

ENHANCING DBXL SUBMIT IN INFOPATH 2007

The following steps outline the procedure for implementing “save your progress” in an InfoPath 2007 form that uses the 2007 Object Model. This document works best with C#, but with a few small modifications can be made to work with VB.NET instead.

This document assumes that you have completed the steps in the Getting Started document pertaining to setting up an InfoPath form to submit to DBXL.

1. ADD THE DBXLSUBMITLOGIC DATA SOURCE

We will begin by adding a simple secondary XML data source to contain some values that will be used by this feature. Please find the DbxlSubmitLogic.xml file included in the **IP2007 Files** folder of this tutorial and open it in any text editor. You will need to make a small change to the file before using it in your form.

The XML file contains four fields:

```
<?xml version="1.0" encoding="utf-8"?>
<SubmitLogic>
  <DocumentType>!!Enter your document type name here!!</DocumentType>
  <Success />
  <ErrorMessage />
  <DocId />
</SubmitLogic>
```

Please modify the **DocumentType** field to contain the name of the DBXL document type where you intend to publish this form. For this tutorial, we will use the document type name “SaveYourProgress.” Your document type name should contain only letters and numbers, and no spaces.:

```
<?xml version="1.0" encoding="utf-8"?>
<SubmitLogic>
  <DocumentType>SaveYourProgress</DocumentType>
  <Success />
  <ErrorMessage />
  <DocId />
</SubmitLogic>
```

Then save the file and close your text editor.

Now it's time to add the XML file to your InfoPath form. Open the **Data Connections** dialog from **Tools->Data Connections...** and click the **Add...** button.

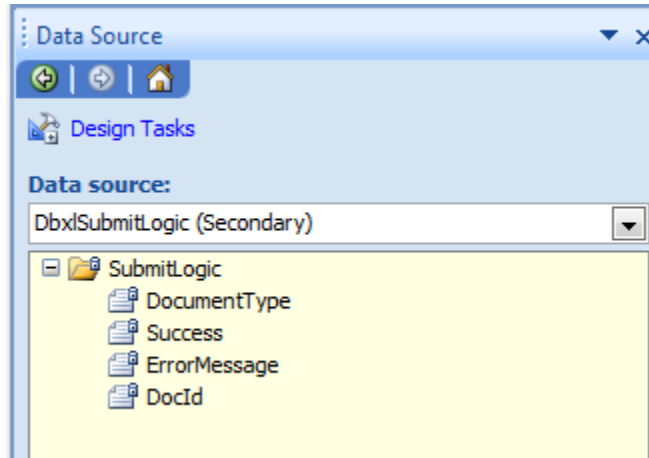
Select the **Receive data** option under **Create a new connection to:** and click **Next >**.

Select **XML Document** and click **Next >**.

Use the **Browse...** button and **Open File** dialog to locate the `DbxlSubmitLogic.xml` file and click **OK**.

Click **Next >** twice and **Finish** to accept all of the remaining defaults.

Your form should now have a new data source called *DbxlSubmitLogic*:



2. ADDING THE DBXL SUBMIT CONNECTION

This step is similar to the steps detailed in the Getting Started document, with a few small differences. Firstly, this time we will not be using a button to invoke the submit connection directly, and secondly, we will create a mapping between the submit connection and the *DocumentType* field in the data source you just added.

Open the **Data Connections** dialog again, and click **Add...** again.

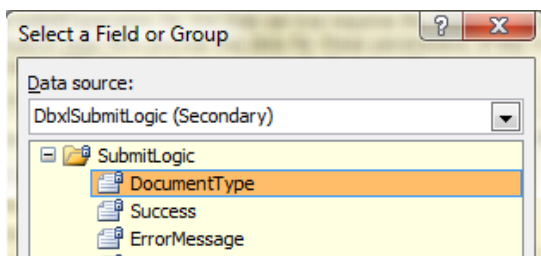
This time, select **Submit Data** under **Create a new connection to:** and click **Next >**.

Select **To a Web service** and click **Next >**.

Enter the URL of the **DbxlDocumentService.asmx** file on your installation of DBXL (please consult the Getting Started guide if you are not sure how to do this), and click **Next >**.

Select the **SubmitDocument** operation and click **Next >**.

For the **docTypeName** parameter, use the **Select Field** button  to select the *DocumentType* field in the *DbxlSubmitLogic* data source:



Make sure to check the **Submit data as a string** option.

For the **xml** parameter, select the **Entire form** option, and again remember to check the **Submit data as a string** option.

Optional – Document metadata values:

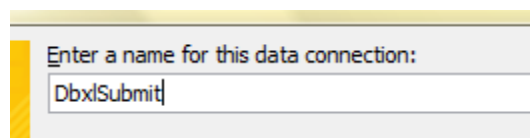
The **SubmitDocument** web method allows specifying three values to use as metadata for each of the forms that are submitted to DBXL. These values are:

- Name – A title for each form
- Author – The name of the person who created the form, or who last modified it, depending on how you would like to use this value.
- Description – A description for each form

If there are any fields in your form that would correspond to any of these metadata values, you may add mappings to them, remembering to check the **Submit data as a string** option.

