

Integrating Qdabra™ DBXL with Nintex Workflow

Contents

Scenario.....	1
Expense Report Form Library.....	2
Workflow Definition.....	4
InfoPath Form Query XML Configuration	6
Workflow Variables.....	7
Call Web Service Configuration.....	8
Error handling settings.....	10
Success Query XML Configuration	10
Executing the workflow	11
Obtaining the DocID of the Submitted Form	18
Web Request Action	19
Summary	21

Scenario

Sales people in the field submit expense reports using an InfoPath form on a SharePoint extranet site. Once saved to the SharePoint library, a Nintex Workflow is used to manage the approval process. The final step of the workflow is to submit the expense report to Qdabra™ DBXL, which then shreds the content for use by the Accounting department's systems and reporting services.

Expense Report Form Library

The Expense Report is a simple form which collects the expense information. The user enters his information and saves the form to the SharePoint library.

Edit

 Submit
  Save
  Save As
  Close
  Paste
  Copy
  Cut
  Print Preview

Commit
 Clipboard
 Views

EXPENSE REPORT

Report Date:
Expense Code:
Start Date:
End Date:

Business Purpose:

Employee Information

Name:
Title:

Department:
ID Number:

E-mail Address:

Manager Information

Name:
E-mail Address:

Itemized Expenses

Date	Description	Category	Cost
2/28/2011	Lunch	Meals	35.88
2/28/2011	Parking	Transportation	5.00
Subtotal			40.88
Less cash advance			0.00
Total expenses			40.88

☒ Add Expense

Item Details

Figure 12: InfoPath form opened in browser mode from SharePoint site

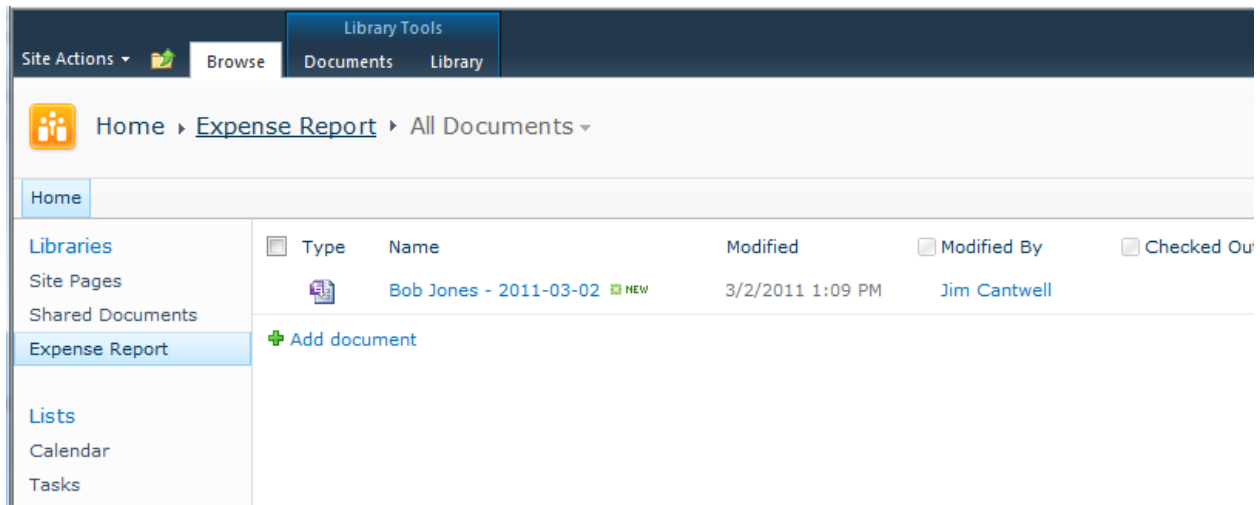


Figure 1: Expense Report form library

Once saved to the library, the manager can run the Approve Expense Report workflow.

