



QDABRA DATABASE ACCELERATOR V2.3

Use Document-Level Permissions for browser-based forms in a data view

With the increasing popularity of *InfoPath Forms Services (IPFS)* running on *Microsoft Office SharePoint Services (MOSS)*, people are wondering how to connect DBXL with a browser-enabled form and respect DBXL document-level permissions. To do this with DBXL, we'll need some form code to access the DBXL web services. This document will give you the steps on how to open forms in the browser using a SharePoint data view that respects DBXL permissions.

PRE-REQUISITES

I will make some assumptions during this tutorial about how the form template is set up. The instructions provided here can be adapted to any form template. However, if you are not an advanced form template developer you may wish to follow the tutorial through once exactly as described before attempting this on a different form template.

1. I will be customizing the ExpenseReport sample that comes with InfoPath 2007. This form template does not contain any existing load or submit handlers. If your form template does, you'll need to modify them as fits your scenario to connect with the sample code I provide in this tutorial.
2. I will assume you have a working MOSS 2007 server, have verified that IPFS is working correctly on simple forms, and that DBXL v2.3 is installed and working on the main SharePoint web site at port 80.

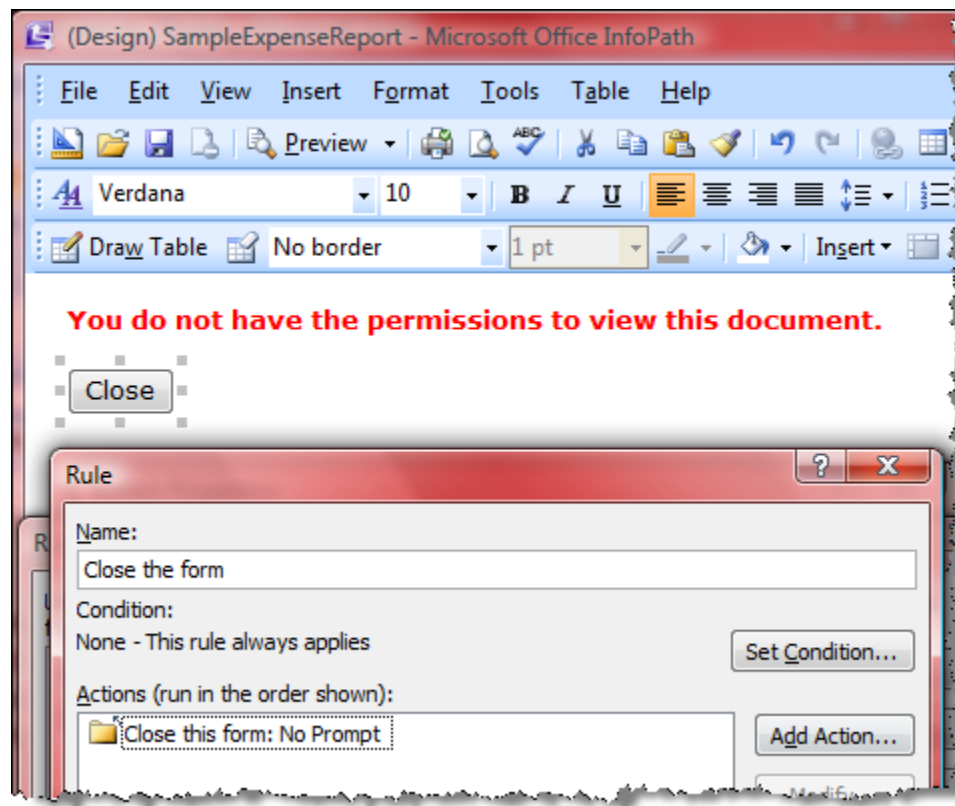
If your form does NOT contain code, you should first read the document called [How to IPFS-enable a Codeless DBXL solution](#). If that document does not satisfy your requirements, then proceed with the steps below.



