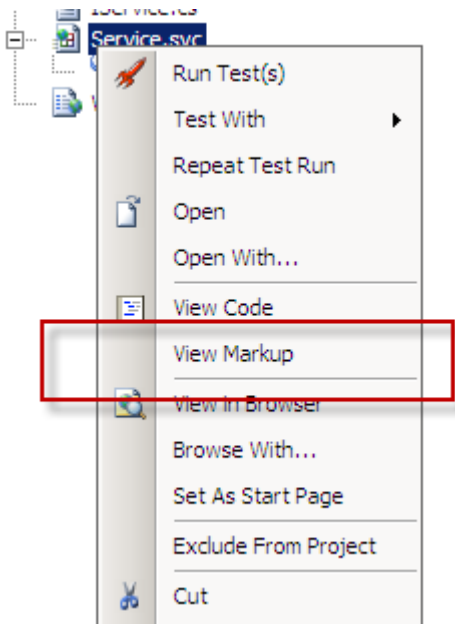
line to

```
<service behaviorConfiguration="SalesForceBDC.Service1Behavior"
name="SalesForceBDC.Service">
```

Also we need to change the binding of the WCF service. The Business Data Catalog only supports `basicHttpBinding` so you need to add this in rather than `wsHttpBinding`.

```
<endpoint address="" binding="basicHttpBinding"
contract="SalesForceBDC.IService">
```

10, Finally (for the renaming at least) you'll need to update Service.svc to reference the correct class. In the solution explorer right click on Service.svc and choose 'View Markup'



Then edit

```
Service="SalesForceBDC.Service1"
```

to

```
Service="SalesForceBDC.Service"
```

11, Now we can check that we have updated all the references correctly. Press F5 to build and run your solution. Click Yes to add the Debug attribute to web.config, and then finally the browser that opens will display your svc page. If you get an error you'll need to go back through and check you have updated the Service1's to Service.

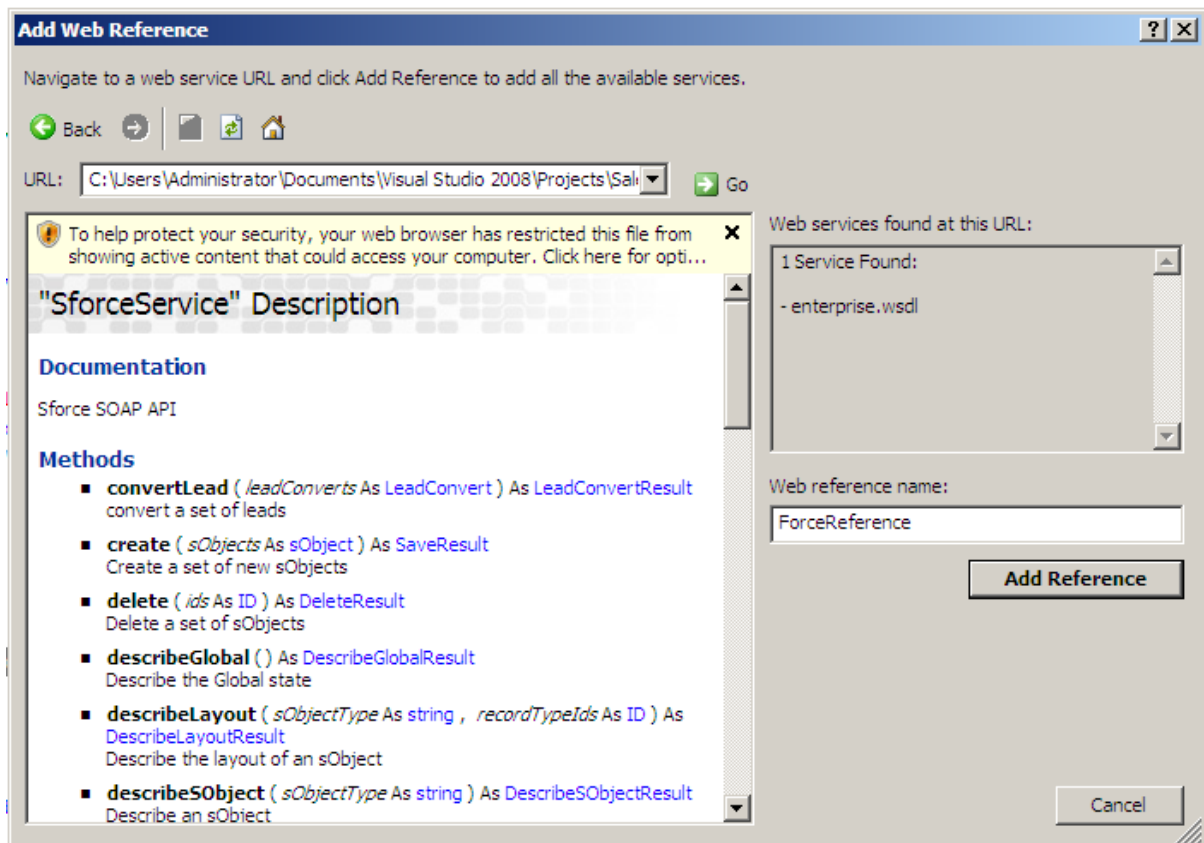
12, Now we can get on with doing some interesting stuff! Firstly we need to add our SaleForce wsdl file we generated in steps 1 -4. Right click on your Visual Studio project and choose 'Add Existing Item'. Browse to where you saved the enterprise.wsdl.xml file and select and add it.