Please leave any of these fields that you do not need unset.

Click **Next >**, and in the next dialog, choose a descriptive name for this data connection. We suggest using the name *DbxlSubmit* or *DBXLSubmit*.



Click **Finish** to finish setting up the data connection.

3. ADDING MANAGED CODE TO YOUR INFOPATH FORM

You may skip this step if your form is already set up to use InfoPath 2007 Managed code. In this situation, simply open Visual Studio Tools for Applications (VSTA) by selecting **Tools->Programming->Microsoft Visual Studio Tools for Applications**

Open the **Form Options** dialog by choosing **Tools->Form Options...** from the menu.

Select the **Programming** category. From the **Form template code language** dropdown, select C#. You may use the **Browse...** button to choose an alternate location for the code project. Click **OK** when you are done.

4. ADDING THE DBXL SUBMIT LOGIC TO YOUR CODE PROJECT

Now it's time to add some code to your code project that will handle the work of submitting your form to DBXL and updating the main data source accordingly afterward.

For this step, this tutorial gives you two options. You may either add the raw source code to your project, or add a DLL assembly with the code already built. Using the raw source code gives you the ability to see how the code works, and modify it if you would like to. Using the DLL saves you from the small hassle of working with the raw source code.

Please select one option or the other, and then proceed.

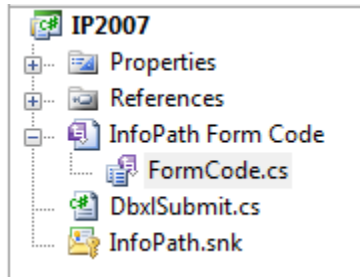
If your project already uses VB.NET code, you must select the DLL option.

OPTION 1: ADDING THE RAW SOURCE CODE

In VSTA, select **Add Existing Item...** from the **Project** menu.

Navigate to the **IP2007 Files** folder of this tutorial and select the *DbxSubmit.cs* file, then click **OK**.

Your project should now contain the *DbxSubmit.cs* file:

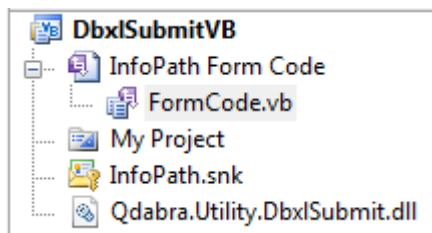


OPTION 2: ADDING THE PREBUILT DLL

In VSTA, select **Add Existing Item...** from the **Project** menu.

Navigate to the **IP2007 Files** folder of this tutorial and select **Executable Files** from the **Files of Type** dropdown box, and select the *Qdabra.Utility.DbxSubmit.dll* file. Click **OK**.

Your project should now contain the *Qdabra.Utility.DbxSubmit.dll* file.



Select **Add Reference...** from the **Project** menu.

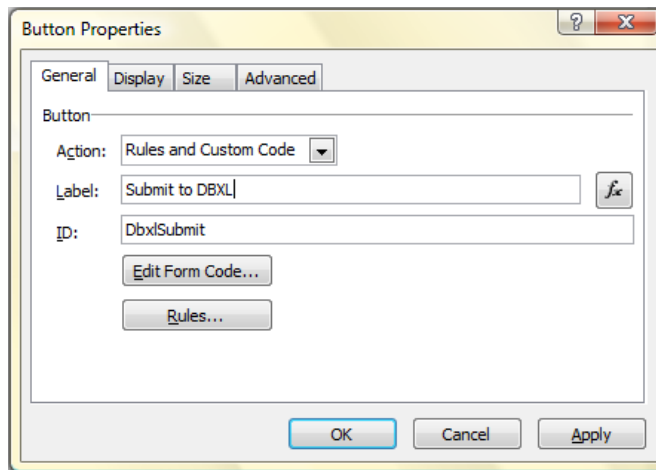
Select the **Browse** tab, and this time browse to the *Qdabra.Utility.DbxSubmit.dll* file **within your code project (not in the IP2007 Files folder)**.

Click **OK**.

5. ADDING THE SUBMIT BUTTON AND ITS CODE TO YOUR FORM

Back in the InfoPath designer, drag a button into your form.

Open the button's properties, and give it any Label and ID that you see fit. We'll use "Submit to DBXL" and "DbxSubmit" respectively.