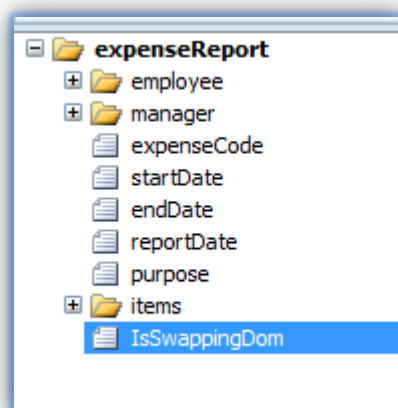
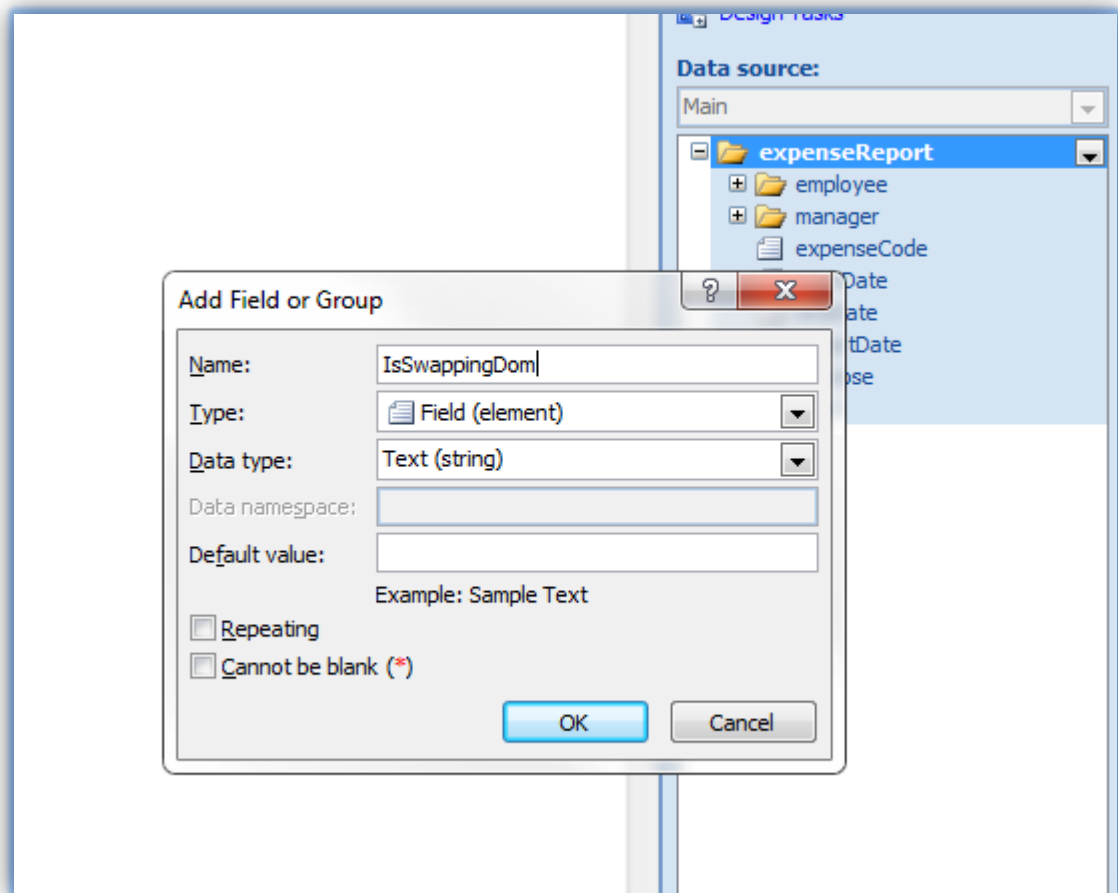
PREPARE YOUR FORM TEMPLATE

For the purposes of this document, we will use the sample Expense Report form template that ships with InfoPath.

1. Launch InfoPath. In the **Getting Started** dialog, click **Design a Form Template > Customize a Sample**, and then select the template called *Sample - Expense Report*.
2. Create a new view called "Unauthorized". In this view, add text such as: *"You do not have the permissions to view this document"*, or any warning message that informs the user about his or her lack of permission to access the document.
3. Add a button with a rule that closes the form.



4. Add a new field to the main data source called "IsSwappingDom" of a data type string.



5. Save the form template to a convenient location such as *C:\SampleExpenseReport*.



ADD THE FORM CODE

SETTING UP THE WEB SERVICE DATA CONNECTIONS

The form code will use two web service data adapters:

- **GetDocument** will retrieve document data from DBXL
- **SubmitDocument** will post a modified document back to DBXL.

First, we'll configure the **GetDocument** data adapter.

1. From the **Tools** menu, choose **Data Connections...**, then click the **Add...** button.
2. Select the **Create a new connection to** radio button. Then choose **Receive data**. Click **Next** in the wizard.
3. Under **From where do you want to receive your data?**, choose **Web service**. Click **Next**.
4. Enter the location of the Qdabra DBXL Document web service. It likely looks like <http://<your-server-name>/QdabraWebService/DbxlDocumentService.aspx>. Click **Next**.
5. Select the operation called **GetDocument**, near the bottom of the list. Click **Next**.
6. For **tns:docId**, click **Set Sample Value...**, enter 0 (that is, zero), click **OK**, then click **Next**.
7. No value is needed for **tns:docId** on the next panel. Just click **Next**.
8. There is no need to store a copy of the data in the form template. Just click **Next**.
9. Uncheck the box **Automatically retrieve data when the form is opened**. Our code will do that instead. Then click **Finish**.

