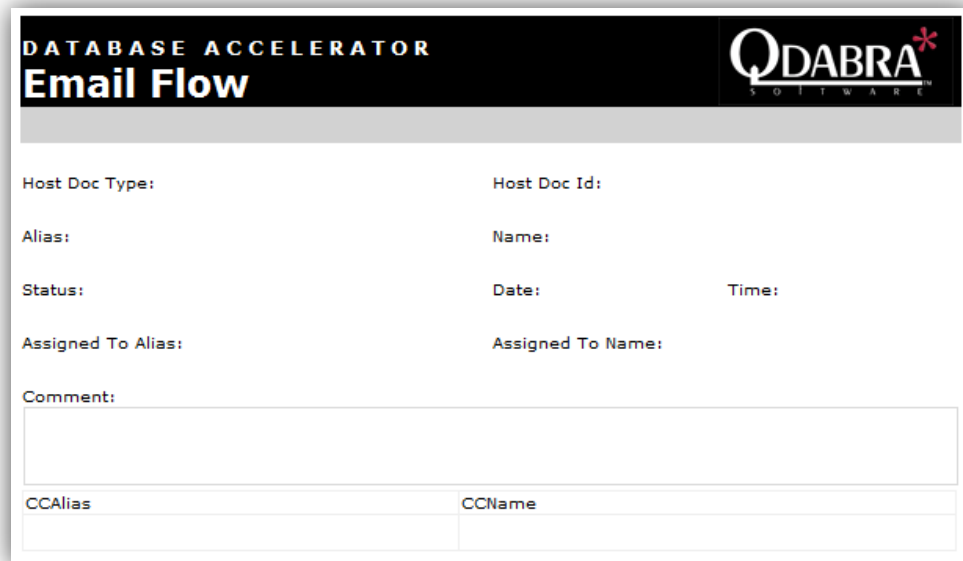


USING QDEMAILFLOW ASSIGNMENTS

WHAT IS QDEMAILFLOW?

QdEmailFlow is one of the System Document Types included in Qdabra's Database Accelerator (DBXL). It allows us to create assignments, which can then be used for email Notifications and/or Permissions. The form template for the System Document Type looks like this:



The screenshot shows a web form titled "DATABASE ACCELERATOR Email Flow" with the QDABRA SOFTWARE logo in the top right. The form contains the following fields:

Host Doc Type:	Host Doc Id:	
Alias:	Name:	
Status:	Date:	Time:
Assigned To Alias:	Assigned To Name:	
Comment:		
CCAlias	CCName	

As seen above, the System Document Type stores the following data:

- Host Doc Type is the Document Type where the assignment was made
- Host Doc ID is the DocID for the document where the assignment was made
- Alias and Name of the assigner
- Alias and Name of the assignee
- Status, Date and Time
- Comments
- Alias and Name of the CC assignee (for the purposes of email)

However, you will never have to edit this content directly. Assignments can be made by using the AssignDocument command in qRules.



THE ASSIGNDOCUMENT QRULES COMMAND

The AssignDocument command in qRules will allow you to create a QdEmailFlow assignment for a document that is stored in DBXL. The command uses the following arguments:

- **submit:** The name of the query (receive) data connection to DBXL
- **docid:** DocID of the target document
- **comment:** QdFlow allows comments to be included with the assignment (optional)
- **status:** Status that will be bestowed upon the target document (optional). If Status is not declared, it will remain blank.
- **alias:** User alias of the user being assigned to the target document
- **name:** Name of the user being assigned to the target document (optional)
- **cc:** Semi-colon separated aliases of users that will be CCd in any email notifications as a result of this assignment (optional)

Some additional Observations:

- This command does not validate the values provided (except for the **/submit** parameter). This means that you need to make sure that the values for the parameters are valid.
- All arguments are literal strings. If you need to specify xpath values for name, author and description, make sure to use a concat operation.

Let's prepare a form to use the command.



CREATE A FORM THAT USES ASSIGNDOCUMENT TO CREATE QDEMAILFLOW ASSIGNMENTS

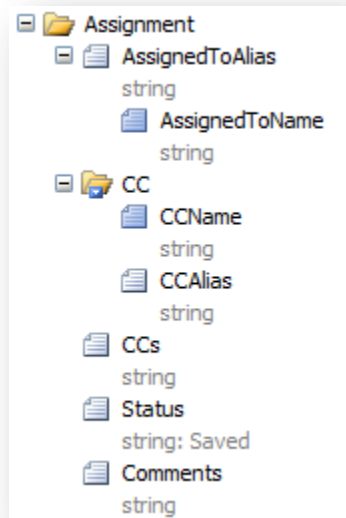
Pre-requisites: The following steps assume the following:

- You have downloaded and installed DBXL. You have access to create document types in DAT.
- You have downloaded and installed qRules v2.3 (or newer).
- You have the qRules User Guide, which is included in every qRules installation and can also be found by [clicking here](#).

