



## Qdabra's Excel to InfoPath Tool User Guide

The Excel to InfoPath Tool provides the ability to convert row data from an Excel sheet (XLS file) to an InfoPath form (XML document). The tool uses a mapping that you specify to convert the Excel data into InfoPath forms.

This tool can be used to do onetime conversions of Excel data into XML data such as when cutting over a process from using Excel to InfoPath. The Excel to InfoPath Tool can also be used on a regular basis to interface data into an InfoPath solution. Many legacy systems support exporting data to Excel format. You can save these Excel files, create a configuration, and set up a task to process the Excel spreadsheets. Since the tool checks for duplicates, only new data rows will result in new XML documents.

Installation and setup .....	2
Configuration .....	2
<configSections> .....	2
<appSettings> .....	3
Mapping file .....	3
GeneralSettings .....	3
DbxlSubmitOptions section .....	4
FileNameInfo section .....	5
KeyParamsInfo section .....	5
Mapping section .....	6
Saving .....	8
Execution scenarios .....	8
Scenario A: Output to a local folder .....	8
Scenario B: Output directly to DBXL .....	9
Scenario C: Output to SharePoint .....	10
Support .....	18



## INSTALLATION AND SETUP

The tool is distributed through an MSI installer, available for purchase on [Qdabra.com](http://Qdabra.com).

To install the tool, run the setup.exe executable file.

By default, the installer places the tool in the directory **C:\Program Files\Qdabra Software\Qdabra Excel to XML Tool** and adds links to documentation in the Start Menu.

To verify that the installation succeeded, navigate to the folder using Windows Explorer. **Throughout this document, “working folder” will refer to the above folder.**

You will need the following to use the Excel to XML tool:

- An Excel Spreadsheet with at least one row of data
- An InfoPath form template
- The Excel to InfoPath tool installed on your computer
- A destination for the XML documents that will be generated by the tool.

Follow these steps to prepare for the data conversion:

1. Copy your Excel spreadsheet to a folder inside the working folder. We will call this the “source” folder. The folder called *HierarchySampleSolution* folder (included in the installation) is an example of a “source” folder and what its contents should be.
2. **Note that a XLSX file should be saved as XLS in order to be compatible with the tool.**
3. Open the configuration file for the tool using Notepad (or the editor of your choice). The configuration file is named XLS\_to\_XML.exe.config and can be found in the working folder.

Note: In some cases (especially on Windows 7), the Program Files folder will be protected. This means that you won't be able to edit the files inside the folder **C:\Program Files\Qdabra Software\Qdabra Excel to XML Tool**. For a simple workaround, simply move the file you need to edit (to your desktop, for example), perform the edit there, save the file, and then move it back into the **C:\Program Files\Qdabra Software\Qdabra Excel to XML Tool** folder.

## CONFIGURATION

The **XLS\_to\_XML.exe** configuration file used by the tool contains comments to help guide you.

<CONFIGSECTIONS>

This section should not be modified.



<http://www.qdabra.com>

Last updated on 4/26/2012 11:01 AM

Copyright © 2006-2010 Autonomy Systems, LLC. All rights reserved.

## <APPSETTINGS>

- **DbxlWebserviceURL:** It contains the address of the DbxlDocumentService.asmx in the DBXL instance where documents should be submitted.

```
<add key="dbxlWebserviceURL"
value="http://YOURSERVER/QdabraWebService/dbxldocumentsevice.asmx" />
```

- **LocalOutPutOnly:** If this setting is true, the form will only save forms to the local disk. Set this to false if you wish to submit the forms to DBXL.

```
<add key="LocalOutputOnly" value="false" />
```

- **UseExternalMapping:** If UserExternalMapping is true, this tool will use an external mapping file (specified in the ExternalMappingFile), to define how it should process the Excel file and generate XML. This parameter will rarely be set to false; true is the default mode.

```
<add key="UseExternalMapping" value="true" />
```

- **ExternalMappingFile:**

```
<add key="ExternalMappingFile"
value="HierarchySampleMapping.xml" />
```

- **VerboseLog:** Setting this key to true will produce extra detailed messages while the tool is running.
- There were additional keys below VerboseLog. However, these have been deprecated in the most recent version of the tool. All additional configurations are now done via the mapping form, which generates a mapping file, and is explained in the next section.

## MAPPING FILE

A mapping file is an XML configuration file which the tool uses to set parameters and create the unique mapping. In the working folder, you will see a new *Forms* folder where the XLS to XML Mapper XSN is located. This is the form used to create your mapping file.

The tool also provides a sample mapping file called **HierarchySampleMapping.xml** located in the working folder. You can use this sample as a guide when creating your own mapping. Let's discuss this mapping file in detail.