Now we'll configure the **SubmitDocument** data adapter.

1. We should be back at the data connections dialog box. Click **Add...**
2. Select the **Create a new connection to** radio button. Then choose **Receive data**. Click **Next** in the wizard. Note: you are choosing **Receive data** even though we think of the operation as submitting the document. This is a result of how the web service method was defined.
3. Under **From where do you want to receive your data?**, choose **Web service**. Click **Next**.
4. Enter the location of the Qdabra DBXL Document web service, same as above in step 6. Click **Next**.
5. Select the operation called **SubmitDocument**, near the bottom of the list. Click **Next**.
6. No value is needed for any of the parameters on this panel. Just click **Next**.
7. There is no need to store a copy of the data in the form template. Just click **Next**.
8. Uncheck the box **Automatically retrieve data when the form is opened**. Our code will do that instead. Then click **Finish**.
9. Back at the data connections dialog box, click **Close**.



WRITING THE CUSTOM CODE TO QUERY AND SUBMIT THE DBXL DOCUMENT

1. Click the **Tools** menu, locate the submenu **Programming**, and choose the item **Loading Event....**
VSTA will launch and show code for an empty event handler, called **FormEvents_Loading**:

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

2. Back in InfoPath Designer, go to the **Tools** menu and choose **Submit Options....**
3. In Expense Report, **Allow users to submit this form** will already be checked. If it is not for your form template, check it now.
4. Choose **Perform custom action using Code**.
5. Click **Edit Code....** VSTA will flash on the task bar. If you switch to it you'll notice another empty submit handler, called **FormEvents_Submit**, has appeared:

```
public void FormEvents_Submit(object sender, SubmitEventArgs e)
{
    // If the submit operation is successful, set
    // e.CancelableArgs.Cancel = false;
    // Write your code here.
}
```

