



QDABRA DATABASE ACCELERATOR V2.3

Adding a Custom SharePoint Column

OVERVIEW

There are situations where you need to add a custom field which cannot be accomplished through the SharePoint UI. One possible way to accomplish this is via SharePoint designer. However, SharePoint has an UpdateList webservice which allows fully customizable columns. Though it carries a high-level of complexity, our SharePoint-UpdateList tool provides a way to add/delete custom columns by using a column definition file.

In this document we will address one specific example that applies to DBXL users. When you create a SharePoint mapping from the DBXL Administration Tool, SharePoint uses the DocID (an integer) as the link to open the document from the SharePoint form library. The example below will show you how to add a custom column that serves as a link.

SHAREPOINT-UPDATELIST TOOL

The SharePoint UpdateList Tool is a command line tool that comes included in this zip package.

Usage:

```
shpupdList -site siteurl -list listname -output xmlfilename -input fielddef.xml -output listxmlout.xml [-delete]
```

Arguments

- Site: The site url which contains the list being modified
- List: The name of the list being modified
- Output: (optional) The name of the output file to write the SharePoint list definition.
- Input: Xml file containing the column definition
- Delete: Parameter indicating to delete the column specified. Default is to add the column.

ADDING A COLUMN BASED ON AN EXISTING COLUMN

A common scenario is making a column similar to an existing column, but making minor html change in how it is displayed. By using the list definition written to the output file, you can obtain the current definition of a column to base your new column on. Using an xml editor, you can copy the field definition and make necessary changes to the DisplayPattern node to customize the display format. The key issue is to make the



Name field unique within the list. If the *Name* field is not unique, SharePoint will automatically append a number, starting at 0; so if you used Title, it would name it Title0.

CREATE A CUSTOM COLUMN FOR YOUR DBXL SHAREPOINT MAPPING

To display a field from the form as a link in the SharePoint form library will require that you first promote the field in the form.

1. Open your InfoPath form in Design mode. Go to **Tools > Form Options > Property Promotion**. Promote form fields as desired. Keep in mind that SharePoint reserves certain names, such as *Name* and *Description*. Therefore, you want to use more unique Column names.

If you haven't done so already, you'll need to create a DBXL document type and add a SharePoint mapping.

2. Open the DBXL Administration Tool, click **New Configuration** and create a new document type.



For more information on creating a document type, please see the **Getting Started** document included in every installation of DBXL. If you have an existing document type, simply attach the updated XSN.

3. In DAT, go to the **SharePoint** tab. Create a SharePoint mapping, making sure to check the **Publish Library and XSN** checkbox. Verify that the fields you promoted are shown on this screen.

The screenshot shows the 'Administration Tool' interface for 'Database Accelerator'. The 'SharePoint' tab is selected. The 'SharePoint List or Library (i.e. Collection)' section is active, showing a configuration for 'CreateaCustomSharePointColumn'. The 'URL' is 'http://dbxl-test4/CreateaCustomSharePointColumn' and the 'Method' is 'Object Model'. The 'Expression' field contains 'Create a Sharepoint Mapping'. The 'Operator' field is empty. The 'Publish Library and XSN' checkbox is checked. The 'Use Forms Services' and 'Propagate Deletes' checkboxes are unchecked. Below this, the 'Promoted Properties' table is displayed, showing three properties: 'FormName', 'FormDescription', and 'FormAuthor'. The 'FormName' property is highlighted with a blue box.

Promoted Property Name	XPath	Column Type	Aggregation Command
FormName	/my:myFields/my:name	string	
FormDescription	/my:myFields/my:description	string	
FormAuthor	/my:myFields/my:author	string	

4. Next, change the xml to use that column instead of the title column. To do this, find all instances of "Title" and change these to the Promoted Property Name you wish to use as the linked column. You will also need to change the **hyperlink** in the xml document to your Qdabra web server name.