## GENERALSETTINGS

This section contains basic information like the Excel filename and output path.



<http://www.qdabra.com>

Last updated on 4/26/2012 11:01 AM

Copyright © 2006-2010 Autonomy Systems, LLC. All rights reserved.

- **ExcelFileName:** The path to the Excel file where the data should come from. This must be an XLS file (not XLSX).

Excel File Name	<i>The path to the Excel file where the data should come from. This must be an XLS file (not XLSX).</i> HierarchySampleSolution/HierarchySample.xls
-----------------	--

- **OutputDirectoryPath:** This is a relative path meaning an "output" directory will be created in the folder where you are executing the program. If you'd like to ensure that the output is always written to the same location, use an absolute path instead.

Output Directory Path	<i>The path where the XML files created should be stored.</i> hierarchyoutput/
-----------------------	---

- **StartingRow:** The row of the Excel file where processing should start. In the case of repeating data, this only applies to the main spreadsheet specified in the top-level mapping.

Starting Row	<i>The row of the Excel file where processing should start. This only applies to the main spreadsheet specified in the top-level mapping.</i> 2
--------------	--

- **LastRow:** This optional field can be used to specify a stopping row if needed. Leave the default value as '0' if you don't wish to specify the last row.

Last Row:	<i>This optional field can be used to specify a stopping row if needed. Leave the default value as '0' if you don't wish to specify the last row.</i> 0
-----------	--

- **TemplateXMLFile:** The XML file to use as a base for generated XML files.

Template XML File	<i>The XML file to use as a base for generated XML files. Open a new, blank form from the published location (DBXL or SharePoint) and save it in the "source" folder with the name <b>template.xml</b>.</i> HierarchySampleSolution/template.xml
-------------------	---

- **Template XSN:** Attach the XSN that will be used to open the XMLs generated by the tool.

Template XSN	<i>The XSN that you are opening the output XML with.</i> <input type="button" value="Click here to attach a file"/>
--------------	--

## DBXLSUBMITOPTIONS SECTION

The **DbxlSubmitOptions** section is only relevant if **LocalOutputOnly** is set to false in the config file. **This section can be omitted if not submitting to DBXL.**

- **DocumentType:** The document type where generated forms will be submitted.

Document Type	<i>The document where generated forms will be submitted</i> CompanyContactInfo
---------------	---

- **FormNameColumn:** The name of the column in the main spreadsheet whose values will be used as the Name metadata for each row.



Form Name Column	The name of the column in the main spreadsheet whose values will be used as the Name metadata for each row Company Name
------------------	--

- **UseIndexes:** Set this to true to use doctype indexes (configured in DAT) to identify forms that have already been submitted to DBXL. If this is false, this tool will use a slower method that doesn't require indexes to be configured.

Use Indexes	Set this to true to use doctype indexes (configured in DAT) to identify forms that have already been submitted to DBXL. If this is false, this tool will use a slower method that doesn't require indexes to be configured. true
-------------	---

- To allow overwriting existing documents in DBXL, you must set **Allow Document Overwrite** to **true**, and provide an xpath filter. This xpath is used to identify documents that have already been submitted.

Document Filter XPath	The following value is required if UseIndexes is false. It is an XPath to identify documents that have already been submitted, and produce no match for those that haven't. The columnname= values specified in the <KeyParamsInfo> section will be substituted for the {0} [, {1}, {2}, ...] placeholder(s). /my:myFields[my:OriginalId = '{0}']
Allow Document Overwrite	Set this to true to allow overwrite of existing documents in DBXL, determined by the XPath filter specified above. true

## FILENAMEINFO SECTION

This section defines how filenames for the generated XMLs are determined. This section should be configured in a way that the filenames generated are unique for each form.

This section may contain one or more *Field Name* fields. Each should indicate the name of a column in the main spreadsheet. The values from these columns will be concatenated (with underscores (\_)) to form a unique file names for each form. The example below yields filenames equal to 1\_Acme.xml, 2\_TheBestCompany.xml, etc.

Field Name
This section may contain one or more <FieldName> fields. Each should indicate the name of a column in the main spreadsheet. The values from these columns will be joined together with underscores (_) to produce the file names for each form.
ID
Company Name
<input checked="" type="checkbox"/> Add Field Name

## KEYPARAMSINFO SECTION


This section can contain one or more **KeyInfo** nodes. **This is only necessary if submitting to DBXL.**

- If *Use Indexes* is true, *Index Name* should indicate the name of one of the target docTypes indexes, and *Column Name* should indicate the name of a column in the main spreadsheet that corresponds to the field stored in that index.
- If *Use Indexes* is false, *Column Name* should indicate a column in the main spreadsheet whose will be used in the <DocumentFilterXPath> format string. *Index Name* can be blank in this case.



