



QDABRA QRULES

Use GetInputParameter to pass data from one form to another

The qRules command **GetInputParameter** returns the value of the key IF it exists.

The command's syntax is **GetInputParameter /key=<keyname>**. In this tutorial, will use this command to push data from one form into another form. We will link two of the sample forms included in InfoPath 2007.

TRAVEL REQUEST

1. Launch InfoPath 2007, select **Design a Form template** and select **Customize a sample**. Select **Travel Request**, change its compatibility to InfoPath (Sample forms default to browser) and save it locally.
2. Click on **Tools > Data Connections**. Click **Add**. Create a new connection to submit data to SharePoint. When asked for the library, enter the URL for a library we will create. Finish the Wizard to create the data connection.

Document library:

File name:
 

Example: Status report or concat("Status Report - ", field1)

☒ Allow overwrite if file exists

3. Insert a button into the form and configure it to submit to this new data connection.



The image shows two overlapping dialog boxes from the Qdabra software. The top dialog is titled 'Button Properties' and has tabs for 'General', 'Display', 'Size', 'Advanced', and 'Browser forms'. The 'General' tab is active, showing a 'Button' section with 'Action' set to 'Submit' and 'Label' set to 'Submit'. There is a 'Submit Options...' button and a text box explaining that the Submit action is for a single data source. The bottom dialog is titled 'Submit Options' and contains configuration settings for the submit command. It includes a checked checkbox 'Allow users to submit this form', radio buttons for 'Send form data to a single destination' (selected) and 'Perform custom action using Rules'/'Perform custom action using Code'. The 'Send form data to a single destination' option has a dropdown menu set to 'SharePoint document library'. Below this is a section 'Choose a data connection for submit:' with a dropdown set to 'SharePoint Library Submit' and buttons for 'Add...', 'Manage...', 'Rules...', and 'Edit Code...'. At the bottom, there is a checked checkbox 'Show the Submit menu item and the Submit toolbar button' with a 'Caption:' field set to 'Su&bmit'. Both dialogs have 'OK', 'Cancel', and 'Apply' buttons.

4. Go to **File > Publish** and deploy the Travel Request form to create a new SharePoint Form Library by the same name used above (TravelRequest).



Document library:

File name:
 

Example: Status report or concat("Status Report - ", field1)

☒ Allow overwrite if file exists

5. Using Internet Explorer, go to the SharePoint library and click New. Fill out the form and submit to test.
6. Repeat the same steps (1 through 6) for the Expense Report sample.

PREPARE THE TRAVEL REQUEST FORM

7. In Internet Explorer, navigate to the Expense Report library, go to **Actions** and select **Open with Windows Explorer**. From here you'll be able to decipher the URL to the xsn: <http://dbxl-test4/ExpenseReport/Forms/template.xsn>
8. In the TravelRequest form, add a field called ExpenseReportURL.

Add Field or Group  

Name:

Type:

Data type:

Data namespace:

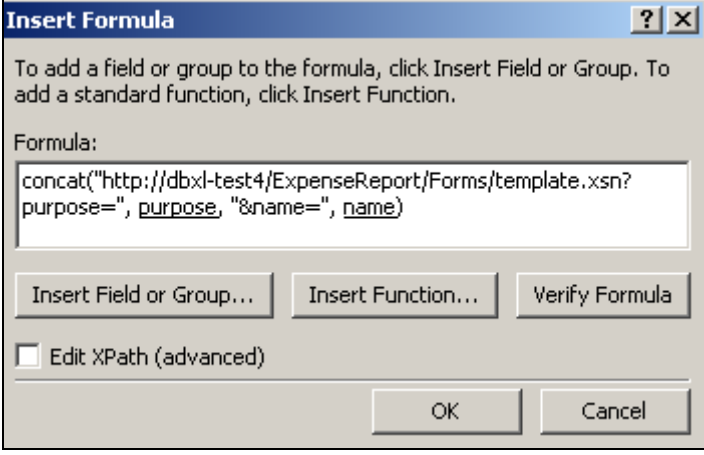
Default value:

Example: Sample Text

☐ Repeating

☐ Cannot be blank (*)

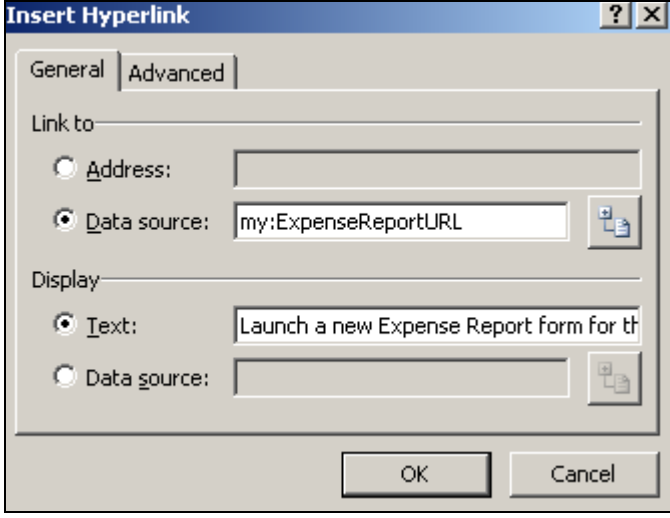
9. Double click on the new xml node in order to set the **Default Value**. In this default value, we will generate the parameters that the form will pass to the ExpenseReport form.