Follow these steps to prepare your form template.

- A. Create a new InfoPath form and inject it with qRules. For details on this, please see the qRules User Guide.
- B. Some of the qRules commands, including AssignDocument, require a *query* (receive) data connection to the Submit Document method of the DBXLDocumentService Web service. Refer to the qRules User Guide Appendix to find step by step instructions on how to create this data connection.
- C. Create a button to submit to DBXL using the qRules command SubmitToDbxl. For detailed steps, please refer to [this document](#).
- D. Add a connection to GetMyInfo.
 1. Go to Data Connections and add a new data connection that receives data from a SOAP Web Service.
 2. When asked for the web service URL, use <http://<servername>/QdabraWebService/AdUserInfo.aspx>.
 3. Select GetMyInfo.
 4. You do not need to set this connection to execute when the form opens.
 5. Finish the data connection wizard.
- E. Add the necessary fields to the form.
 1. Add a group to your schema and call it Assignment. This group will hold the fields we'll need to perform the assignment.
 2. Design the schema by adding fields as shown by the screenshot below.





3. Add controls to your canvas to display these fields. That way your users can fill out the assignment information as they are filling out the form.
4. Optional: You can add validation using the Active Directory web service. For example, you could design the form such that the user only needs to enter the aliases (AssignedToAlias and CCAlias). Rules would then validate these aliases by using the Active Directory web service and return the names (AssignedToName, CCName). We include screenshots of the rules for the AssignedToAlias field in case you'd like to set this up. Similar rules could be used for the CCAlias field. However, it is not necessary to proceed. For more information on the Active Directory Web Service, please refer to [Task 2 in the Getting Started document](#) and to [this document](#).



Details for:
Get Assignee Name

Condition:
None - Rule runs when field changes

Rule type:
Action

Run these actions: * Add ▼

Set a field's value: username = .

Query using a data connection: GetUserInfo

Set a field's value: @AssignedToName = Value[Key = "cn"]

☐ Don't run remaining rules if the condition of this rule is met

Details for:
Invalid User Alias

Condition:
AssignedToAlias is not blank and
AssignedToName is blank

Rule type:
Action

Run these actions: * Add ▼

Show dialog box message: User not found. Please make sure you have entered the correct U...

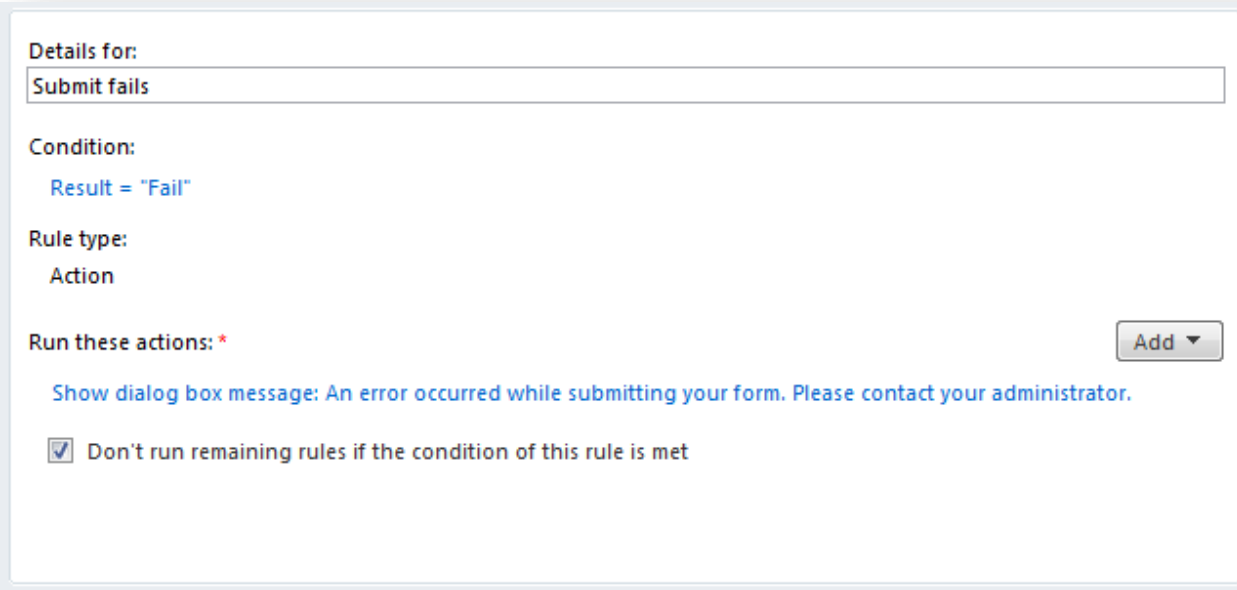
Set a field's value: , = ""

☐ Don't run remaining rules if the condition of this rule is met

- F. Add a rule to stop if submit failed.
1. On your submit button, add a new rule.