13, We can now use this wsdl.xml file to generate the web service proxy classes we use to talk to the Salesforce.com web services. To do this right click again on your Visual Studio project and choose

'Add Web Reference'. In the form that pops up to add our reference in, rather than adding a url to a web service we need to add the physical path to our wsdl.xml file. For me this is:

C:\Users\Administrator\Documents\Visual Studio
2008\Projects\SalesForceBDC\SalesForceBDC\enterprise.wsdl.xml

Click the 'Go' button so that Visual Studio can validate the wsdl file and it will bring back information in the main control in the window. Finally change the web reference name to be ForceReference.



14, We want to be able to return a list of Salesforce accounts to the Business Data Catalog Data List Web Part, and create an association relationship so that when you select an account it returns the list of contacts for that account.

Now when you add a web reference to enterprise.wsdl.xml you will get an account and contact class generated for you - as well as the rest of the Salesforce objects you may need. These classes are huge though as they have a large number of properties - so we are going to define our own simple classes that will get loaded with the data and returned as List<> arrays. This also allows us to add the necessary WCF attributes to our class's properties.

IService.cs is where we define the contracts for our web methods and also the classes for account and contact. Double click IService.cs to open it and clear out the interface methods and class it generates in there for you automatically so you are left with this:

```
namespace SalesforceBDC
{
    [ServiceContract]
    public interface IService
```

```
    {  
    }  
}
```

15, We are going to define some simple properties for our Account and Contact classes. You can either type these in or copy and paste the code from here. This should be added below the closing IService squiggly bracket

```
[DataContract]  
public class Account  
{  
    [DataMember]  
    public string AccountId  
    {  
        get;  
        set;  
    }  
  
    [DataMember]  
    public string Name  
    {  
        get;  
        set;  
    }  
  
    [DataMember]  
    public string PhoneNumber  
    {  
        get;  
        set;  
    }  
}  
  
[DataContract]  
public class Contact  
{  
    [DataMember]  
    public string ContactId  
    {  
        get;  
        set;  
    }  
  
    [DataMember]  
    public string AccountId  
    {  
        get;  
        set;  
    }  
  
    [DataMember]  
    public string Name  
    {  
        get;  
        set;  
    }  
  
    [DataMember]
```

```

        public string Phone
        {
            get;
            set;
        }
    }
}

```

16, Within our interface class we want to define the methods that our WCF service is going to implement. These will be web methods that implement the various BDC methods such as the Finder, SpecificFinder and IdEnumerator methods.

```

[ServiceContract]
public interface ISalesForceBDC
{
    [OperationContract]
    List<Account> GetAccounts ();

    [OperationContract]
    Account GetAccountById (string accountId);

    [OperationContract]
    List<string> GetAccountIds ();

    [OperationContract]
    List<Contact> GetContacts ();

    [OperationContract]
    Contact GetContactById (string contactId);

    [OperationContract]
    List<string> GetContactIds ();

    [OperationContract]
    List<Contact> GetContactsByAccountId (string accountId);
}

```

17, Now that our ServiceContract and DataContracts have been defined we can move on to the concrete implementation of our Services in Service.svc.cs. Again we'll need to clear out the sample code that was created for us by Visual Studio so we can start with this:

```

public class Service : IService
{
}