The 'Insert Formula' dialog box is shown. It has a title bar with a question mark and a close button. The main text says: 'To add a field or group to the formula, click Insert Field or Group. To add a standard function, click Insert Function.' Below this is a 'Formula:' label and a text box containing the formula: `concat("http://dbxl-test4/ExpenseReport/Forms/template.xsn?purpose=", purpose, "&name=", name)`. There are three buttons: 'Insert Field or Group...', 'Insert Function...', and 'Verify Formula'. At the bottom, there is a checkbox labeled 'Edit XPath (advanced)' which is currently unchecked, and 'OK' and 'Cancel' buttons.

If we check the checkbox for Edit XPath, we'll see the full paths: `concat("http://dbxl-test4/ExpenseReport/Forms/template.xsn?purpose=", ../my:purpose, "&name=", ../my:name)`

10. Insert a **Hyperlink** control to link to this URL.

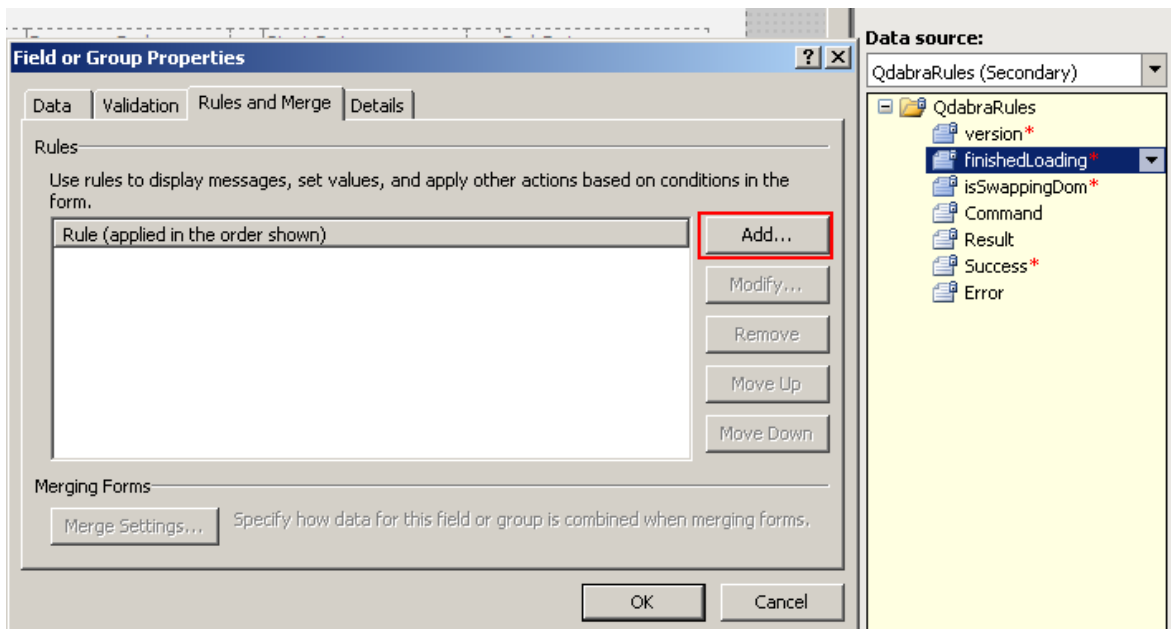


The 'Insert Hyperlink' dialog box is shown. It has a title bar with a question mark and a close button. There are two tabs: 'General' and 'Advanced'. The 'General' tab is selected. Under 'Link to', there are two radio buttons: 'Address:' and 'Data source:'. The 'Data source:' radio button is selected, and the text box next to it contains 'my:ExpenseReportURL'. Under 'Display', there are two radio buttons: 'Text:' and 'Data source:'. The 'Text:' radio button is selected, and the text box next to it contains 'Launch a new Expense Report form for th'. There are 'OK' and 'Cancel' buttons at the bottom.

