QDABRA QRULES

# Use qRules to submit to DBXL

qRules is intended for anyone who would like to leverage the power of Microsoft Office InfoPath without writing code. The library provides a set of advanced commands that can be used declaratively via rules.

## INTRODUCTION

This document provides an example on how to use the *SubmitToDBXL* qRules command. The *SubmitToDBXL* command submits the document to Qdabra DBXL. For this reason, this command requires Qdabra's DBXL.

The basic command **SubmitToDbxl** takes the following arguments:

- **submit**: name of the *query* (receive) data connection to DBXL
- **name:** form name (optional)
- **author:** form author (optional)
- **desc:** form description (optional)
- **dsname:** optional
- **doctype:** optional

An example of this command is:
*SubmitToDbxl /submit=SubmitDocument /name=name /author=author /desc=description*

However, note that in that example, all of the arguments are literal strings. If you need to specify xpath values for *name*, *author* and *description*, make sure to use a concat operation, as shown in this next example:

*concat("SubmitToDbxl /submit=SubmitDocument /name=", field1, " /author=", field2, " /desc=", field3)*

The command has two additional parameters, both optional. /dsname allows you to submit a datasource other than the main data source to DBXL. This can be used when you have child doctypes and have added their XML as a secondary data source.

/doctype allows you to explicitly specify the doctype to submit to. This is necessary when the template is going to be accessed from somewhere other than DBXL (like SharePoint) or when you're submitting a data source other than the main data source.

These are advanced scenarios, not covered by this document.

Please see the qRules User Guide included in the installation of qRules to get additional details. This tutorial will walk through the two examples above, but it assumes that you have read the User Guide first.

## CREATE AND INJECT THE FORM

A. Create a new form.

   1. Create a new, blank form in InfoPath.

2. Click **File** | **Save As** and save the form to a location on the same machine where qRules was installed.

3. Close InfoPath.

B. Add qRules to your solution:

4. Launch the tool by clicking on **Start** > **Programs** > **Qdabra** > **Tools** > **qRules** > **qRules Injector**.

5. Click on the **Browse** button and select the XSN file saved in step 1.

6. Click the **Inject** button.

7. Click **OK** in the confirmation dialog and close the Injector.

C. Create a query (receive) data connection to the Submit Document method of the DBXLDocumentService Web service

8. Open the form in Design mode in InfoPath.

9. Click **Tools** | **Data Connections** and click **Add**.

10. Select **Receive data** and click **Next**.

11. Select **Web service** and click **Next**.

12. Enter the URL to the location of DbxlDocumentService, usually *http://<servername>/QdabraWebService/DbxlDocumentService.asmx*, and click **Next**.

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

14. You do not need to set any values for this data connection. Click **Next** twice.

15. Uncheck the checkbox for **Automatically retrieve data when form is opened** and click **Finish**.

16. Click **Close** in the Data Connections dialog.

## DESIGN THE FORM

D. Design the form and add the qRules command

17. In the **Controls** taskpane, select **Text Box** and **Button**, in order to insert these into the view.

18. In the **Data Source** taskpane, right click on the one available node and select **Copy XPath**. We will use this in the next step, so you may want to temporarily paste it into Notepad.

19. Double click on the button and change the Label to **SUBMIT**.

20. Click on **Rules**, then click **Add** and then click **Add Action**.

21. Select **Set a field's value** from the **Action** dropdown.

22. For **Field**, select the **Command** node from the secondary data connection called **QdabraRules**.

23. For **Value**, enter ***SubmitToDbxl /submit=SubmitDocument /name=name /author=author /desc=description***. If you named your data connection something besides *SubmitDocument*, please make sure to use that name here.

24. In the Data Source taskpane, select the secondary data source for the SubmitDocument data connection.

25. Drag and drop the **DocID** node into the view (*/dfs:myFields/dfs:dataFields/tns:SubmitDocumentResponse/tns:docId*). Place it under the button that was previously added.

**Extra credit**: The qRules command returns the DocID of your new document, in case you need to use it for other actions in the form.

26. Save the form and close InfoPath.

E. Create the document type and test

27. In DAT, create a new DocType using the form you have created.

28. Click on the icon (⬇) to open the form in InfoPath.

29. Enter a value into the field and click the button. The DocID of the submitted document will be displayed in the DocID text box!