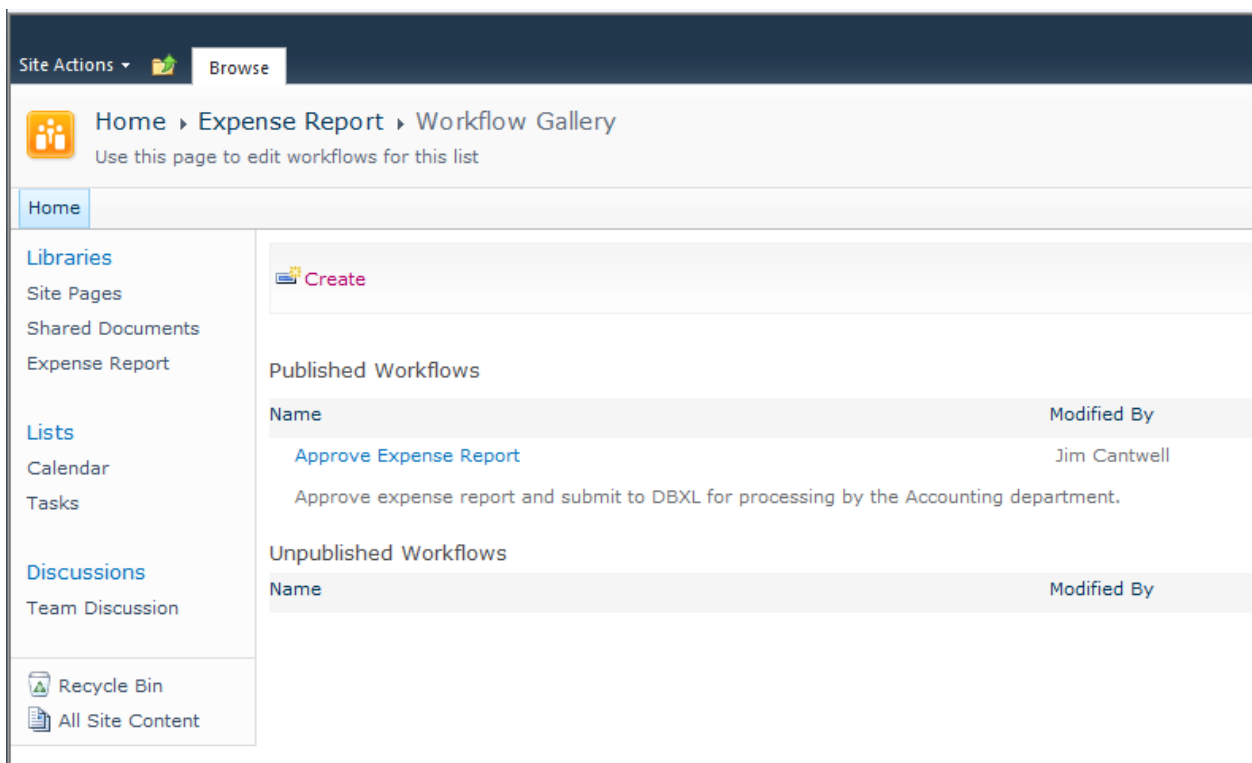


Figure 2: Approve Expense Report workflow

Workflow Definition

The first step of the workflow is to copy the InfoPath form into a workflow variable. This is done using a Query XML action in Nintex Workflow. Then the Qdabra™ DBXL webservice is called, passing it the content of the InfoPath form. The SubmitDocument result is then checked for success or failure. If the document submitted successfully, it deletes it from the library. If it failed to submit, it sends an email notification about the error.