Key Params Info	
<p>This section defines how forms are located in DBXL, to check whether they have been submitted already. It can contain one or more &lt;KeyInfo&gt; nodes.</p> <ul style="list-style-type: none"> <li>- If &lt;UseIndexes&gt; is true, <i>indexname=</i> should indicate the name of one of the target doctype's indexes, and <i>columnname=</i> should indicate the name of a column in the main spreadsheet that corresponds to the field stored in that index.</li> <li>- If &lt;UseIndexes&gt; is false, <i>columnname=</i> should indicate a column in the main spreadsheet whose value will be used in the &lt;DocumentFilterXPath&gt; format string. <i>indexname=</i> can be blank in this case.</li> </ul>	
Key Info	
Index Name	Column Name
OriginalID	ID

The indexes used by the Excel to XML tool (above) must match the indexes set up in the DBXL Administration Tool (below):

Indexes <span style="float: right;"> Update</span>	
Name	XPath
OriginalID	<u>/my:myFields/my:OriginalId</u>

## MAPPING SECTION

In this section each Excel spreadsheet column is mapped to an XML field.

- **Parent Sheet:** This section specifies the mapping to the root node in the target XML and the parent sheet in the Excel file.
  - **Root:** The XPath of a node in the target XML, relative to which all other XPaths will be evaluated.
  - **Sheet Name:** Should indicate the main (top level) spreadsheet of the Excel file.

Mapping	
Root	Sheet Name
/my:myFields	MainValues

- **Fields:** The list of fields from the main spreadsheet that should be inserted into the generated XML forms.
  - **Column Name** – the name of a column in the main spreadsheet
  - **Node XPath** – the path to the corresponding XML field (relative to the *Root* node specified above).
  - **Data Type** – data type of the corresponding field in the form.
  - **Nilable** – uncheck this box if the corresponding field 'cannot be blank'.
  - **Special** – used for fields that use InfoPath functions such as *today()*.

Fields				
Column Name	Node XPath	Data Type	Nilable	Special
ID	<u>my:OriginalId</u>	string ▼	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Company Name	<u>my:CompanyName</u>	string ▼	<input checked="" type="checkbox"/>	<input type="checkbox"/>
PhoneNumber	<u>my:PhoneNumber</u>	string ▼	<input checked="" type="checkbox"/>	<input type="checkbox"/>



- **Child Sheet:** Indicates mappings to child spreadsheets in the target Excel file, to be mapped to the parent spreadsheet and into the XML forms. This section is optional.
  - **ParentKeys** - This section should contain one or more nodes.
    - **Child Field** – the name of a column in this child spreadsheet, that has values that correspond to values in a column of the parent spreadsheet (indicated with the *Parent Field* node)
    - **Parent Field** – the name of the column in the parent spreadsheet that defines the relationship of this child sheet to the parent sheet.

Parent Keys	
Child Field	Parent Field
CompanyID	ID

Follow the same guidelines as above when adding the Mapping info for the child sheets.

Mapping				
Root	Sheet Name			
my:ProductsOwned/my:Product	ProductsOwned			

Fields				
Column Name	Node XPath	Data Type	Nilable	Special
ProductName	my:Name	string	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Repeat same pattern as above when adding child sheets. Child mappings can be nested inside other child mappings to handle parent -> child -> grandchild [-> great grandchild, etc] relationships.



Child Sheet				
Parent Keys				
Child Field	Parent Field			
CompanyID	ID			
Add Parent Key				
Mapping				
Root	Sheet Name			
my:Contacts/my:Contact	ContactDetails			
Fields				
Column Name	Node XPath	Data Type	Nullable	Special
Name	my:Name	string	<input checked="" type="checkbox"/>	<input type="checkbox"/>
PhoneNumber	my:PhoneNumber	string	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Add Field				
Child Sheet				
Parent Keys				
Child Field	Parent Field			
PersonID	ID			
Add Parent Key				
Mapping				
Root	Sheet Name			
my:E-mailAddress	E-mail Addresses			
Fields				
Column Name	Node XPath	Data Type	Nullable	Special
E-mail Address	.	string	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Add Field				
Add Child Sheet (Inside Section)				
Add Child Sheet (Outside Section)				
Add Child Sheet (Outside Section)				

## SAVING

When finished creating the mapping, click the button to generate the mapping.xml file that will be used by the tool.

In addition, you can use **File > Save As** to save a backup of this configuration, in case you need to edit it at a later time.