```

Then we need to implement each of the methods defined in our interface. We'll just go over the GetAccounts() method and supporting method in detail, the rest of the methods you can get from the supporting code that is available for download.

```

private SforceService binding = null;

private QueryResult ExecuteSalesForceQuery (string queryText)
{
    binding = new SforceService ();
    LoginResult loginResult = binding.login ("username", "password");
    binding.Url = loginResult.serverUrl;
}

```

```

        binding.SessionHeaderValue = new SessionHeader { sessionId =
loginResult.sessionId };

        QueryResult qr;

        binding.QueryOptionsValue = new QueryOptions();
        binding.QueryOptionsValue.batchSize = 250;
        binding.QueryOptionsValue.batchSizeSpecified = true;

        qr = binding.query(queryText);

        return qr;
    }

    public List<Account> GetAccounts ()
    {
        List<Account> accounts = new List<Account>();

        QueryResult qr = ExecuteSalesForceQuery("Select Id, Industry, Name,
Phone from Account");

        bool cont = true;

        while (cont)
        {
            foreach (sObject record in qr.records)
            {
                ForceReference.Account account =
(ForceReference.Account)record;

                Account acc = new Account();
                acc.AccountId = account.Id;
                acc.Name = account.Name;
                acc.PhoneNumber = account.Phone;

                accounts.Add(acc);
            }

            //handle the loop + 1 problem by checking the most recent
queryResult
            if (qr.done)
                cont = false;
            else
                qr = binding.queryMore(qr.queryLocator);
        }

        binding = null;

        return accounts;
    }
}

```

The first private variable called binding is our web service proxy class. This is defined as a class variable so we can make use of the ExecuteSalesForceQuery method from our eventual web methods. The ExecuteSalesForceQuery method simply accepts some query text as a parameter, and executes this query returning a Salesforce QueryResult object.

Back in our `GetAccounts` method we can iterate over this `QueryResult` object, creating our own `Account` object and setting its properties before adding it to the `accounts List<>` that we eventually return.

This code is based on the samples up on the `SalesForce - Force.com` web site which I suggest you take a look at if you are interested in `SalesForce` development beyond this article:

http://www.salesforce.com/us/developer/docs/api/index_Left.htm

18, You can now go ahead and create the other methods that we need to present the account and contact data to the `Business Data Catalog`. You can get hold of the code to do this by getting the accompanying code download.

19, Once our other `BDC` methods are implemented you can test they work by pressing `F5` in `Visual Studio`. We can now test our web service using the `WcfTestClient` tool. Make a note or copy and paste the url presented in the browser that just popped up, it'll be something similar to:

<http://localhost:49528/Service.svc>

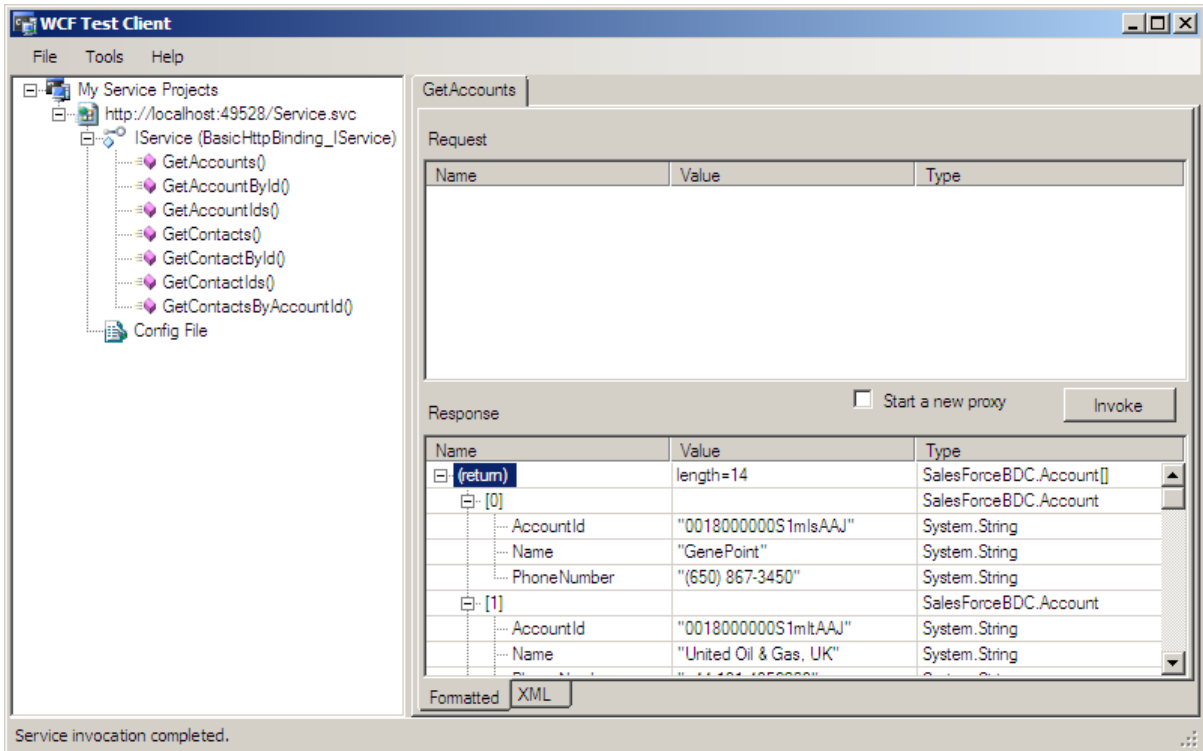
20, Open up a `Visual Studio 2008` command prompt from the `Start` menu and type the command `WcfTestClient`