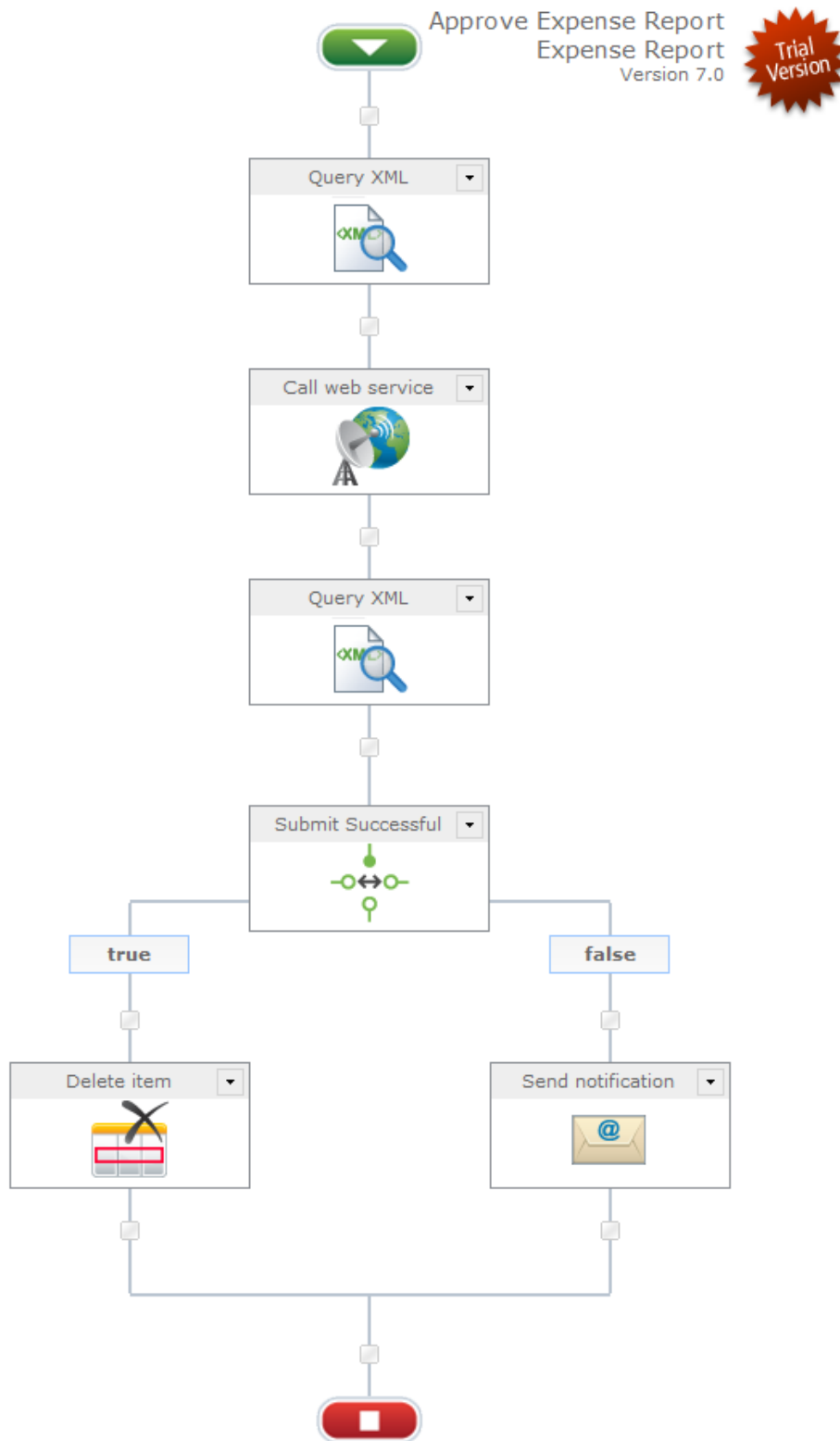