## EXECUTION SCENARIOS

Once the source documents are ready and the configuration has been created, you are ready to proceed. Typically, you should use a test spreadsheet with just a couple rows of data to quickly verify the configuration, and then run the tool on the full data set once you have verified that the configuration is working as desired. If you perform any changes to the configuration and want to retest, remember to delete the existing documents so that new ones can be created.

### SCENARIO A: OUTPUT TO A LOCAL FOLDER

The default settings on the tool allow you to export to a folder on your machine. You can change the output folder by changing the *Output Directory Path* under *General Settings* of the mapping file:



<http://www.qdabra.com>

Last updated on 4/26/2012 11:01 AM

Copyright © 2006-2010 Autonomy Systems, LLC. All rights reserved.

Output Directory Path	<i>The path where the XML files created should be stored.</i> hierarchyoutput/
-----------------------	---

Once you have set the output path:

1. Make sure that the “output” folder is empty.
2. Open the original form and click **File > Save** to generate a new, blank form. Save it in the “source” folder with the name template.xml, and close InfoPath.
3. Open a command prompt window, change directory to the working folder and then execute XLS\_to\_XML.exe.

The log will be displayed and the file processed. Verify the documents created in the output directory. If there are any issues, you can go back and change the configuration.

## SCENARIO B: OUTPUT DIRECTLY TO DBXL

In order to store the output documents directly in DBXL:

1. If you have not done so already, create a Document Type configuration in your DBXL instance using your InfoPath form.
2. Open **XLS\_to\_XML.exe.config** in a text editor and change the value of the `LocalOutputOnly` key to `false`
3. Change the value of the `dbxlWebserviceURL` key to the DBXL URL where you wish to output XMLs to.

```
<!-- This is the address of the DbxlDocumentService.asmx in the DBXL instance where documents
should be submitted, if submitting to DBXL.-->
<add key="dbxlWebserviceURL" value="http://YOURSERVER/QdabraWebService/dbxldocumentservice.asmx"/>
<!-- If this setting is true, the form will only save forms to the local disk. Otherwise it
will try to submit them to DBXL. -->
<add key="LocalOutputOnly" value="false" />
```

4. Save and close **XLS\_to\_XML.exe.config**.
5. In the mapping file, **DbxlSubmitOptions** section, enter the document type configuration you created in step 1 – this is where generated forms will be submitted.

Document Type	<i>The document where generated forms will be submitted</i> CompanyContactInfo
---------------	---

4. Open the original form from DAT, and click **File > Save** to generate a new, blank form. Save it in the “source” folder with the name template.xml, and close InfoPath.
5. Set Indexes section in DAT. In case of the sample form, it will be OriginalID for Name, and XPath/my:myFields/my:OriginalId for XPath

Indexes <span style="float: right;">Update</span>	
Name	XPath
OriginalID	XPath/my:myFields/my:OriginalId

Insert index



6. Open a command prompt window, change directory to the working folder and then execute XLS\_to\_XML.exe.

The rows in the Excel spreadsheet will be processed, first by checking if they exist in DBXL. Every row that does not already exist as a document will be created in two locations: your local output directory and your DBXL Document Type configuration.

**Integration with SQL and SharePoint:** Because DBXL allows SQL and SharePoint mappings, an extension of this scenario is to import documents into a DBXL document type that has been configured with a SQL or SharePoint mapping. Once imported, the documents will be mapped accordingly.

**Overwriting:** You can modify your XLS to XML mapping file so that the tool updates DBXL documents that have already been uploaded instead of skipping them. It will not check which ones have been modified. It will just overwrite any that it finds matches for.

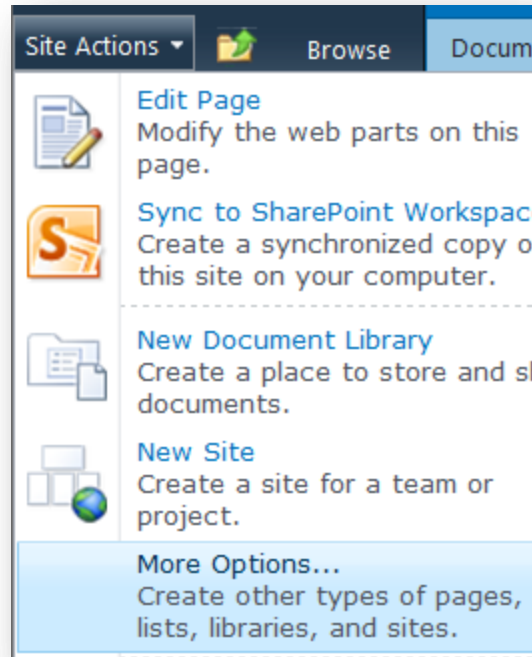
To enable overwrite, make sure that your mapping file has node called AllowDocumentOverwrite with the value true in the DbxlSubmitOptions group.

```
<ns1 :AllowDocumentOverwrite>true</ns1 :AllowDocumentOverwrite>
```