This opens up the `WcfTestClient` application which you can use to execute the `WCF` methods we have just built.

21, Right click on '`My Service Projects`' and choose to '`Add Service...`'

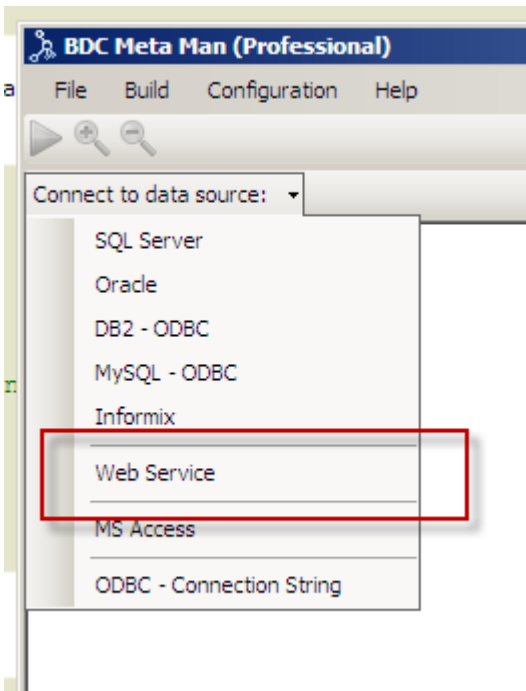
22, In the form that pops up paste the url you copied in step 19 as your endpoint address

23, You can now see all your `WCF` methods listed in the left hand pane, if you double click on one of them then you get some options to set parameters if the method expects it and also invoke the method. This is great as it allows you to debug your code as the `WcfTestClient` executes it which can make resolving problems much easier.

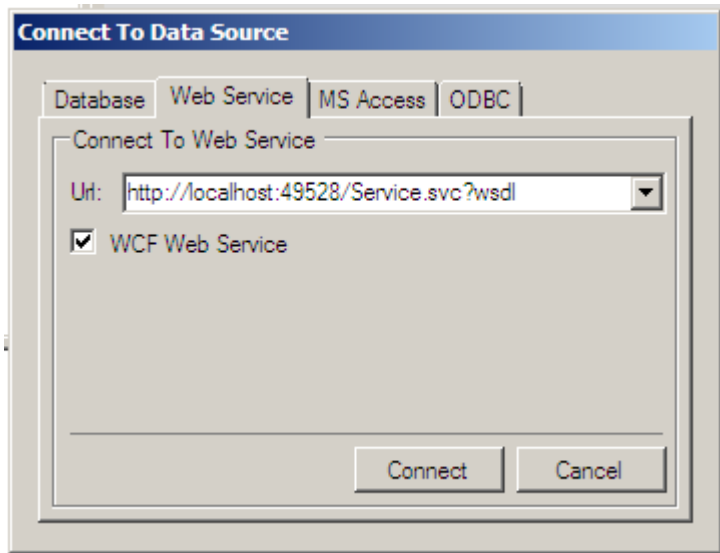


Once we've tested each of our methods with the WctTestClient we can now move on to defining our Business Data Catalog application definition file which will then allow us to use our Salesforce data within SharePoint. To do this we are going to use our tool BDC Meta Man. Everything we do in this walk through you can use the free Developer license of BDC Meta Man.

