



QDABRA™ EXCEL TO INFOPATH TOOL

HOW TO CONVERT REPEATING DATA FROM EXCEL TO XML

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 and outputs the InfoPath forms to a SharePoint library.

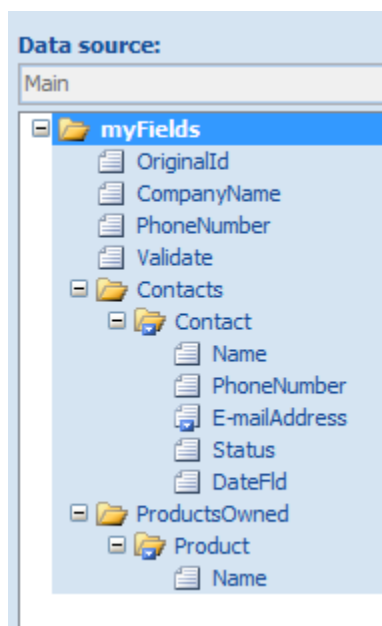
PRE-REQUISITE

The tool is distributed through an MSI installer, available for purchase on Qdabra.com. Run the setup.exe executable file to install the tool. By default, the installer places the tool in the following directory:

C:\Program Files\Qdabra Software\Qdabra Excel to XML Tool

For this tutorial you will need to have a server with **DBXL** installed.

- Create an XSN with repeating section. This tutorial uses the below schema.



- Create a new configuration in DBXL Database Accelerator tool (DAT) (this tutorial uses **CompanyContactInfo**) with the xsn created in the above step.



- Create an index similar to the below one, and click **Update** button.

Indexes Update	
Name	XPath
<u>OriginalId</u>	/my:myFields/ <u>my:OriginalId</u>

Insert index

- Click **Save** to save the configuration.

Please refer this [blog post](#) to learn more about the Excel to InfoPath tool installation and initial setup process.

EXPLANATION OF THE CONFIGURATION FILE FOR A REPEATING SECTION

The **XLS_to_XML.exe** configuration file is an XML file which the tool uses to set parameters and create the unique mapping for each solution.

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

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

- **LocalOutPutOnly** key: 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" />
```



- **UseExternalMapping** key: If UserExternalMapping is true, this tool will use an external mapping file (specified in the ExternamMappingFile, to define how it should process the Excel file and generate XML.

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

- **ExternalMappingFile** key:

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

The **SampleMapping.xml** configuration file is an XML file which the tool uses to set parameters and create the unique mapping for a repeating section. Let's discuss this mapping file in detail.

- The **GeneralSettings** section: This section contains basic information like the Excel filename, output path, etc.
 - **ExcelFileName**: The path to the Excel file where the data should come from. This must be an XLS file (not XLSX).

```
<ExcelFileName>HierarchySampleSolution/HierarchySample.xls</ExcelFileName>
```

- **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.

```
<OutputDirectoryPath>\\dbxl-Test4\ShPOutput_Repeating</OutputDirectoryPath>
```

- **StartingRow**: The row of the Excel file where processing should start. This only applies to the main spreadsheet specified in the top-level mapping.

```
<StartingRow>2</StartingRow>
```

- **TemplateXMLFile**: The XML file to use as a base for generated XML files. Typically this can be obtained just by extracting the XSN as its source files.

```
<TemplateXMLFile>HierarchySampleSolution/template.xml</TemplateXMLFile>
```

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

```
<DocumentType>CompanyContactInfo</DocumentType>
```

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

```
<FormNameColumn>Company Name</FormNameColumn>
```

- **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.



```
<UseIndexes>>true</UseIndexes>
```

- The **FileNameInfo** section: 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 concatenated (with underscores (_)) to form a unique file names for each form. The below Config yields to 1_Acme.xml, 2_ TheBestCompany.xml, etc.

```
<FieldName>ID</FieldName>  
<FieldName>Company Name</FieldName>
```

- The **KeyParamsInfo** section: This section can contain one or more <KeyInfo> nodes. If <UseIndexes> is true, indexname= should indicate the name of one of the target docTypes indexes, and columnname= should indicate the name of a column in the main spreadsheet that corresponds to the field stored in that index. If <UseIndexes> is false, columnname= should indicate a column in the main spreadsheet whose will be used in the <DocumentFilterXPath> format string. indexname= can be blank in this case.