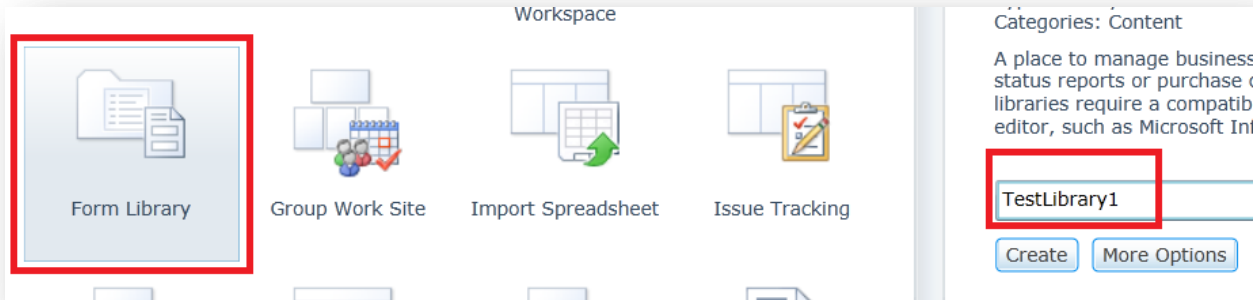
## SCENARIO C: OUTPUT TO SHAREPOINT

1. Open your SharePoint library and create a document library or form library:
  - a. Select **Site Actions > More Options...**





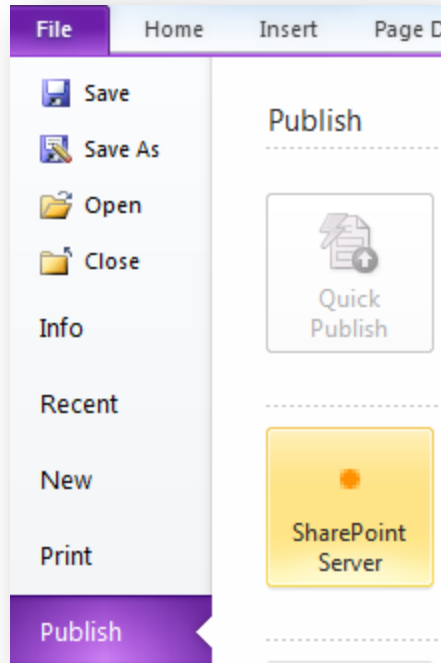
- b. Select **Form Library**, and enter a name and click **Create**.



- 2. Publish your form to the library you created above:

- a. Select **File > Publish > SharePoint Server**.





- b. Enter the location to your SharePoint site.
- c. Click **Next**.
- d. Select **Form Library**, and click **Next**.

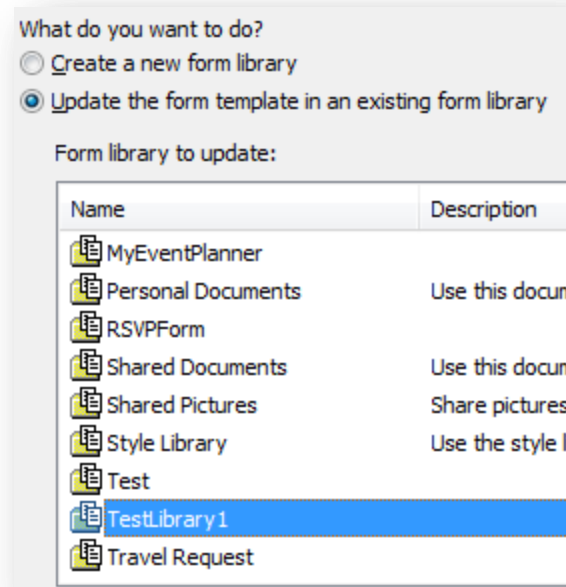
What do you want to create or modify?

**Form Library**  
 Publish this form template as a template in a form library. A form library stores forms based on this form template. Users can open and fill out forms in the library. You can specify which fields in the template appear as columns in the library.