The zip package you have downloaded includes an unmodified xml file called TitleDbxlLinkColumn.xml. To run this scenario, I replaced the three instances of "Title" with "FormName", as seen below. In our scenario, notice that **FormName** matches the promoted property we created in Step 1.

```
<?xml version="1.0" encoding="utf-8"?>
<List>
  <Fields>
    <Method ID="1">
      <Field Name="DbxlLinkedTitle" Type="Computed" DisplayName="Title"
      AuthoringInfo="(linked to dbxl document)"
      ReadOnly="TRUE" FromBaseType="FALSE" HIDDEN="TRUE">
        <FieldRefs>
          <!-- <FieldRef Name="Description0" /> -->
          <FieldRef Name="FileLeafRef" />
        </FieldRefs>
      </Field>
    </Method>
  </Fields>
</List>
```



```
<FieldRef Name="FSObjType" />
<FieldRef Name="Created_x0020_Date" />
<FieldRef Name="FileRef" />
<FieldRef Name="ContentTypeId" />
<FieldRef Name="PermMask" />
<FieldRef Name="Title" />
</FieldRefs>
<DisplayPattern>
  <!-- BEGIN: This is where the DBXL Link is injected-->
  <HTML><![CDATA[<A onfocus="OnLink(this)" HREF="http://DBXL-
TEST4/QdabraWebService/Documents/]]></HTML>
  <UrlBaseName HTMLEncode="TRUE">
    <LookupColumn Name="FileLeafRef" />
  </UrlBaseName>
  <HTML></HTML>
  <UrlBaseName HTMLEncode="TRUE">
    <LookupColumn Name="FileLeafRef" />
  </UrlBaseName>
  <HTML><![CDATA[.xml"]]></HTML>
  <!-- END: This is where the DBXL Link is injected-->
  <IfEqual>
    <Expr1>
      <Column Name="Description0" />
    </Expr1>
    <Expr2></Expr2>
    <Then>
      <UrlBaseName HTMLEncode="TRUE">
        <LookupColumn Name="Title" />
      </UrlBaseName>
    </Then>
    <Else>
      <Column Name="Description0" />
    </Else>
  </IfEqual>
  <HTML><![CDATA[</A>]]></HTML>
</DisplayPattern>
</Field>
</Method>
</Fields>
</List>
```

5. Run the command line tool. To do this, open a command prompt and run the following command:

```
shpUpdList.exe -site http://dbxl-test4/ -list CreatedCustomSharePointColumn -input TitleDbxlLinkColumn.xml
```

You'll see an output similar to the following:

```
Processing TitleDbxlLinkColumn.xml
Adding fields...
add field Name="DbxlLinkedTitle" DisplayName="DbxlLinkedTitle"
update field Name="DbxlLinkedTitle" DisplayName="FormName"
Total elapsed time: 13.80
```

Though the column has been added, we still need to modify the view.

6. In Internet Explorer, navigate to your SharePoint Form Library. Click **Settings > Form Library Settings**. Under the section called **Views**, click on **All Documents**.
7. Under Columns, check the **FormName (linked to dbxl document)** column. You can also uncheck other columns, or reorganize them as desired. When done, click **OK**.



You'll be returned to the Form Library and you'll see the new, linked column.

FormName	Modified	Modified By	Checked Out To	FormDescription	FormAuthor
name1 ! NEW	2/2/2010 6:12 AM	Ernesto Machado (Test)		description1	author1
new from sharepoint ! NEW	2/2/2010 6:14 AM	Ernesto Machado (Test)		description2	ernesto

MODIFY OR DELETE A COLUMN

If you want to edit a column, you must first delete it.

To delete a column, simply append the **-delete** parameter at the end of your command line command, using the same xml file that was used to create the custom column.

Once deleted, you can recreate the column as desired following the same steps detailed above.

OTHER SCENARIOS

The xml form used by the command line tool can be modified to fit different scenarios. If you have a scenario not covered in this document, such as browser forms or forms with code, please contact us at support@qdabra.com for assistance.