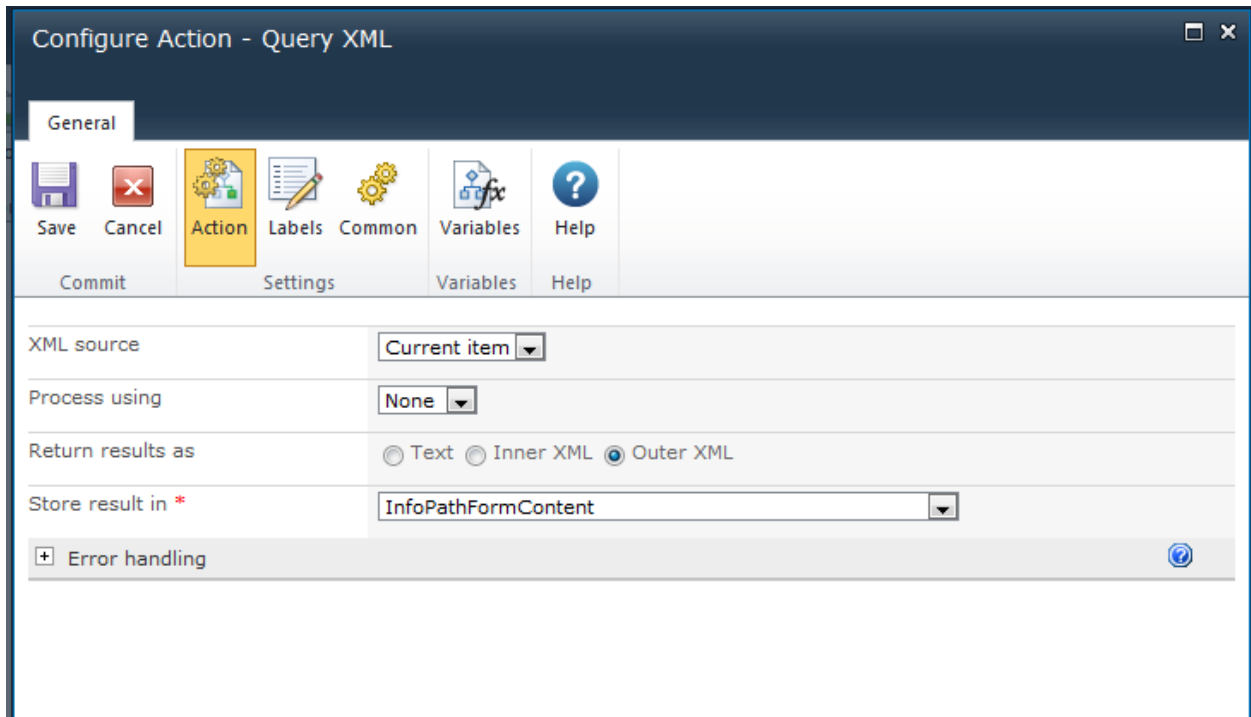