24, Open BDC Meta Man and choose to connect to a new Web Service



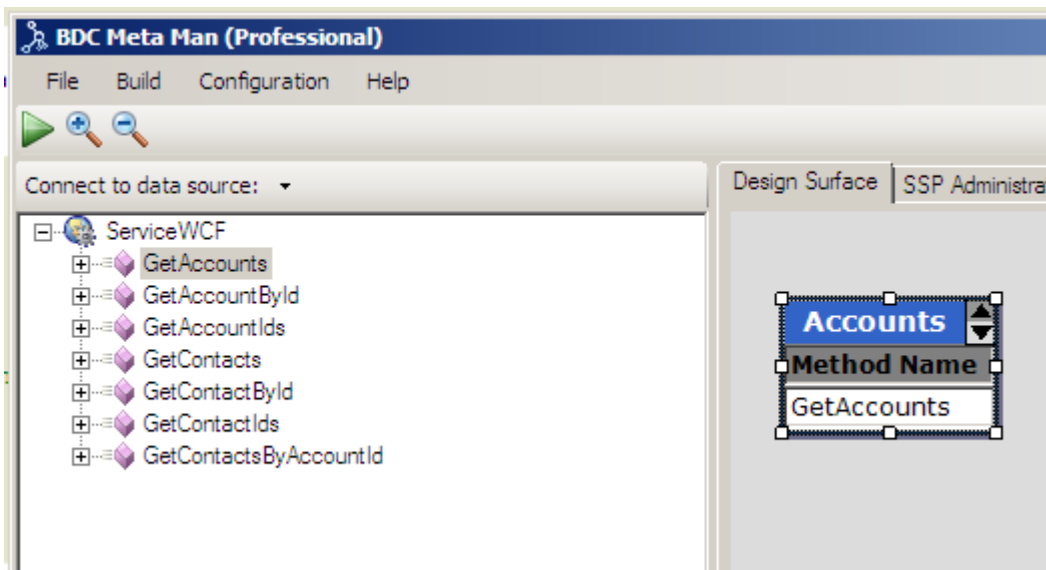
25, Paste in the url of our Salesforce WCF services - but also make sure you add ?wsdl to the end of the url. Finally tick the checkbox to let BDC Meta Man know this is a WCF web service



26, BDC Meta Man will now list the web service methods available to you in the top left window. Drag and drop the GetAccounts method onto the design surface.

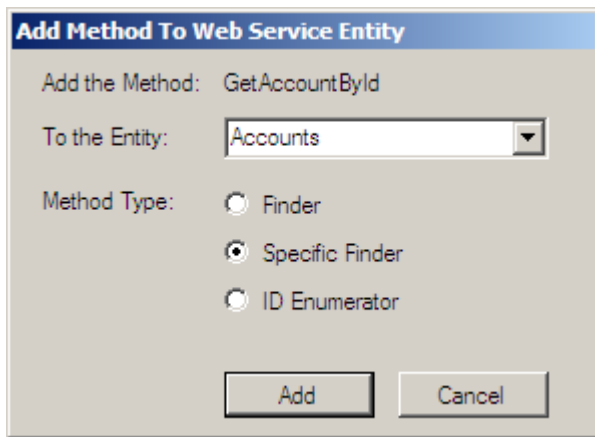
27, When prompted click Yes to create a new web service entity

28, Give the entity a name of Accounts and leave the method type as Finder -> click Create. Now we can see our basic BDC entity on the BDC Meta Man design surface.