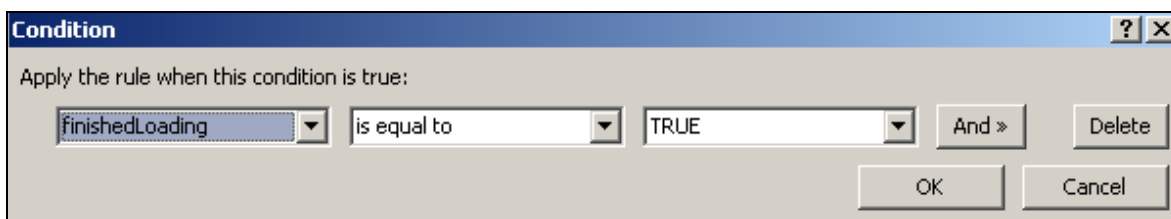
11. Save the form and redeploy the form to your SharePoint Library.

INCORPORATE QRULES

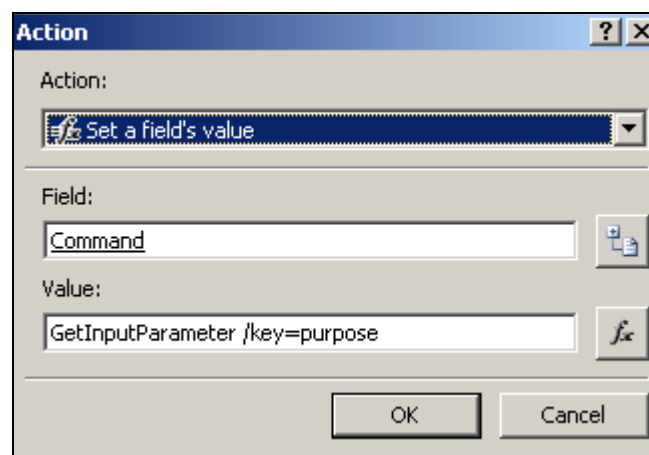
12. Launch the qRules Injector and inject qRules into the Expense Report form. (Make sure that the form is closed before injecting.)
13. Open the **ExpenseReport** form in InfoPath's design mode.
14. We will now add a rule to the **finishedLoading** field in the **QdabraRules** secondary data source.



- a. For this rule, add a condition such that it only executes when ***finishedLoading*** is TRUE.



- b. The first action will set the command to get our first key, as seen below:



- c. The second action sets the ***purpose*** field in the form to the value of the ***Result***.



Action

Action: Set a field's value

Field: purpose

Value: Result

OK Cancel

Repeat these two actions for every parameter. In our case, we are passing two parameters, so we have a total of four actions.

Rule

Name: Get parameters when finished loading

Condition: finishedLoading = TRUE Set Condition...

Actions (run in the order shown):

- Set a field's value: Command = "GetInputParameter /key=purpose"
- Set a field's value: purpose = Result
- Set a field's value: Command = "GetInputParameter /key=name"
- Set a field's value: name = Result

Add Action... Modify... Remove Move Up Move Down

☐ Stop processing rules when this rule finishes

OK Cancel

15. Save the form and redeploy the form to your SharePoint Library.

There is no way to get to the input parameters outside of the Loading event first occurring, and, in InfoPath, rules always run before code. Therefore, Open Rules will not have access to the input parameter data. This is why we designed our form to use a rule in **finishedLoading** to load the information, instead of using Open Rules.

TEST IT OUT



1. From Internet Explorer, launch a new Travel Request form and fill out the fields.
2. Click on the hyperlink to launch the Expense Report.
3. See the fields get populated.

The screenshot displays two Microsoft Office InfoPath forms side-by-side. The left form, 'Form2 - Microsoft Office InfoPath', is a 'Travel Request' form. It contains fields for 'Business Purpose' (filled with 'this is my purpose'), 'Traveler Information' (Name: 'John Doe'), 'Trips' (a table with one row for a trip from 'From' to 'To' on '4/8/2010'), and 'Preferences' (Trip Class, Car Class, Seat Location). A red box labeled 'Step 1' highlights the 'Business Purpose' and 'Traveler Information' fields. A red arrow points from this box to a red box labeled 'Step 2' in the 'Expense Report' form. The right form, 'Form3 - Microsoft Office InfoPath', is an 'EXPENSE REPORT' form. It contains fields for 'Report Date' (4/8/2010), 'Business Purpose' (filled with 'this is my purpose'), and 'Employee Information' (Name: 'john doe'). A red box labeled 'Step 3' highlights the 'Business Purpose' and 'Employee Information' fields. A red arrow points from the 'Step 2' box to the 'Step 3' box. A red box labeled 'Step 2' is also present in the 'Travel Request' form, containing a hyperlink: 'Launch a new Expense Report form for this trip.'.

You can extrapolate this simple example to use qRules to pass data from one form to another.

InfoPath changes all input parameters to lower case. While qRules handles for this by also modifying the case of the provided /key, it is simplest to just use lower case for the parameter key.