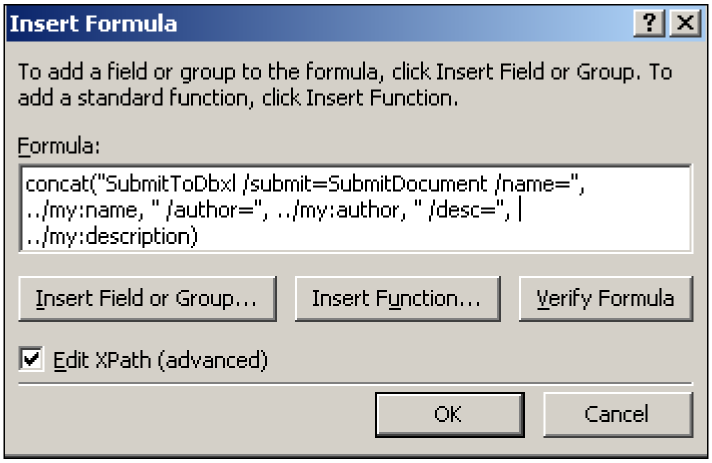
The problem: the values for /*name*, /*author* and /*desc* are hardcoded in our command. **How can we make them dynamic?**

## CONCAT THE COMMAND

F. Modify the form

30. Add three fields (text boxes) to your form: Name, Author and Description.

31. Modify the rule that sets the qRules command, such that it uses those fields in a concat() function, like this: *concat("SubmitToDbxl /submit=SubmitDocument /name=", name, " /author=", author, " /desc=", description)*. The underline indicates that this is the field that you created in step 6. If you click the **Edit Xpath (advanced)** checkbox, you'll see the full xpaths of the fields.



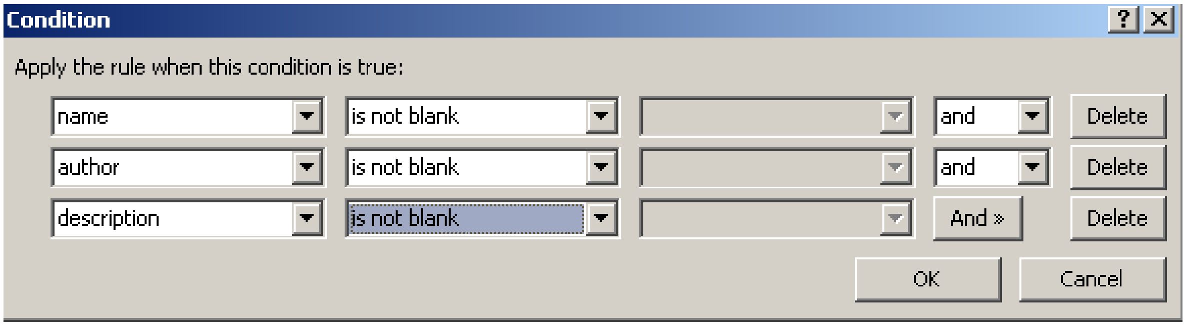
Insert Formula
To add a field or group to the formula, click Insert Field or Group. To add a standard function, click Insert Function.
Formula:

```
concat("SubmitToDbxl /submit=SubmitDocument /name=",
../my:name, " /author=", ../my:author, " /desc=",
../my:description)
```

Insert Field or Group...
Insert Function...
Verify Formula
☑ Edit XPath (advanced)
OK
Cancel

Note that you cannot use concat if any of the parameters are rich text. In that scenario, please use the parameters introduced in qRules v2.4 (explained in the qRules User Guide). For example: *SubmitToDbxl /submit=DBXLSubmit /namexpath=/my:myFields/my:AssignedToName /descxpath=/my:myFields/my:Comments*

**Extra credit**: On your button's rule, create a condition that requires the fields (*name*, *author* and *description*) to NOT be blank, like this:



| | | | | |
|---|---|---|---|---|
| name | is not blank | | and | Delete |
| author | is not blank | | and | Delete |
| description | is not blank | | And » | Delete |

G. Update the DBXL Document Type and test

32. Save the form template and re-attach it to your DBXL document type in DAT.

33. Save the DBXL Document Type and click the icon () to open the form in InfoPath.

34. Fill out the form fields (*name*, *author*, *description*) and click the **Submit** button.

35. In DAT, switch to the **Documents** tab to see your new document, and the *name*, *author* and *description* values in the respective columns.

## SUBMIT AND CONTINUE EDITING

The **Getting Started** document that comes with DBXL includes steps to create a button that submits to DBXL. However, those steps force the form to close when you submit. When the form is re-opened, it will contain the Qdabra Processing Instructions (PI).

What about creating a button that submits and allows the user to continue working on the form? This is where qRules comes in handy. You can submit via qRules, continue working on the form, and submit again: a duplicate is not created!

**Extra credit**: Use the RemoveDbxlPi command to remove the Qdabra Processing Instructions. When you click *Submit* AFTER running the RemoveDbxlPi command, the form will be considered new. In other words, this allows you to create a new document identical to an existing document.