Figure 3: Workflow diagram

InfoPath Form Query XML Configuration

The first step in the workflow is to store the content of the current item into a workflow variable, storing it as the Outer XML.



Configure Action - Query XML

General

Save Cancel Action Labels Common Variables Help

Commit Settings Variables Help

XML source: Current item

Process using: None

Return results as: ☐ Text ☐ Inner XML ☒ Outer XML

Store result in *: InfoPathFormContent

+ Error handling

Figure 4: Query XML Configuration

Workflow Variables

I have 5 variables defined for storing results during the workflow process.

InfoPathFormContent – stores the xml content of the InfoPath form

QdabraResults – stores the xml returned from SubmitDocument

GotError – Stores the Nintex Workflow error indicator

ErrorText – Stores error text if an error occurs

Success – variable to store the Success flag returned as part of the QdabraResults xml

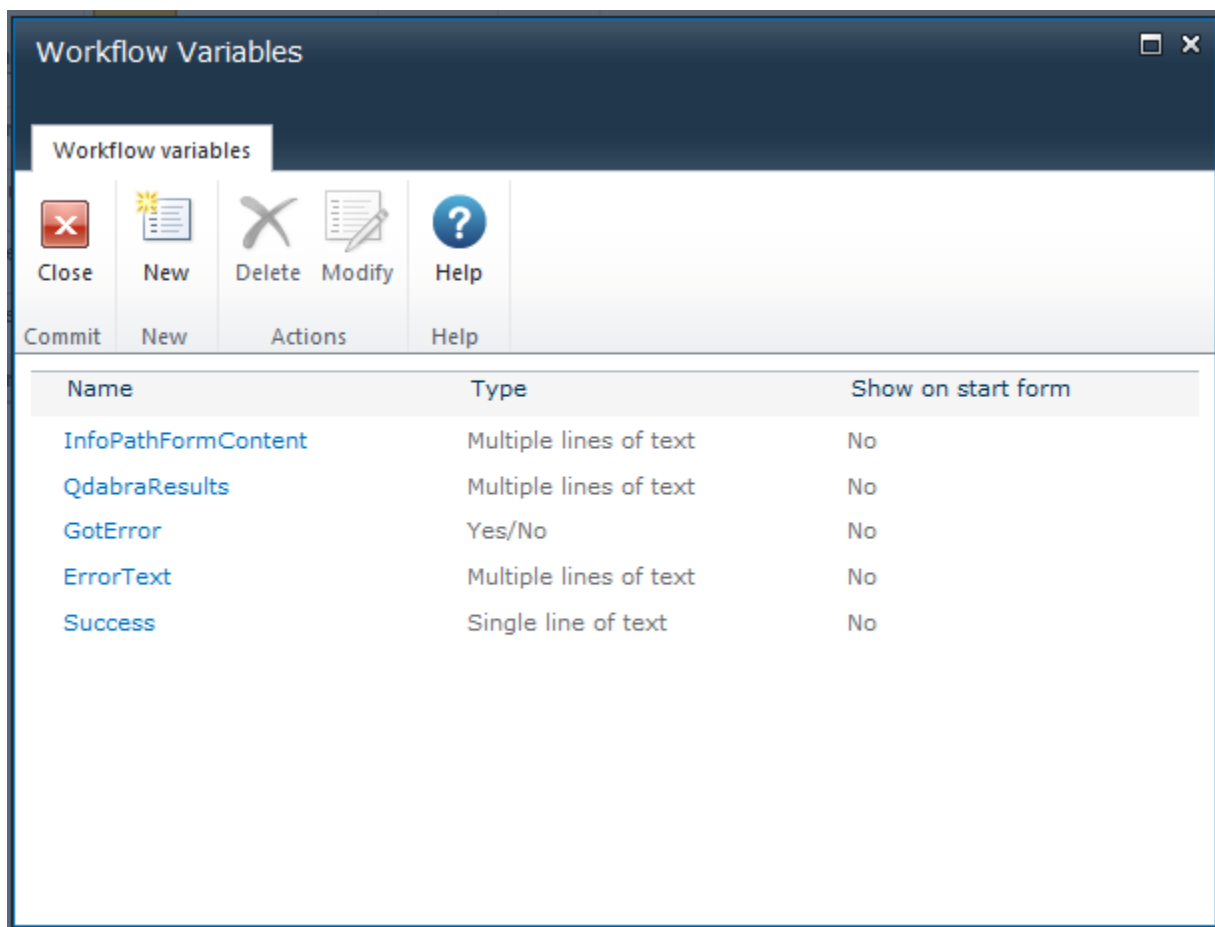


Figure 5: Workflow variables

Call Web Service Configuration

Setting up the web service call to SubmitDocument requires just entering the url to the DbxlDocumentService.asmx page on your server hosting the DBXL web service. To avoid cross-site security issues, DBXL should be installed on the same web application. In the web service message, parameters are supplied using a hard-coded document type name, the xml form content encoded as a string. The optional DBXL name, author and description properties are set to be populated using the SharePoint list properties, Name, Modified By and Title respectively.

For credentials, you can specify your credentials directly, or specify credentials from the secure store service. The account will need to have Add permissions for the DBXL document type specified.