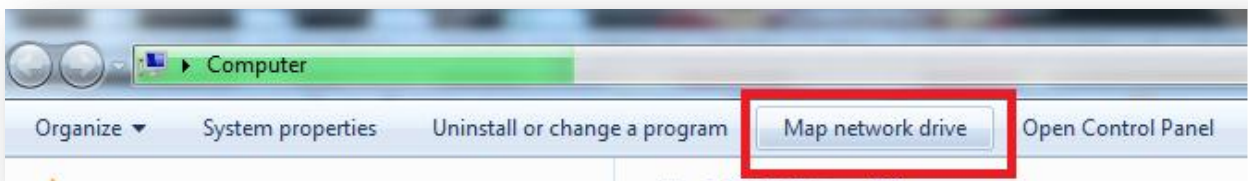
**Site Content Type (advanced)**  
 A site content type allows this form template to be used in multiple libraries and sites. You can specify which fields in the template appear as columns in the library.

- e. Select **Update the form template in an existing form library** checkbox, and select the above created form library.



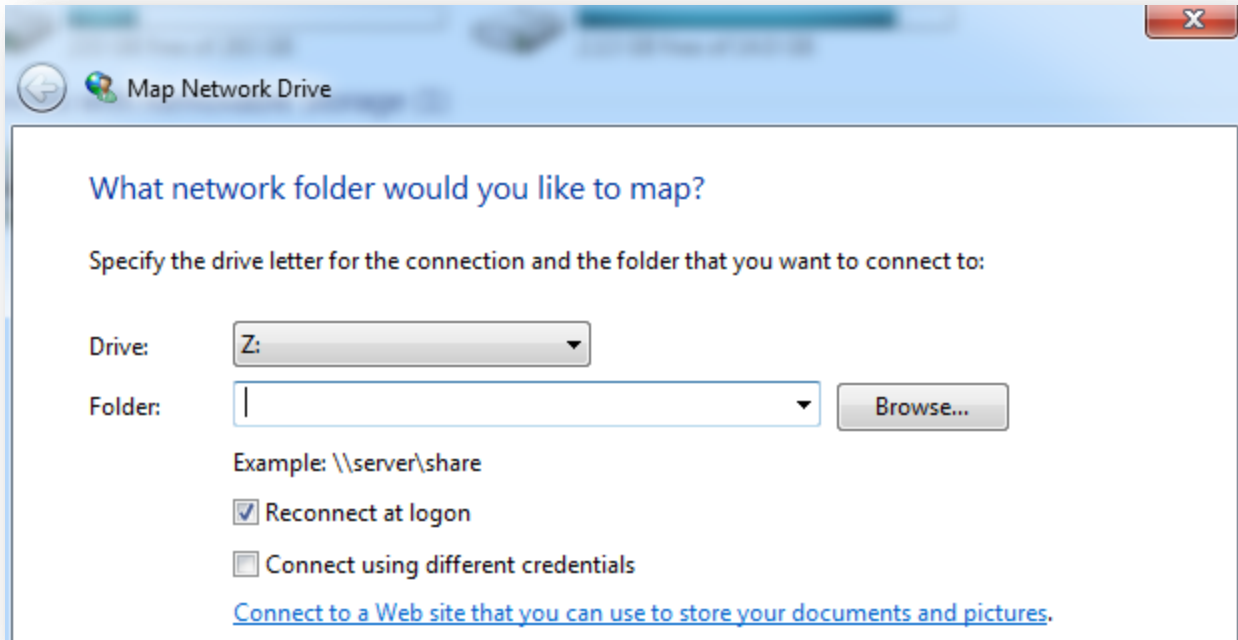


- f. Click **Next** twice, and click **Publish**.
3. Create a windows drive mapping for a SharePoint Library: Drive mapping can be created to any SharePoint library to facilitate document management using the standard windows explorer applet.
    - a. Open Internet Explorer and navigate to the above published SharePoint library.
    - b. Copy the URL up to the /Forms (ex: your URL should be similar to this one - <http://<SharePointSite>/<FormLibraryName>>).
    - c. Open windows explorer and select **Manage network drive**.



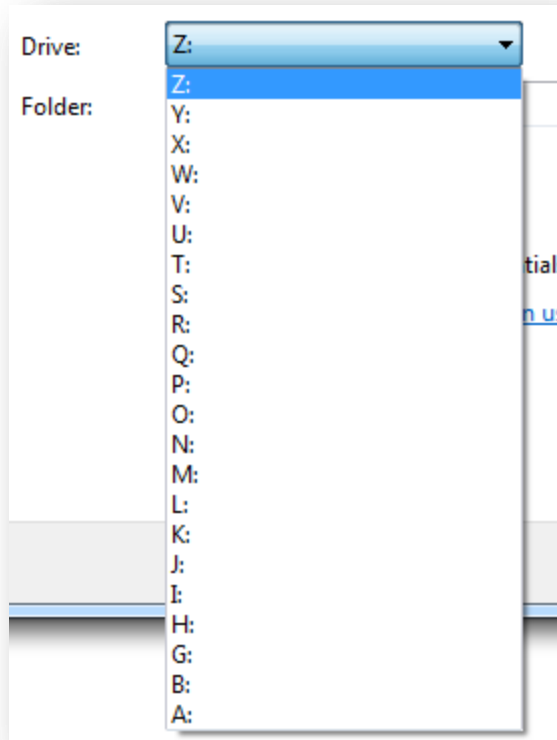
- d. You will see the **Map Network Drive** dialogue box as shown below.