Click the **Edit Form Code...** button and InfoPath should automatically create an event handler for your button and position your cursor inside it:

```
public void DbxlSubmit_Clicked(object sender, ClickedEventArgs e)
{
    // Write your code here.
}
```

Scroll to the top of your **FormCode** file, and add a **using** statement to let VSTA know that you intend to use the DbxlSubmit logic from your form code:

```
using System.Xml.XPath;
using mshtml;
using Qdabra.Utility;
```

In VB.net, the equivalent line would be

```
Imports Qdabra.Utility
```

Now return to the button's event handler method and add the following code:

C#

```
DbxlSubmitResult result = DbxlSubmit.Submit(this, "DbxlSubmit");

XPathNavigator submitLogicNav = DataSources["DbxlSubmitLogic"].CreateNavigator();
submitLogicNav.MoveToChild("SubmitLogic", string.Empty);

submitLogicNav.SelectSingleNode("Success").SetValue(XmlConvert.ToString(result.Success));
submitLogicNav.SelectSingleNode("ErrorMessage").SetValue(result.Message);
submitLogicNav.SelectSingleNode("DocId").SetValue(result.DocId.ToString());
```

VB.NET

```
Dim result As DbxlSubmitResult = DbxlSubmit.Submit(Me, "DbxlSubmit")

Dim submitLogicNav As XPathNavigator
```

```

submitLogicNav = DataSources("DbxlSubmitLogic").CreateNavigator()
submitLogicNav.MoveToChild("SubmitLogic", string.Empty)

submitLogicNav.SelectSingleNode("Success").SetValue(XmlConvert.ToString(result.Success))
submitLogicNav.SelectSingleNode("ErrorMessage").SetValue(result.Message)
submitLogicNav.SelectSingleNode("DocId").SetValue(result.DocId.ToString())

```

Let's take a look at this code line by line:

```
DbxlSubmitResult result = DbxlSubmit.Submit(this, "DbxlSubmit");
```

This line calls the **Submit()** method in the DbxlSubmit logic, which attempts to submit the form to DBXL and update the main data source accordingly. This method takes two arguments:

formHost – A reference to the FormCode's form host object. You should typically just use the `this` keyword here (`Me` in VB.NET).

dataConnection – The name of the Submit data connection that is set up to submit to DBXL.

Note that this value is case sensitive.

The **Submit()** method returns a simple object of the type **DbxlSubmitResult** which contains four values about the success or failure of the submit, and the result, if it succeeded:

Success – A Boolean value indicating whether the submit succeeded or not.

ErrorMessage – A string indicating the reason the submit failed if it failed, or a blank string if the submit succeeded.

DocId – The DocId of the submitted document if the submit succeeded, or 0 if it failed.

RefId – The RefId of the submitted document if the submit succeeded, or a blank string if it failed.

```
XPathNavigator submitLogicNav = DataSources["DbxlSubmitLogic"].CreateNavigator();
```

This line retrieves a reference to an XPathNavigator for the **DbxlSubmitLogic** data source that you created in step 1. We will use this reference to store some values pertaining to the submit in this data source, so that they will be accessible from within InfoPath.

Note that "DbxlSubmitLogic" here is the name of the secondary data source. If you used a different name or capitalization for this data source, please adjust this line accordingly.

```
submitLogicNav.MoveToChild("SubmitLogic", string.Empty);
```

This line repositions the *submitLogicNav* XPathNavigator at the *SubmitLogic* root node of the data source.

```

submitLogicNav.SelectSingleNode("Success").SetValue(XmlConvert.ToString(result.Success))
submitLogicNav.SelectSingleNode("ErrorMessage").SetValue(result.Message)
submitLogicNav.SelectSingleNode("DocId").SetValue(result.DocId.ToString())

```

These three lines take the values stored in the *result* object, and store them in nodes in the *DbxlSubmitLogic* data source so that they will be accessible from within InfoPath.

Ensure that your project builds by selecting the **Build [ProjectName]** option from the build menu. Fix any errors that occur and then close VSTA.

6. SHOWING THE DBXLSUBMITLOGIC VALUES IN YOUR FORM

Back in the InfoPath designer, open the **Data Source Taskpane**, and select the *DbxlSubmitLogic* data source.

With your left mouse button, drag the **Success**, **ErrorMessage**, and **DocId** fields into your form to represent them as textboxes.

The above step is primarily for demonstration purposes. In actual practice, you would probably want to represent these values in your form in a more graceful manner, and use the value of the **Success** field to determine the behavior of your form.

If you don't have any fields in your main data source yet, drag a TextBox control from the **Controls** taskpane to represent a value stored in your main data source. Our finished example form looks like this:

Field value

Result:

Success:

Error Message:

Doc Id:

Now publish your form to a location on your hard drive and open up DBXL's DAT. Create a new document type with the name you chose in step 1 of this tutorial, and attach the published form. Save your document type.

7. TRYING OUT THE FORM

Open the form template from DAT using the Download icon:



The form should open in InfoPath.

Enter some sample values for your form's fields (not the result fields), and click Result.

If the submit worked as expected, you should see a "true" value in the success field, and a docid number in the DocId field:

Field value

Result:

Success:

Error Message:

Doc Id:

Repeatedly clicking the button should cause any modification in the form's values to be submitted, but will not create new documents. You can verify this in the **Documents** tab of DAT:

Documents						 Refresh	
						 Delete  Delete All	
						 Reshred  Reshred All	
Doc. ID	Ref. ID	Version #	Name	Author	Description	New	
9010	3	1				Open 	