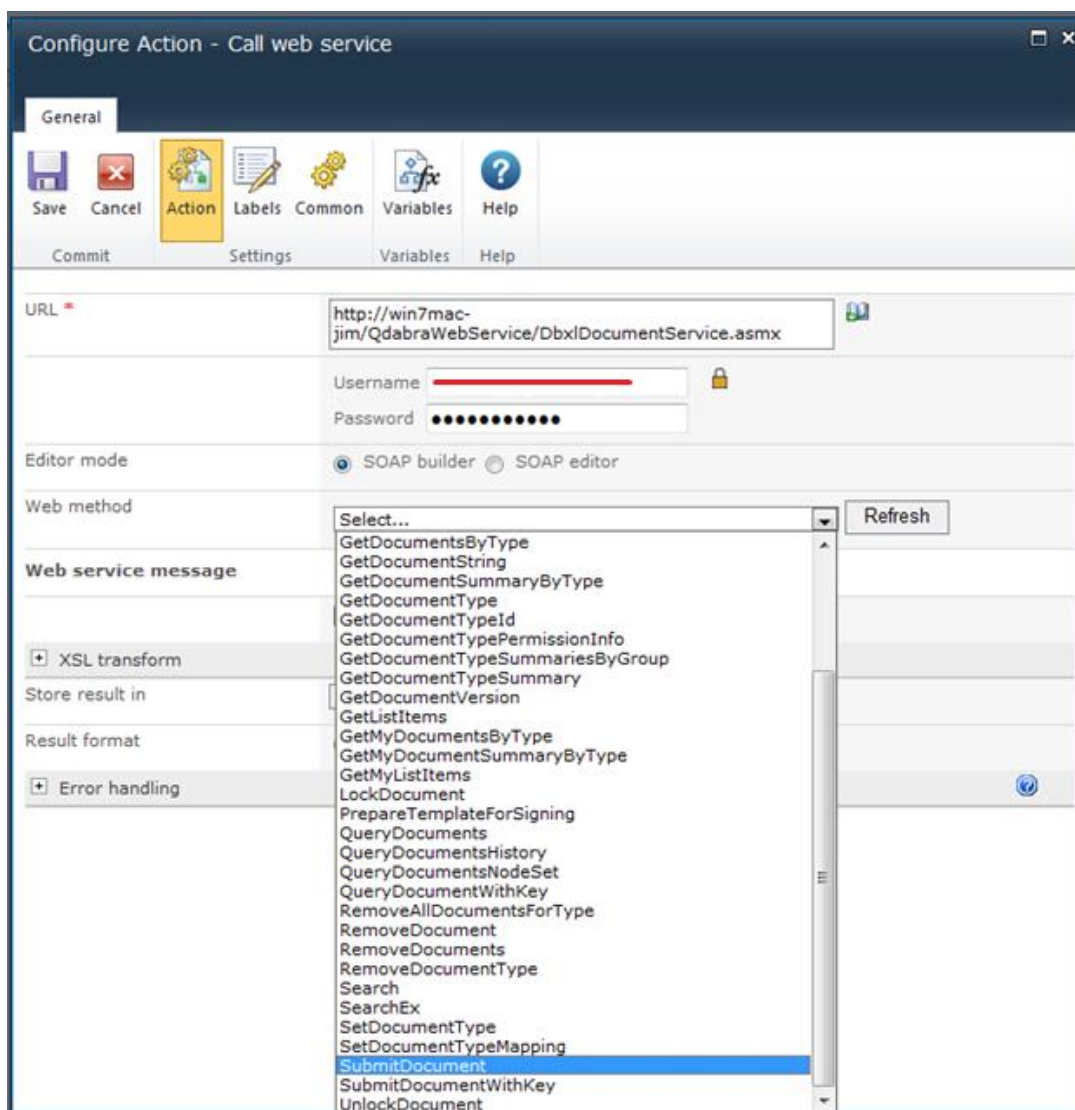