6. Back in InfoPath Designer, click **OK** to dismiss the submit handler dialog box.
7. We'll need to add some subroutines to the form code which will handle querying the web service we added earlier. You will see below that we are defaulting the document type name as "MyDocType". You can use a different name but make sure to use the same name when you create your document type in DBXL later on. Note that the code below requires Visual C#. Please insert this code just after the closing brace for the empty **FormEvents_Loading** function:

```
public string GetDocTypeName()
{
    string docType = "MyDocType" /* Default value */;

    return docType;
}
private bool LoadFromDbxl(string docId)
{
    try
    {
        XPathNavigator domGetDocument =
        DataSources["GetDocument"].CreateNavigator();

        // Set docId argument for the DBXL web service call. The
        namespaces in your XPath may change -- use "Copy XPath" in the data
    }
}
```



```
source task pane to get the correct XPath to this node.

domGetDocument.SelectSingleNode("/dfs:myFields/dfs:queryFields/tns:GetDoc
ument/tns:docId", NamespaceManager).SetValue(docId);

        // Invoke the web service method to query DBXL for the
document.
        DataConnection connGetDocument =
DataConnections["GetDocument"];
        connGetDocument.Execute();

        // Check for error. The namespaces in your XPath may
change -- use "Copy XPath" in the data source task pane to get the
correct XPath to this node.
        if
(!domGetDocument.SelectSingleNode("/dfs:myFields/dfs:dataFields/tns:GetDo
cumentResponse/tns:GetDocumentResult/tns:Success",
NamespaceManager).ValueAsBoolean)
            throw new Exception("Failed to retrieve the form from
the server.");

        // Replace main DOM with obtained document.
        XPathNavigator root =
MainDataSource.CreateNavigator().SelectSingleNode("/child:*",
NamespaceManager);

        // Get a reference to the my:myFields node of the
GetDocument DOM. The namespaces in your XPath may change -- use "Copy
XPath" in the data source task pane to get the correct XPath to this
node.
        XPathNavigator newDoc =
domGetDocument.SelectSingleNode("/dfs:myFields/dfs:dataFields/tns:GetDocu
mentResponse/tns:docInfo/tns:Content/child:*", NamespaceManager);

        // If your "IsSwappingDom" node is located elsewhere,
this xpath will change
        newDoc.SelectSingleNode("my:IsSwappingDom",
NamespaceManager).SetValue("true");
        root.InnerXml = newDoc.InnerXml;
        // If your "IsSwappingDom" node is located elsewhere,
this xpath will change

root.SelectSingleNode("/my:expenseReport/my:IsSwappingDom",
NamespaceManager).SetValue("false");

        // Add new or updated QdabraDBXL PI so subsequent saves
will overwrite DBXL document.
        InsertQdabraDbxlPi(docId);
    }
    catch (Exception ex)
    {
        return false; // Failure.
    }
    return true; // Success.
```



```
    }
    private bool SubmitToDbxl(string docType, string name, string
author, string description)
    {
        XPathNavigator domMainDocument =
MainDataSource.CreateNavigator();
        XPathNavigator domSubmitDocument =
DataSources["SubmitDocument"].CreateNavigator();

        try
        {
            // Set the arguments for the SubmitDocument web service
call.

domSubmitDocument.SelectSingleNode("/dfs:myFields/dfs:queryFields/tns:Sub
mitDocument/tns:docTypeName", NamespaceManager).SetValue(docType);

            // Notice that tns:xml will contain the entire main
document being submitted to DBXL.

domSubmitDocument.SelectSingleNode("/dfs:myFields/dfs:queryFields/tns:Sub
mitDocument/tns:xml",
NamespaceManager).SetValue(MainDataSource.CreateNavigator().OuterXml);

domSubmitDocument.SelectSingleNode("/dfs:myFields/dfs:queryFields/tns:Sub
mitDocument/tns:name", NamespaceManager).SetValue(name);

domSubmitDocument.SelectSingleNode("/dfs:myFields/dfs:queryFields/tns:Sub
mitDocument/tns:author", NamespaceManager).SetValue(author);

domSubmitDocument.SelectSingleNode("/dfs:myFields/dfs:queryFields/tns:Sub
mitDocument/tns:description", NamespaceManager).SetValue(description);

            // Invoke the web service method to submit the document
data to DBXL.
            DataConnection connSubmit =
DataConnections["SubmitDocument"];
            connSubmit.Execute();

            // Check for error.
            if
(!domSubmitDocument.SelectSingleNode("/dfs:myFields/dfs:dataFields/tns:Su
bmitDocumentResponse/tns:SubmitDocumentResult/tns:Success",
NamespaceManager).ValueAsBoolean)
                throw new Exception("Submit failed");

            // Add new or updated QdabraDBXL PI so subsequent saves
will overwrite DBXL document.
            string docId =
domSubmitDocument.SelectSingleNode("/dfs:myFields/dfs:dataFields/tns:Subm
itDocumentResponse/tns:docId", NamespaceManager).Value;
            InsertQdabraDbxlPi(docId);
        }
    }
```



```
        catch
        {
            return false; // Failure.
        }

        return true; // Success.
    }

    private void InsertQdabraDbx1Pi(string docId)
    {
        string docType = GetDocTypeName();

        XPathNavigator domMainDocument =
MainDataSource.CreateNavigator();

        // Remove any existing QdabraDBXL PI.
        XPathNavigator oldPi =
domMainDocument.SelectSingleNode("/processing-instruction() [local-name(.)
= 'QdabraDBXL']", NamespaceManager);
        if (oldPi != null)
        {
            oldPi.DeleteSelf();
        }

        if (docId != "")
        {
            // Add new or updated QdabraDBXL PI so subsequent saves
will overwrite DBXL document.
            string newPi = String.Format("<?QdabraDBXL docid=\"{0}\" "
doctype="{1}\" ?>", docId, docType);
            domMainDocument.SelectSingleNode("/processing-
instruction() [local-name(.) = 'mso-infoPathSolution']",
NamespaceManager).InsertBefore(newPi);
        }
    }
}
```

By setting the IsSwappingDom node to true when loading the XML into the form, rules can be based off the value of IsSwappingDom. This way you can prevent running rules or other code during the loading of the XML.

NOTE: When adding the code above, the namespace in your XPath may change. Use "Copy XPath" in the data source task pane to get the correct XPath to these nodes.