2. The condition on this new rule will be **Result = "Fail"**. If the command SubmitToDbxl, configured in step C, happens to fail, we'll catch the error and stop.
3. Add an action to show a dialog box message, with a message such as **"An error occurred while submitting your form. Please contact your administrator."**
4. Check the option to *"Don't run remaining rules if the condition of this rule is met"*.



Details for:
Submit fails

Condition:
Result = "Fail"

Rule type:
Action

Run these actions: * Add ▼

Show dialog box message: An error occurred while submitting your form. Please contact your administrator.

☒ Don't run remaining rules if the condition of this rule is met

- G. Add two rules to call AssignDocument if submit succeeds.

Why do we need two rules? Although CC assignment is optional, it cannot be blank. Therefore we need two rules: one when there are no CCs specified, and another in the case where CCs are used.

1. On the submit button, add a new rule and call it **Assign without CC**.
2. Add a condition to this rule: CCs is blank.
3. Add an action to set the value of the **Command** node in the **QdabraRules** 2DS to our command:
concat("AssignDocument /submit=DBXLSubmit /myinfo=GetMyInfo /docid=", docId, "
/comment=", Comments, " /status=", Status, " /alias=", AssignedToAlias). Note that:
 - **DBXLSubmit** is the name of the data connection to the DBXLDocumentService created in step B.
 - **GetMyInfo** is the name of the Active Directory data connection created in step D.
 - The **docId** node is located in the **DBXLSubmit** secondary data connection.
 - The **Comments**, **Status** and **AssignedToAlias** nodes all come from the main data source (and thus from the controls added to the form in step E).



Details for:

Condition:
 CCs is blank

Rule type:
 Action

Run these actions: * Add ▾

☐ Don't run remaining rules if the condition of this rule is met

4. Add another rule and call it **Assign with CC**.
5. Add a condition to this rule: CCs is not blank.
6. Add an action to set the value of the **Command** node in the **QdabraRules** 2DS to our command:
 concat("AssignDocument /submit=DBXLSubmit /myinfo=GetMyInfo /docid=", docId, "
 /comment=", Comments, " /status=", Status, " /alias=", AssignedToAlias, " /cc=", CCs). Note that:
 - **DBXLSubmit** is the name of the data connection to the DBXLDocumentService created in step B.
 - **GetMyInfo** is the name of the Active Directory data connection created in step D.
 - The **docId** node is located in the **DBXLSubmit** secondary data connection.
 - The **Comments**, **Status**, **AssignedToAlias**, and **CCs** nodes all come from the main data source (and thus from the controls added to the form in step E).





Details for:
Assign with CC

Condition:
CCs is not blank

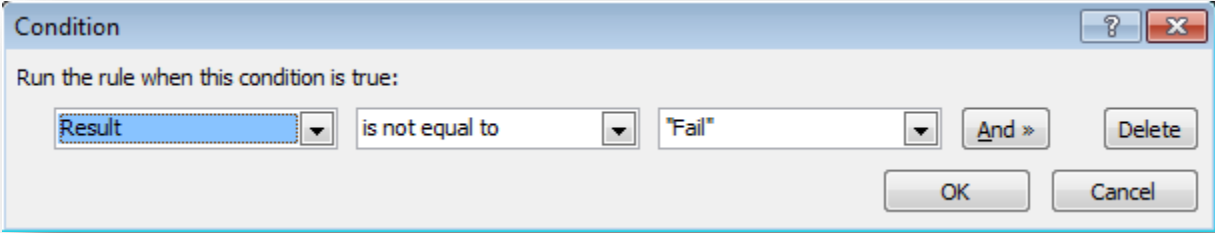
Rule type:
Action

Run these actions: * Add ▾

Set a field's value: Command = concat("AssignDocument /submit=DBXLSubm...

☐ Don't run remaining rules if the condition of this rule is met

- H. Add a rule to determine if the assignment was successful.
1. Add a new rule and call it **Assignment successful**.
 2. Add the condition **Result is not equal to "Fail"**.