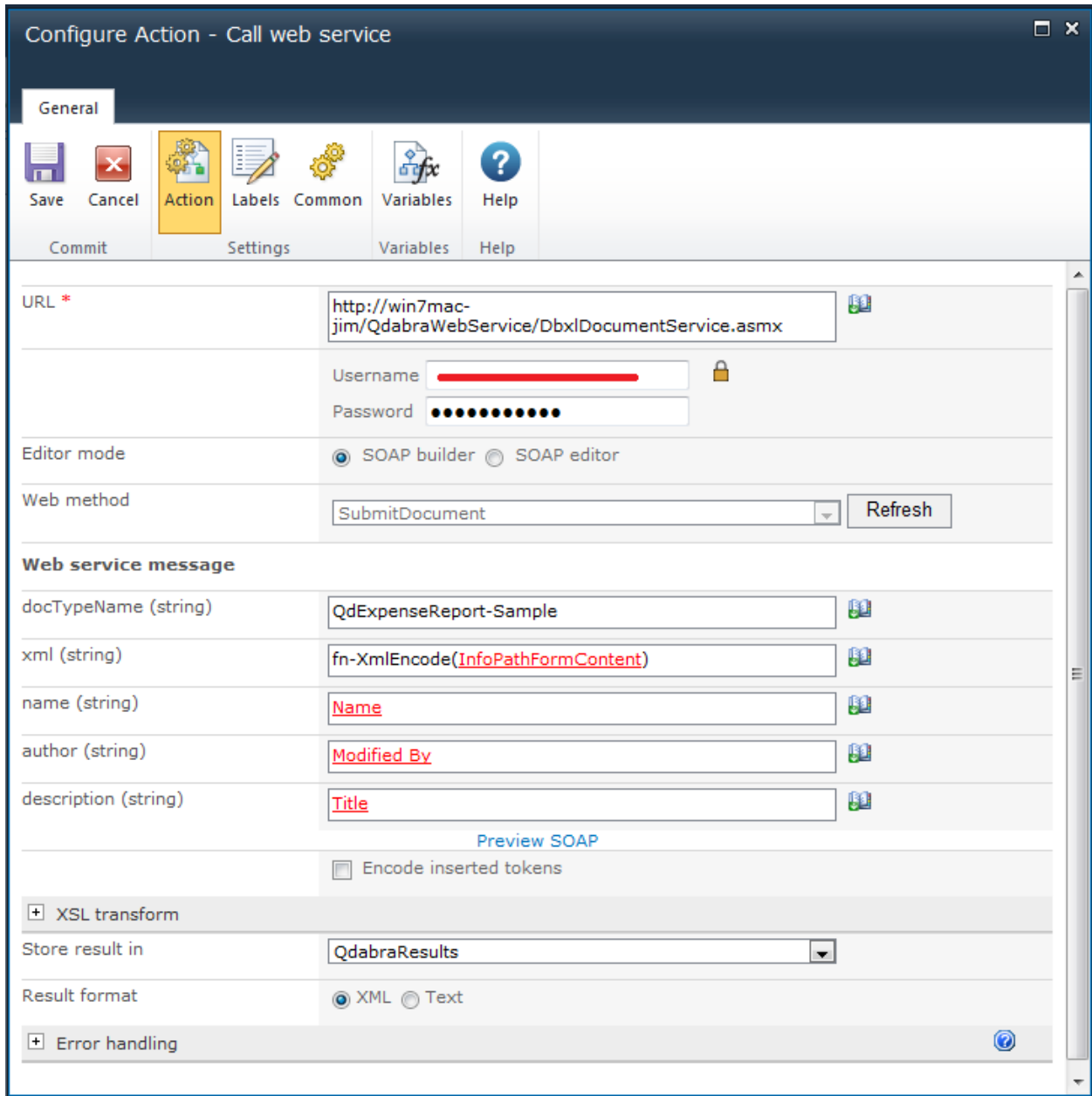