- **tns:docId:** `/dfs:myFields/dfs:queryFields/tns:GetDocument/tns:docId`



```
private bool LoadFromDbx1(string docId)
{
    try
    {
        XPathNavigator domGetDocument = DataSources["GetDocument"].CreateNavigator();

        // Set docId argument for the DBXL web service call. The namespaces in your XPath may change
        domGetDocument.SelectSingleNode("/dfs:myFields/dfs:queryFields/tns:GetDocument/tns:docId", N
```

- **tns:Success:**

```
/dfs:myFields/dfs:dataFields/tns:GetDocumentResponse/tns:GetDocumentResult/tns:Success
```

spaces in your XPath may change -- use "Copy XPath" in the data source task pane to get the correct
 .SingleNode("/dfs:myFields/dfs:dataFields/tns:GetDocumentResponse/tns:GetDocumentResult/tns:Success").
 .led to retrieve the form from the server.");

- **tns:Content:**

```
/dfs:myFields/dfs:dataFields/tns:GetDocumentResponse/tns:docInfo/tns:Content/child::*
```

When you copy the XPath to this node, append /child::*

Document DOM. The namespaces in your XPath may change -- use "Copy XPath" in the data source
 de("/dfs:myFields/dfs:dataFields/tns:GetDocumentResponse/tns:docInfo/tns:Content/child::*").

- **IsSwappingDom:** If your "IsSwappingDom" node is located elsewhere, make sure to copy the correct XPath in these lines:

```
// If your "IsSwappingDom" node is located elsewhere, this xpath will change
newDoc.SelectSingleNode("my:IsSwappingDom", NamespaceManager).SetValue("true");
root.InnerXml = newDoc.InnerXml;
// If your "IsSwappingDom" node is located elsewhere, this xpath will change
root.SelectSingleNode("/my:expenseReport/my:IsSwappingDom", NamespaceManager).S
```

For more information, please read the comments provided in the code.

8. Next, we will fill in the **FormEvents_Loading** event handler. Insert the following code into the body of the **FormEvents_Loading** function:

```
this.NamespaceManager.GetNamespacesInScope(XmlNamespaceScope.All);

    string docId = null;

    //Do a case-insensitive search for the docid querystring
    parameter on the URL.
    foreach (System.Collections.Generic.KeyValuePair<string,
string> inputParameter in e.InputParameters)
```



```
{
    if (inputParameter.Key.Equals("docid",
StringComparison.OrdinalIgnoreCase))
    {
        docId = inputParameter.Value;
    }
}

if (!String.IsNullOrEmpty(docId))
{
    if (!LoadFromDbxl(docId))
    {
        e.SetDefaultView("Unauthorized");
    }
}
}
```

9. Finally, we will fill in the code for the **FormEvents_Submit** handler. Insert the following code into the body of the **FormEvents_Submit** function.

```
// These document properties can be customized as desired

string title = "Expense Report";
string user = Application.User.UserName;
string description =
MainDataSource.CreateNavigator().SelectSingleNode("/my:expenseReport/my:p
urpose", NamespaceManager).Value;

SubmitToDbxl(GetDocTypeName(), title, user, description);

e.CancelableArgs.Cancel = false;
```

Note that the strings “title”, “user”, and “description” refers to DBXL’s document properties (Name, Author and Description). You might want to change the values in the code to suit your form. You will see the result when you submit documents later on:

Documents							
Doc. ID	Ref. ID	Version #	Name	Author	Description	New	
56	1	1	Expense Report	jo-anne	test	Open	☐
57	2	1	Expense Report	jo-anne	test2	Open	☐
58	3	1	Expense Report	jo-anne	test3	Open	☐

10. **Save** the form code in VSTA, and then switch back to the InfoPath Designer.



CONVERT THE DATA CONNECTIONS FOR SERVER USE

In order for the form template to be able to access the web service methods, the data connections must be converted to udcx files on a server and approved.

1. Create a Data Connections Library on your SharePoint server. (You may skip these steps if you already have an existing data connections library.)
 - a. In your web browser, navigate to your SharePoint web site.
 - b. Click on **Document Center**.
 - c. On the right side of the page, click on the **Site Actions** menu and choose **Create**.
 - d. Under the **Libraries** heading, choose **Data Connection Library**.
 - e. Enter a name, such as *MyDataConnections* and click **OK**.
 - f. You will now have a new Data Connections Library at a URL like <http://<your-server-name>/Docs/MyDataConnections/Forms/AllItems.aspx>.
2. Convert the form template's data connections.
 - a. In the InfoPath Designer, from the **Tools** menu, open **Data Connections...**
 - b. At this time we need to remove the *Main submit* data connection that came with the sample template since it will prevent deployment on the server. Make sure *Main submit* is selected and click **Remove**. Then answer **Yes** to the dialog.
 - c. Select *GetDocument* and click **Convert**.
 - d. In the dialog, enter the path to the udcx file you wish to create in your data connections library. This should be of the form <http://<your-server-name>/Docs/MyDataConnections/GetDocument.udcx>. Click **OK**.
 - e. Select *SubmitDocument* and click **Convert**.
 - f. In the dialog, enter the path to the udcx file you wish to create in your data connections library. This should be of the form <http://<your-server-name>/Docs/MyDataConnections/SubmitDocument.udcx>. Click **OK**.
 - g. In the data connections dialog, click **Close**.
3. Each udcx must be approved on the server.
 - a. Navigate to the data connections library.
 - b. Click the dropdown menu that appears when hovering near **GetDocument** and choose **Approve/reject** from this menu.
 - c. Click the **Approved** option on the page that comes up. Click **OK**.
 - d. Click the dropdown menu that appears when hovering near **SubmitDocument** and choose **Approve/reject** from this menu.
 - e. Click the **Approved** option on the page that comes up. Click **OK**.