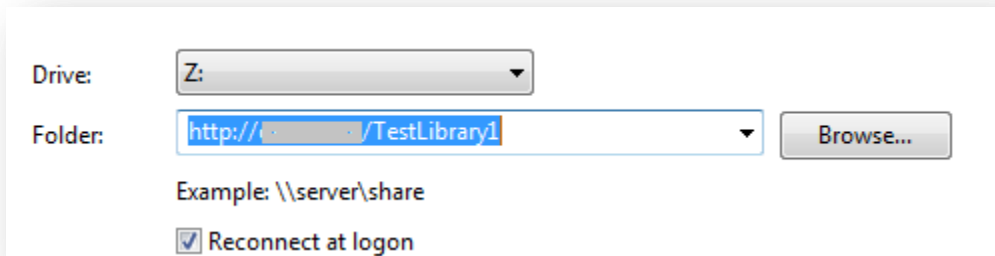


- e. Next select the drop down menu arrow to the right of the Drive entry box and select an available drive letter of your choice. This tutorial uses Z drive.



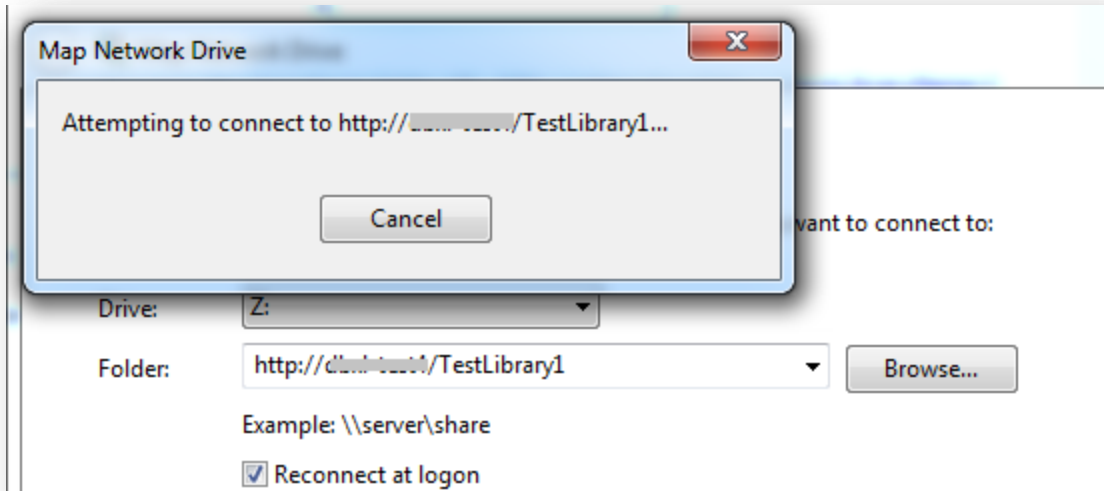


- f. Now type the SharePoint folder URL (<http://<SharePointSite>/<FormLibraryName>>).

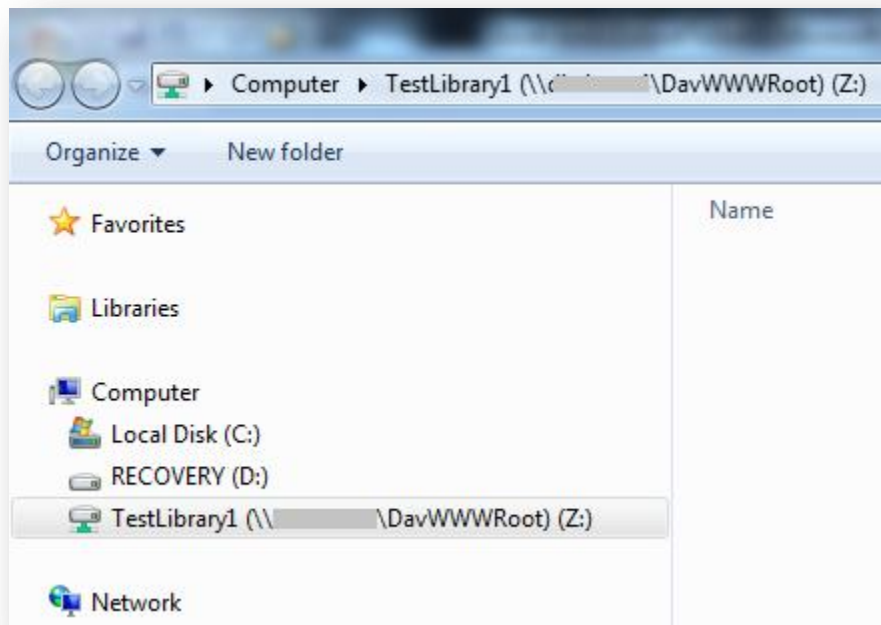


- g. Click the **Finish** button to complete the mapping process.