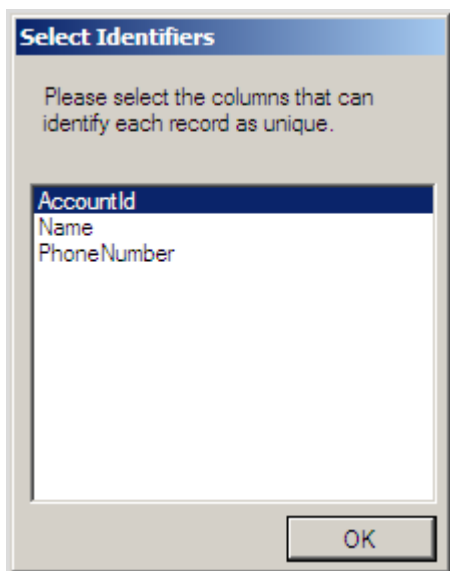


29, Now drag and drop the GetAccountById method onto the design surface. This time however choose No to the option of creating a new entity as we want to add this method to our Account entity we just created.

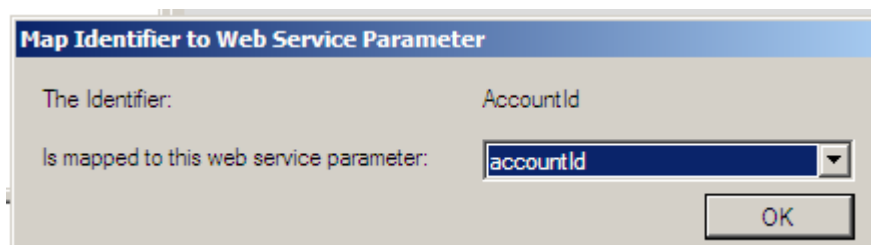
30, Ensure the entity selected is Accounts and check the SpecificFinder method type



31, We have a little more configuration to do for this method - we now have to select which column(s) can uniquely identify each row of data. In our case this is easy as it is AccountId, so just ensure this is selected and click OK



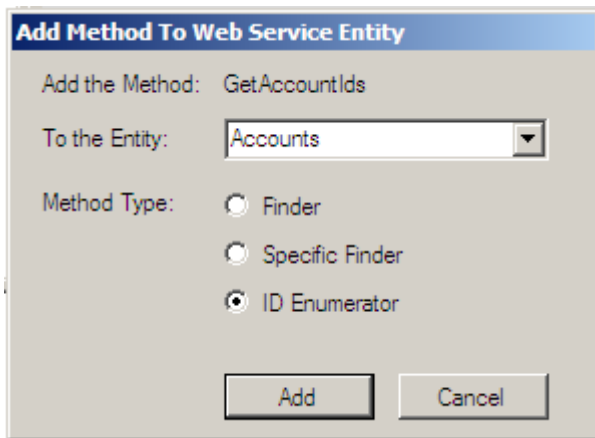
32, Finally for this web method we have to match each Identifier selected in step 2 to an input parameter. Luckily again for us this is easy as we only have one identifier, and one input parameter for the method, so again we can click OK



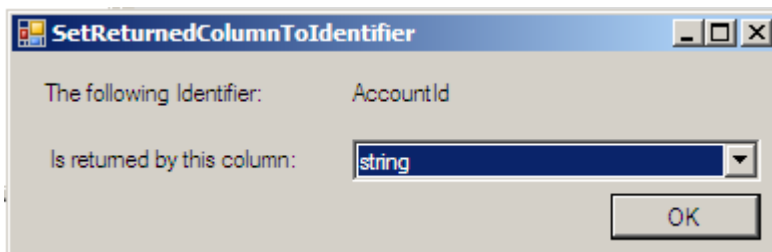
and that is our SpecificFinder method setup

33, Finally if we want to be able to crawl, index and search our Salesforce accounts our entity needs an IdEnumerator method. Drag and drop the GetContactIds method onto the design surface, again clicking No so we can add this method to our Accounts entity.