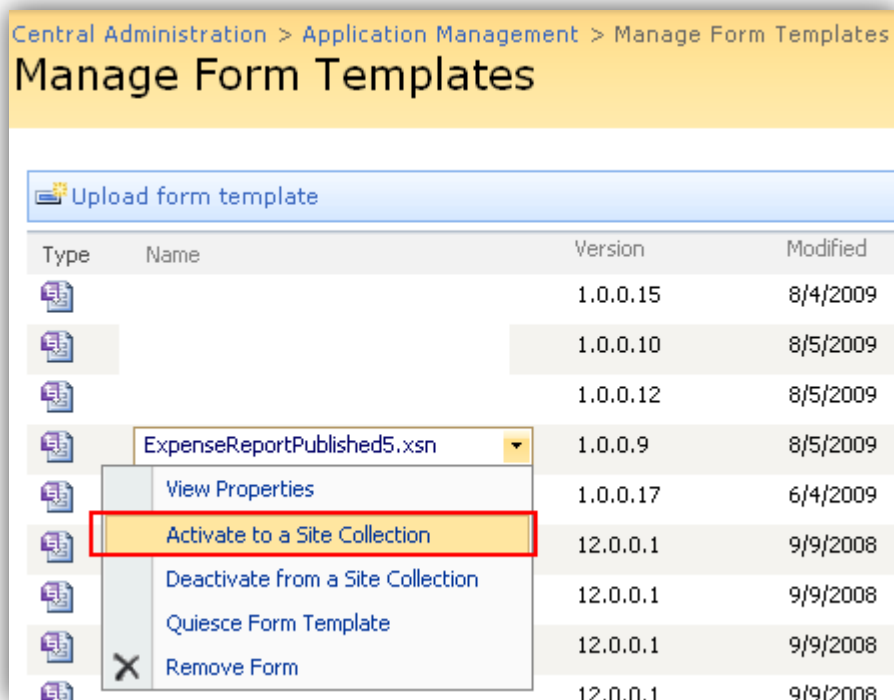


PUBLISH YOUR XSN AS AN ADMINISTRATOR-APPROVED FORM TEMPLATE

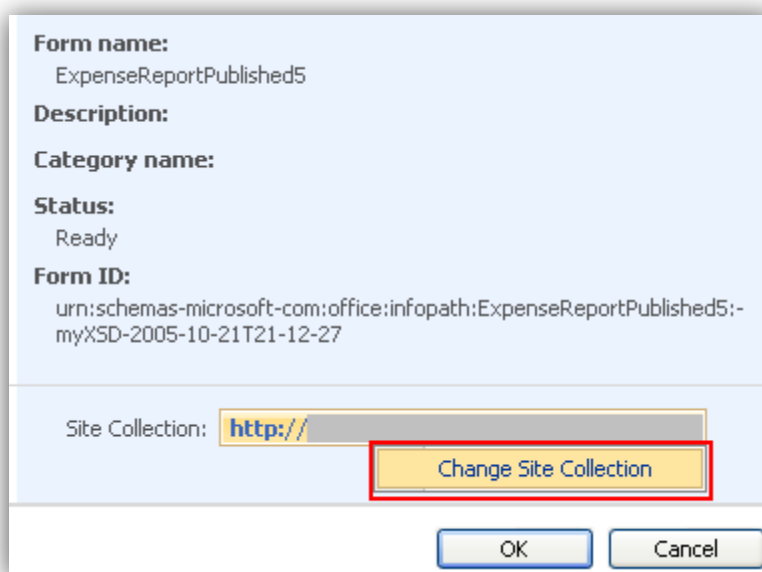
1. In the InfoPath Designer, go to the **File** menu and choose **Publish...**
2. In the publishing wizard, select **To a SharePoint server with or without InfoPath Forms Services**. Click **Next**.
3. Enter the location of your SharePoint server and click **Next**.
4. Make sure that the check box **"Enable this form to be filled out by using a browser"** is checked. Select the option **"Administrator-approved form template (advanced)"**. Click **Next**.
5. Under **Specify a location and file name for the form template**, enter a path and filename. Click **Next**.
6. Add/remove the fields that you want to be displayed as columns in your data view and click **Next**.
7. Click **Publish**, and then click **Close**.