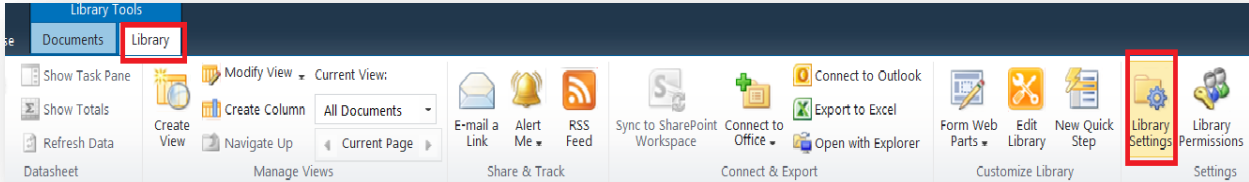
h. You should see the newly mapped drive as shown below.



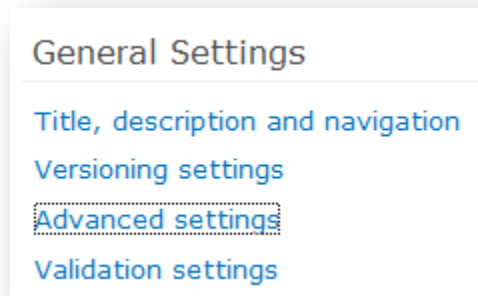
4. Open the original form from SharePoint, and click **File > Save** to generate a new, blank form. Save it in the "source" folder with the name template.xml, and close InfoPath.



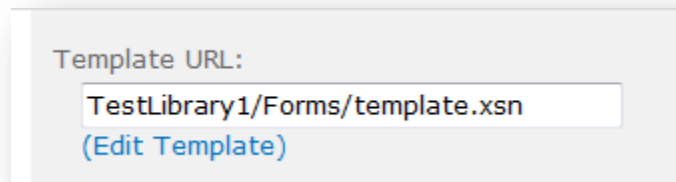
5. Update the template.xml file:
  - a. In SharePoint, open the form library that the form was published to, and click **Library > Library Settings**:



- b. Under **General Settings**, select **Advanced settings**.



- c. Copy the **Template URL** part to Notepad.



- d. Prefix the copied URL with <http://<SharePointSite>/>. So your complete URL should look like this - <http://<SharePointSite>/TestLibrary1/Forms/template.xsn>.
  - e. Open template.xml (from step 4) in Notepad and locate (near the top of the file) a line that looks like this:

```
<?mso-infoPathSolution name="<>" href="manifest.xsf" solutionVersion="<>" productVersion="<>"
PIVersion="<>" ?>
```



- f. Replace **href** attribute value with this value:  
<http://<SharePointSite>/<FormLibrary>/Forms/template.xsn>

**Note:** If you have admin-deployed your InfoPath form template, the URL above will be different. The key thing to remember is that the href in your template.xml MUST MATCH the server location where the XSN is located.

- g. Save and close template.xml.
6. Modify the Config file:
    - a. Set **LocalOutPutOnly** to **true**:

```
<add key="LocalOutputOnly" value="true" />
```

- ol style="list-style-type: none;">  - ol style="list-style-type: none;">  - b. Modify the mapping xml file to set the OutputDirectoryPath.  
<ns1:OutputDirectoryPath>Z:/</ns1:OutputDirectoryPath>
7. Open a command prompt window, change directory to the working folder and then execute XLS\_to\_XML.exe.

The rows in the Excel spreadsheet will be processed and create an output to a SharePoint folder.

## SUPPORT

If you have questions about the information in this document, please contact us for assistance.

Licensed customers can contact us via [Support@Qdabra.com](mailto:Support@Qdabra.com).

Alternatively, please use the [InfoPathDev.com Qdabra Product support forums](http://InfoPathDev.com/Qdabra/Product/support/forums) to request help.



<http://www.qdabra.com>

Last updated on 4/26/2012 11:01 AM

Copyright © 2006-2010 Autonomy Systems, LLC. All rights reserved.