Condition

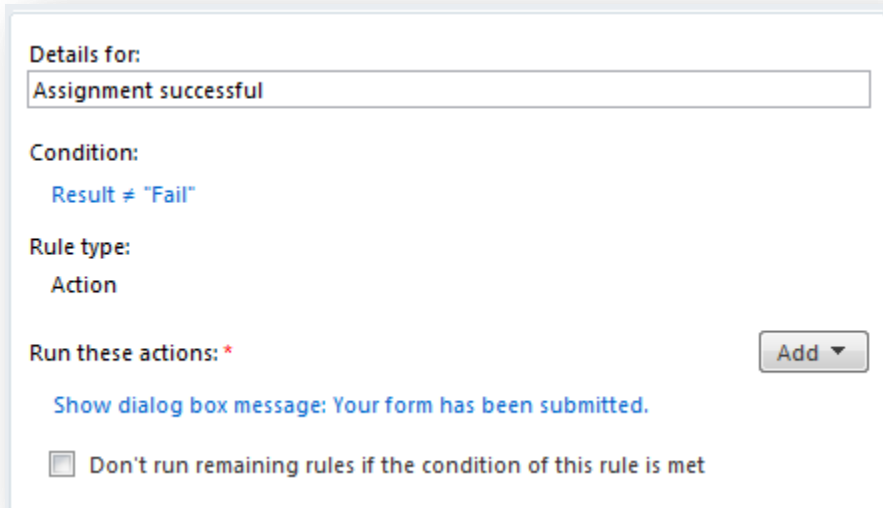
Run the rule when this condition is true:

Result ▾ is not equal to ▾ "Fail" ▾ And >> Delete

OK Cancel

3. Add an action to display a dialog message, such as "Your form has been submitted."





Details for:
Assignment successful

Condition:
Result $\neq$ "Fail"

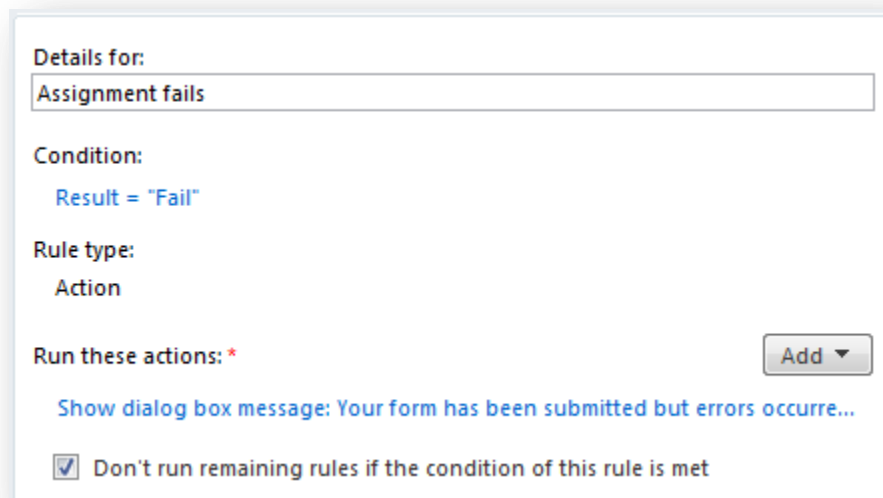
Rule type:
Action

Run these actions: * Add ▾

Show dialog box message: Your form has been submitted.

☐ Don't run remaining rules if the condition of this rule is met

- I. Add a rule to determine if the assignment failed.
 1. Add a new rule and call it **Assignment fails**.
 2. Add the condition **Result = "Fail"**.
 3. Add an action to display a dialog message such as "Your form has been submitted but errors occurred during assignment. Please contact an administrator."
 4. Check the option to "Don't run remaining rules if the condition of this rule is met".



Details for:
Assignment fails

Condition:
Result = "Fail"

Rule type:
Action

Run these actions: * Add ▾

Show dialog box message: Your form has been submitted but errors occurre...

☒ Don't run remaining rules if the condition of this rule is met

Save your form as QdFlowSample.xsn.



<http://www.qdabra.com>

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

HOW TO ENABLE QDEMAILFLOW PERMISSIONS FOR A DOCUMENT TYPE

A Document Type that uses QdEmailFlow assignments allows you to use permissions based on assignment.

This tutorial will detail how to enable QdEmailFlow Permissions for a Document Type and will explain the results of the enforced permissions model.