Now that the XSN has been admin-deployed via InfoPath, we need to deploy the form template in SharePoint's Central Administration.

8. Open the **SharePoint 3.0 Central Administration** tool from the *Programs* menu on your server. This will navigate a web browser to the administration page for your server.
9. Click **Applications Management**.
10. Under **InfoPath Forms Services**, and click **Manage form templates**.
11. Click **Upload form template**.
12. Enter the file path to which you published your finished template. Click **Upload**.
13. Click **OK** on the resulting success page.
14. Click the menu that expands when hovering over the form template you just added and choose **Activate to a site collection**.



15. Click the **Site collection** selector menu and choose **Change Site Collection**. (Note: SharePoint almost always defaults to the *incorrect* site collection the first time a form template is activated.)



16. In the pop-up window, click the **Web Application** selector menu and choose **Change Web Application**.



Web Application: http://	
URL	http://
Title	Jo-Anne
Description	
Primary administrator:	
E-mail address:	Jo-Anne.Alfara@Qdabra.com
Database Name	

Change Web Application

OK Cancel

17. Click **SharePoint - 80**, and then click the **/** to activate to the entire site collection. Click **OK**. The pop-up window will disappear.

Name	URL
SharePoint - 27388	http://
SharePoint - 80	http://

URL
/

18. Click **OK**.



CREATE A DOCUMENT TYPE IN DBXL

1. Open the *DBXL Administration Tool*. This can be found by navigating to the DBXL start page on your server (e.g. <http://<your-server-name>/QdabraWebService/default.htm>) and then clicking **DBXL Administration Tool**.
2. When DAT launches, click on **New Configuration**.
3. In the **General** tab of DAT, find the **Name** text box and enter the name of your document type. As you will recall, we pasted in a function called **GetDocTypeName** that contained a hard-coded string **MyDocType**. This field is how the form template connects to DBXL. Enter **MyDocType** in the **Name** box.
4. Click on the **XSN File** attachment control and click **Attach**. Browse to the administrator-approved form template that you published in your local drive (e.g. *C:\ExpenseReport.xsn*), and attach the file.
5. Click **Save**.



CONFIGURE DOCUMENT-LEVEL PERMISSIONS IN DBXL

1. In DAT, click on the **Permissions** tab.
2. Under Document Level Permissions table, click Insert document level permissions.
3. Enter a name for the permissions (e.g. Test).
4. Enter the XPath of the field in your form that contains the Active Directory info. In this case, we will be using the Name field in Expense Report form.
5. Check **Read** and **Write** permissions.
6. Check **Enforce Permissions** check box.
7. Click on the **Save** button in DAT.

DATABASE ACCELERATOR Administration Tool

Document Type: MyDocType

Save Save As... Catalog

General Database **Permissions** SharePoint Documents Taxonomy Tree

☒ Enforce Permissions

Document Type Level Permissions

At least one role must be given Admin permission at the document type level. Otherwise a global DBXL admin is required to edit the document type. Only one user should be configured per row. Do not list users delimited with semicolons.

Name	Role Name	Add	Read	Write	L-Read	L-Write	Del	Admin
------	-----------	-----	------	-------	--------	---------	-----	-------

☒ Insert document type level permissions

Document Level Permissions

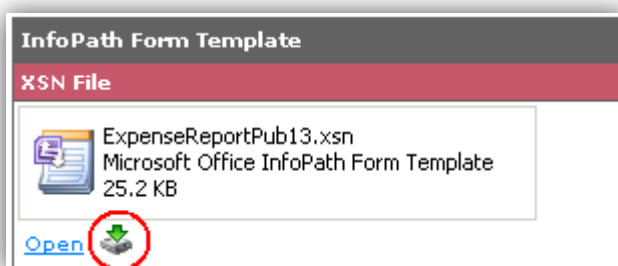
Name	Membership XPath	Read	Write	L-Read	L-Write	Del
Test	my:expenseReport/my:employee/my:name	☒	☒	☐	☒	☐

☒ Insert document level permissions



CREATE A SHAREPOINT DATA VIEW TO ACCESS YOUR DBXL FORMS

1. Submit 1-3 documents. To do this, click the download button in the **General** tab to open a blank form, fill out and submit.