34, Ensure the method type selected is IdEnumerator

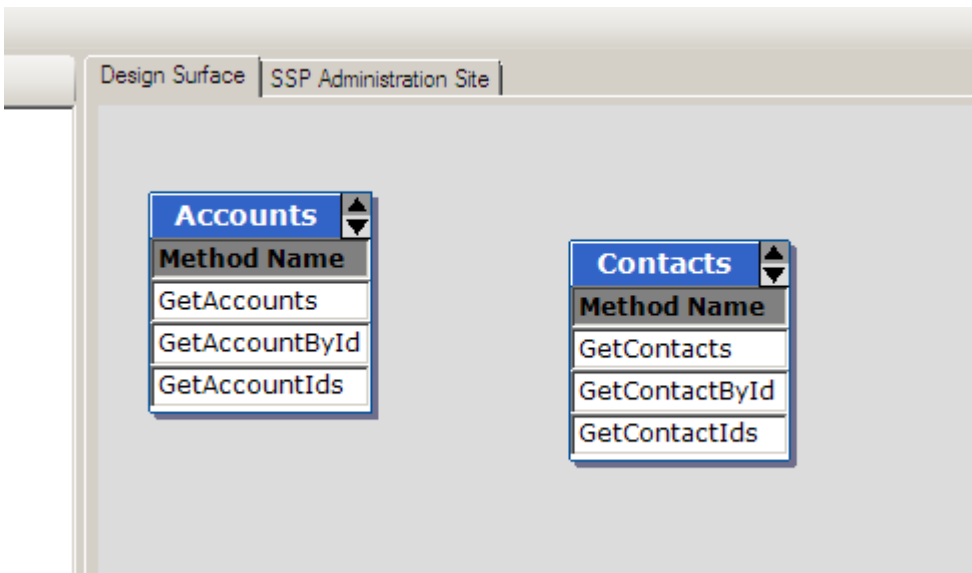


35, We now need to map the Accounts identifier to the value being return. Things are really easy as we just have one identifier so we can accept the default values and click OK



Now our Accounts Business Data Catalog entity has been configured correctly.

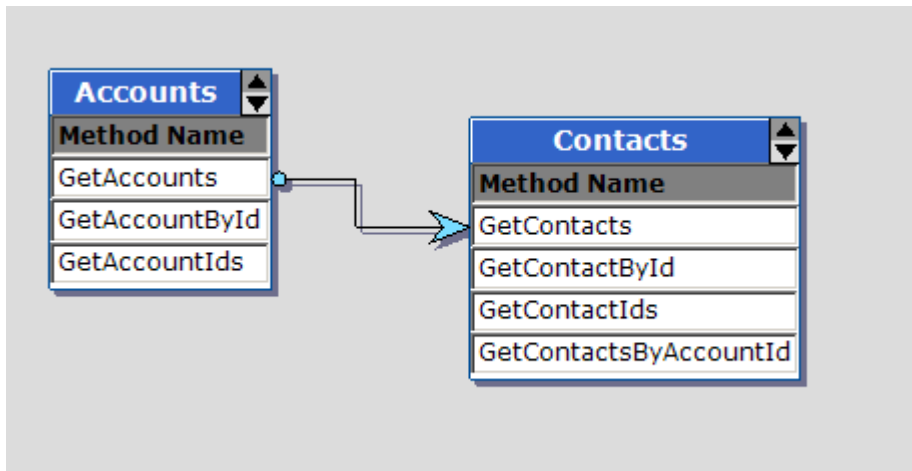
36, Please repeat steps 26 - 35 for the Contacts methods so that you create a Contacts entity.



Your design surface should look as above once step 36 has been completed.

37, The final job we need to do is to create an association method between Accounts and Contacts. Drag and drop the mouse cursor starting on the Accounts entity and finishing on the Contacts entity. Make sure you always drag and drop from the parent entity (Accounts) to the child entity (Contacts).

38, Upon joining the two entities together you'll be prompted to select which web method should be used for this association. Select GetContactsByAccountId, and then map the AccountId identifier to the accountId input parameter.



39, That's it! All we need to do now is select a place to generate our BDC application definition file by going Configuration -> Settings from the main menu and entering an 'Application Definition Filename'. Then you can click on the green Go button in the top left of the screen to generate all the XML. If you want to see all the hard work that BDC Meta Man has done for you click Yes when you are asked whether you want to view the XML in your registered editor.

40, Now we need to import our Business Data Catalog application definition file to MOSS 2007. Open up Central Administration and navigate to your Shared Service Provider admin page.

41, On your Shared Services Administration page click the 'Import Application Definition' link in the Business Data Catalog section.

42, Browse to where you generated the XML file with BDC Meta Man and click the import button.

Pressuming SharePoint imported your BDC file correctly we are now ready to go ahead and use our Salesforce data in SharePoint.