SET UP THE SCENARIO

1. Create a document type based on QdFlowSample.xsn.
 - a. In DAT, click on **New Configuration**.
 - b. For the InfoPath form control, attach the QdFlowSample.xsn file.
 - c. Enter a name for your configuration, such as **QdFlowSample**.
 - d. Click on the **Permissions** tab and check the **Enforce Permissions** checkbox.
 - e. Click **Save**, and then click **OK** in the confirmation dialog.
2. Modify web.config.
 - a. Navigate to `C:\inetpub\wwwroot\QdabraWebService` and open web.config in the text editor of your choice, such as Notepad.
 - b. Find the `<providers>` node and declare your Document Type in the `<providers>` node. The section will look like this:


```
<provider name="QdFlow"
type="Qdabra.Dbxl.QdFlow.PermissionProvider,
Qdabra.Dbxl.QdFlow" connectionString="Data
Source=<sqlinstance>;Initial Catalog=<database>;Integrated
Security=True">
  <doctypes>
    <doctype name="QdEmailFlow" />
    <doctype name="QdFlowSample" />
  </doctypes>
</provider>
```
 - c. Save and close web.config.
3. Enforce permissions on QdEmailFlow
 - a. In DAT, click to edit the QdEmailFlow System Document Type.
 - b. Click on the **Permissions** tab and check the **Enforce Permissions** checkbox.
 - c. Click **Save**, and then click **OK** in the confirmation dialog.

TEST THE PERMISSIONS MODEL

4. Add documents to the new DocType.
 - a. Return to DAT and click the  icon for the new Document Type.
 - b. Fill out the form including the assignment fields. Submit.
 - c. Submit another document but assign it to a second user.
 - d. Note the DocID of each of these two new documents. (You'll find the DocIDs in the Documents tab in DAT.)



If you are able to log in as the second user, you'll be able to verify that the user has permissions to the assigned (second) document, but the user should not have access to the first document (assigned to you).

If you plan on creating a permissions model that requires both QdEmailFlow, as described above, and Document Type permissions, you must give all users (NT AUTHORITY\AUTHENTICATED USERS) *Read* permission on the QdEmailFlow Document Type.

For more information on permissions in DBXL, please read [this document](#).



THREE WAYS TO CONFIGURE PROVIDER PERMISSIONS IN WEB.CONFIG

There are three ways of entering providers in web.config.

- **Default:** Simply add the document type name into web.config providers node. Because Form2 is not listed for the provider, this document type configuration will not be covered by QdEmailFlow permissions:

```
<provider name="QdFlow"
type="Qdabra.Dbxl.QdFlow.PermissionProvider, Qdabra.Dbxl.QdFlow"
connectionString="Data Source=<sqlinstance>;Initial
Catalog=<database>;Integrated Security=True">
  <doctypes>
    <doctype name="QdEmailFlow" />
    <doctype name="Form1" />
  </doctypes>
</provider>
```

- **Exclusion:** We will **exclude** Form2 from the provider, and include all other document types.

```
<provider name="QdFlow"
type="Qdabra.Dbxl.QdFlow.PermissionProvider, Qdabra.Dbxl.QdFlow"
connectionString="Data Source=autonomy2\development;Initial
Catalog=QdabraSamples-DBXL-TEST3;Integrated Security=True">
  <doctypes exclusionRule="true">
    <doctype name="Form2" />
  </doctypes>
</provider>
```

- **Inclusion:** We will **include** QdEmailFlow and Form1 in the provider, and exclude all other document types.

```
<provider name="QdFlow"
type="Qdabra.Dbxl.QdFlow.PermissionProvider, Qdabra.Dbxl.QdFlow"
connectionString="Data Source=autonomy2\development;Initial
Catalog=QdabraSamples-DBXL-TEST3;Integrated Security=True">
  <doctypes exclusionRule="false">
    <doctype name="QdEmailFlow" />
    <doctype name="Form1" />
  </doctypes>
</provider>
```

Some notes:

- The Document Type names within the provider node are not case-sensitive.
- Exclusion and inclusion providers are mutually exclusive.
- Every <provider> node must have a valid connection string in order to work

This flexibility is convenient if you have a long list of document types to include or exclude from these QdEmailFlow permissions.



SUPPORT

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

Licensed customers can contact us via Support@Qdabra.com.

You can also use the [InfoPathDev.com](http://www.infopathdev.com) [Qdabra Product support forums](http://www.infopathdev.com) to request help from the community.

CONTACT INFORMATION

Qdabra Software
218 Main Street, Suite 731
Kirkland, WA 98033

Phone:	877.544.2389
Email:	Support@Qdabra.com
Website:	http://www.qdabra.com
Community:	http://www.infopathdev.com



<http://www.qdabra.com>

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