You will then see your documents submitted when you go to the Documents tab:

Documents						Delete	Delete All
Doc. ID	Ref. ID	Version #	Name	Author	Description		
185	1	1	Expense Report	jo-anne			
186	2	1	Expense Report	jo-anne	test1		
187	3	1	Expense Report	jo-anne	test2		

2. Create a new SharePoint List.

The SharePoint Data View will display our forms. Please follow the steps outlined in the document called [Create a SharePoint Data View that uses GetListItems to access DBXL](#) forms to create the data view. This will require SharePoint Designer. Once you're done creating the data view using the instructions in the document, you can proceed to the steps below.

TEST YOUR SOLUTION

The steps below assume that you are a member of the **DBXLAdmins** group. You can check this in the DBXL web.config file.

To submit documents:

1. Open a new form in the browser using the URL format:
[http://<servername>/ layouts/formserver.aspx?XsnLocation=/FormServerTemplates/<template published>.xsn&OpenIn=Browser](http://<servername>/layouts/formserver.aspx?XsnLocation=/FormServerTemplates/<template published>.xsn&OpenIn=Browser)
2. Enter your Active Directory info or domain alias (e.g. <server name>\<user name>) in the Name field. Remember that we set the name field of the Expense Report form in our document level permissions.



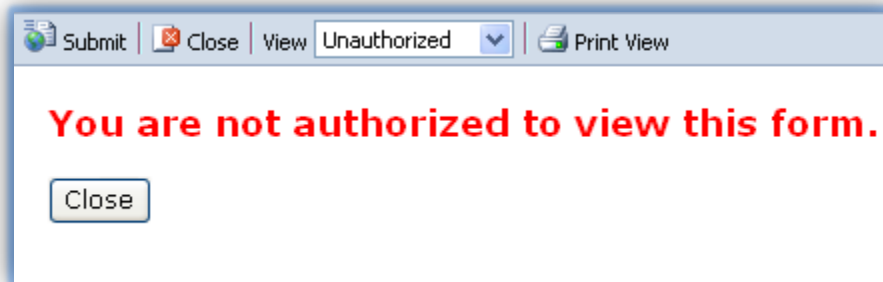
The screenshot shows an 'EXPENSE REPORT' form. It has fields for 'Report Date' (8/6/2009), 'Expense Code' (empty), and 'Business Purpose' (test7). Below these is a section titled 'Employee Information' with a 'Name' field containing '<servername>\testuser'. The 'Name' field is highlighted with a red rectangle.

3. Submit the form.
4. Fill out a second form, this time entering someone else's username.
5. In DAT **Documents** tab, click **Refresh** and note the DocIDs of the documents that you submitted.
6. In the server machine, go to **Computer Management > Local Users and Groups > Groups**. Double click on the **DBXLAdmins** group and remove your user alias from the list. Click **Apply**.
7. Refresh your data view and verify that only the document you submitted with your Active Directory info or domain alias is displayed in the view.

The screenshot shows a table titled 'ExpenseDataView7'. It has a header row with 'DocID', 'Name', 'Author', and 'Description'. Below the header, there are three rows of data. The first row has DocID 56, Name 'Expense Report', Author 'jo-anne', and Description 'test'. The second row has DocID 57, Name 'Expense Report', Author 'jo-anne', and Description 'test2'. The third row has DocID 58, Name 'Expense Report', Author 'jo-anne', and Description 'test3'.

DocID	Name	Author	Description
56	Expense Report	jo-anne	test
57	Expense Report	jo-anne	test2
58	Expense Report	jo-anne	test3

8. In your browser, open the first document you submitted (when using your user alias) in the format: http://<servername>/_layouts/formserver.aspx?XsnLocation=/FormServerTemplates/<templatepublished>.xsn&OpenIn=Browser&DocID=<DocID>
Alternatively, you can simply click the DocID link of the document.
9. Verify that form is opened without errors.
10. In your browser, open the second document you submitted (when using a different user alias) in the same format.
11. You will get an error message stating that you do not have permissions to open the document.



In the same way, you can test permissions using DBXL:

12. In Internet Explorer, type in the address
http://<servername>/QdabraWebService/Documents/<DocID>/<DocID>.xml, using the DocID for the document submitted previously, then enter.
13. Verify that you are allowed to open the first document with your user alias, but are not allowed to open the document submitted using a different user alias.

When opening an existing form using the **Open** link in the documents tab of DAT, you will get an error message: *"InfoPath cannot open the form."* To work around the problem, click the download icon () instead of the **Open** link and you should be able to view the document. Please also use this icon when opening the form template from the **General** tab.