```
<KeyInfo indexname="OriginalId" columnname="ID"/>
```

- The **Mapping** section: In this section each Excel spreadsheet column is mapped to an XML field. The main mapping sheetname= should indicate the main (top level) spreadsheet of the Excel file, root= is the XPath of a node in the target XML, relative to which all other XPaths will be evaluated.

The basic format that this section follows is:

```
<Fields>  
  <!-- The list of fields from the main spreadsheet that should be inserted  
  into the generated xml forms. name= is the name of a column in the main  
  spreadsheet, and xpath is the path to the corresponding XML field (relative  
  to the root= attribute of this <Mapping> node). -->  
  <Field name="" xpath="" />  
  <Field name="" xpath="" />  
  ...  
  ...  
</Fields>  
  <!-- 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. -->  
  <ChildSheets>  
    <!-- Indicates a mapping between a child spreadsheet and a parent  
    spreadsheet. -->  
    <ChildSheet>  
      <!-- Defines how the child spreadsheet is related to the parent  
      spreadsheet. -->  
      <ParentKeys>  
        <!-- This section should contain one or more <ParentKey> nodes  
        childfield= is 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 parentfield= attribute) -->  
        <ParentKey childfield="" parentfield="" />
```



```
</ParentKeys>
<!-- This indicates where data located by this mapping should be
inserted into the form, root= indicates the location in the XML where the
data located should be inserted. This XPath should be relative to the root=
attribute of the parent mapping, and should be a repeating node.
sheetname= is the name of the spreadsheet that corresponds to this child
mapping.-->
  <Mapping root="" sheetname="">
    <Fields>
      <!-- name= should be the name of a column in this child
spreadsheet, xpath= should be relative to the root= XPath of this child
mapping. -->
      <Field name="" xpath="" />
      <Field name="" xpath="" />
      ...
    </Fields>
  </Mapping>
</ChildSheets>
  <ChildSheet>
    <!--<Repeat same pattern as above. Child mappings can be nested
inside other child mappings to handle parent -> child -> grandchild [-> great
grand child, etc] relationships.-->
    </ChildSheet>
    <!--<Repeat another child sheet in the above manner>-->
  </ChildSheet>
</ChildSheets>
```

The mappings for the sample included in the tool looks like this:

```
<Mapping root="/my:myFields" sheetname="MainValues">
  <Fields>
    <Field name="ID" xpath="my:OriginalId"/>
    <Field name="Company Name" xpath="my:CompanyName"/>
    <Field name="PhoneNumber" xpath="my:PhoneNumber"/>
    <Field name="Validate" xpath="my:Validate" type="boolean" />
  </Fields>
</ChildSheets>
  <ChildSheet>
    <ParentKeys>
      <ParentKey childfield="CompanyID" parentfield="ID"/>
    </ParentKeys>
    <Mapping root="my:ProductsOwned/my:Product" sheetname="ProductsOwned">
      <Fields>
        <Field name="ProductName" xpath="my:Name"/>
      </Fields>
    </Mapping>
  </ChildSheet>
  <ChildSheet>
    <ParentKeys>
      <ParentKey childfield="CompanyID" parentfield="ID"/>
    </ParentKeys>
```



```
<Mapping root="my:Contacts/my:Contact" sheetname="ContactDetails">
  <Fields>
    <Field name="Name" xpath="my:Name"/>
    <Field name="PhoneNumber" xpath="my:PhoneNumber"/>
    <Field name="Status" xpath="my:Status"/>
    <Field name="DateFld" xpath="my:DateFld" nillable="true"
type="date"/>
  </Fields>
  <ChildSheets>
    <ChildSheet>
      <ParentKeys>
        <ParentKey childfield="PersonID" parentfield="ID"/>
      </ParentKeys>
      <Mapping root="my:E-mailAddress" sheetname="E-mail Addresses">
        <Fields>
          <Field name="E-mail Address" xpath="."/>
        </Fields>
      </Mapping>
    </ChildSheet>
  </ChildSheets>
</Mapping>
```

EXECUTION

If you carried all the above steps successfully, then you are ready to submit documents to DBXL.

- Open a command prompt window.
- Navigate to the working folder.
- Execute XLS_to_XML.exe.

Now, the rows in the Excel spreadsheets will be processed, first by checking if they exist in DBXL. Every row that does not already exist as a document will be (a) created in your local output directory and (b) saved to your DBXL Document Type configuration.

Open DAT to verify the submitted documents.



Document Type: CompanyContactInfo Save Save As... Catalog

General Database Permissions SharePoint Documents Taxonomy Tree

Export Documents

Import Documents

Query Documents

To limit the number of documents shown, enter a starting Doc. ID and the number of documents to show, and then click query. You can then jump forward or backward by the number of shown documents, or click Query All to show all documents. To search for a Document with a specific Ref. ID, type the ID into the field, and then click Query.

Starting Doc. ID: 1477 Ref. ID: 1 Range: 20 Previous Next Query Query All

Refresh

Documents Delete Delete All Reshred Reshred All

Doc. ID	Ref. ID	Version #	Name	Author	Description	New	
1477	175	1	Acme, Inc.	Excel prepop		Open	
1478	176	1	The Best Company	Excel prepop		Open	
1479	177	1	Widgets, Ltd.	Excel prepop		Open	
1480	178	1	Sprockets Corp.	Excel prepop		Open	
1481	179	1	mycompany	Excel prepop		Open	
1482	180	1	my new company	Excel prepop		Open	
1483	181	1	another company	Excel prepop		Open	
1484	182	1	new entry	Excel prepop		Open	