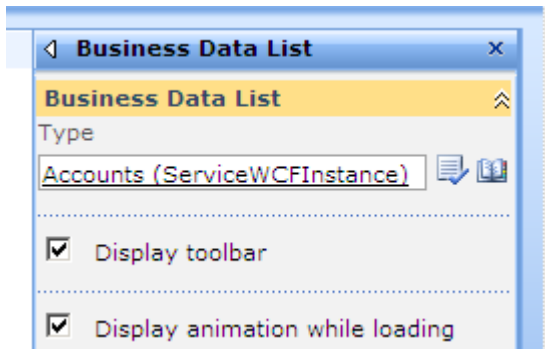
43, Open up the SharePoint site where you'd like to make use of your Salesforce data. Ensure this is a site that has the 'Office SharePoint Server Enterprise Site features' enabled. A standard team site is perfect for this.

44, Put the web part page into edit mode and click on the 'Add a Web Part' link for the zone where you want to place the BDC web parts.

45, From the web part gallery that opens up select the Business Data List and Business Data Related List web parts to add. Click the 'Add' button to add them.

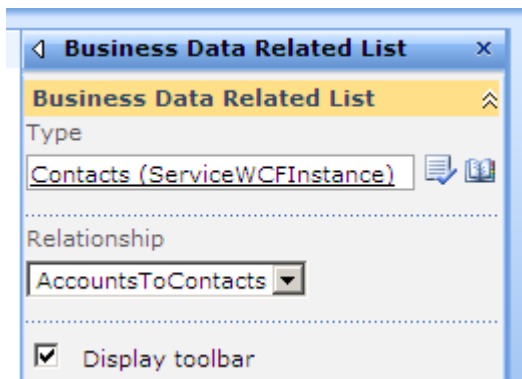
46, First we need to configure the Business Data List web part. To do this click on the link that prompts you to 'Open the toolpane and choose the type of data to display'

47, In the web part toolbar that opens, click the address button to pick a BDC entity and select the Salesforce Account entity.



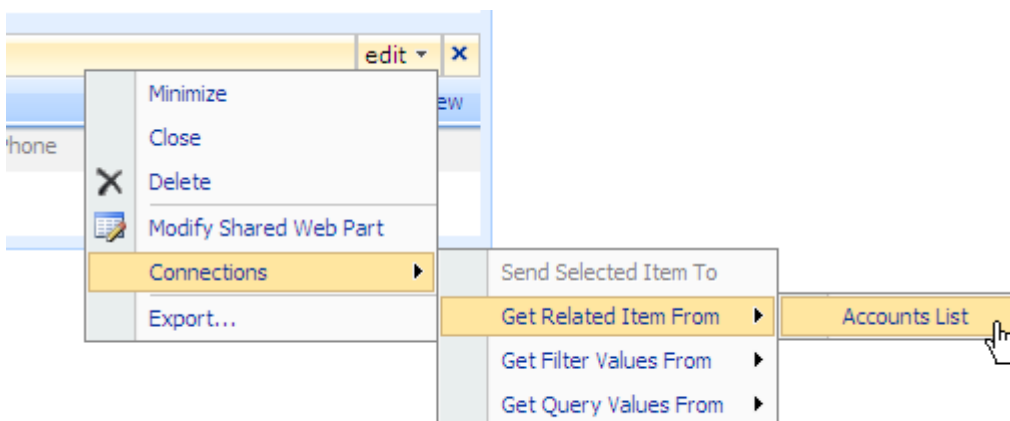
48, Click OK within the web part toolbar to close it and you will see your BDC Data List web part fill with your Salesforce accounts!

49, Now we need to configure the related list web part. Click on the link within the web part to configure it, and this time using the entity picker select the Salesforce Contacts entity (depending on whether the BDC is already being used on your setup this may be the only option available). Click OK on the web part toolbar to close it down.



50, You'll now see the web part giving you a hint that it needs to be connected. To do this click the following:

Edit -> Connections -> Get Related Item From -> Account List



After doing this you'll see that the Accounts list has changed slightly to include a radiobutton next to each account. Select an account by clicking on a radio button and you'll see the Contacts are brought back for that account record.....excellent!!!

This is the end of demonstrating how to get started with using your Salesforce data within the Business Data Catalog but you should check out the other BDC support articles at <http://www.lightningtools.com/bdc-meta-man/using-business-data-catalog-and-bdc-metaman.aspx> on how to configure search and the business data column.

To get you up and running quickly we are hosting the Salesforce web service at <http://salesforce.lightningtools.com/service.svc>

You can try using the web service here for simple demo purposes, or download the source code and modify the username and password to get your own Salesforce data into SharePoint.