Figure 6: Selecting the SubmitDocument web method

The InfoPathFormContent is wrapped in a fn-XmlEncode call to correctly pass the xml content as a string.



Configure Action - Call web service

General

Save Cancel **Action** Labels Common Variables Help

Commit Settings Variables Help

URL *

Username

Password

Editor mode ☒ SOAP builder ☐ SOAP editor

Web method Refresh

Web service message

docTypeName (string)

xml (string)

name (string)

author (string)

description (string)

[Preview SOAP](#)

☐ Encode inserted tokens

XSL transform

Store result in

Result format ☒ XML ☐ Text

Error handling

Figure 7: Call web service settings

Error handling settings

Each of the actions has the error handling settings as configured in Figure 7.

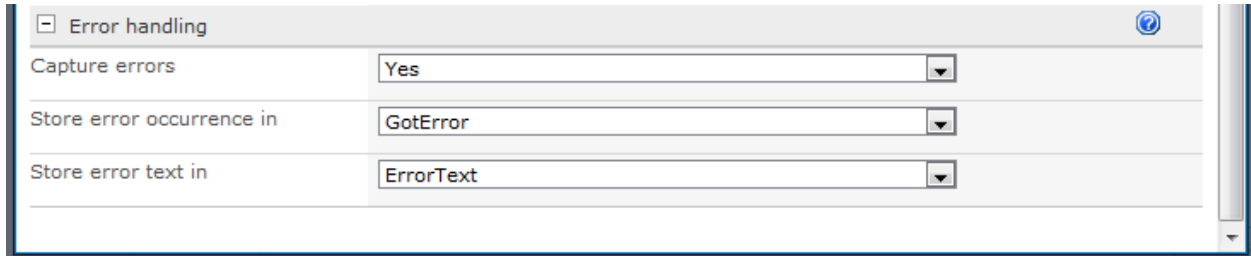


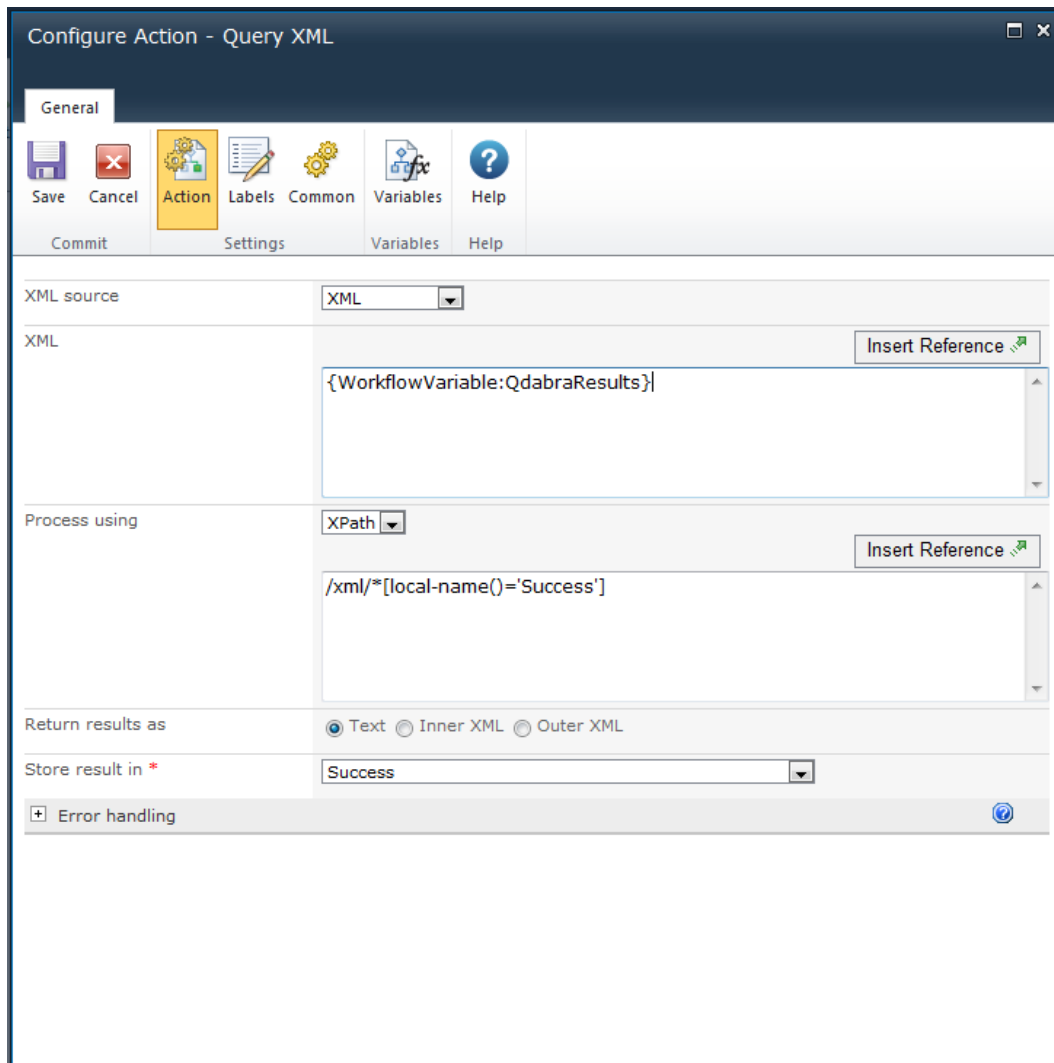
Figure 8: Error handling settings

Success Query XML Configuration

The result returned from SubmitDocument stored in the QdabraResult variable is in the form:

```
<xml>
  <Success xmlns="http://qdabra.com/webservices/">true</Success>
  <Errors xmlns="http://qdabra.com/webservices/" />
  <ExecutionTime
xmlns="http://qdabra.com/webservices/">00:00:06.2551211</ExecutionTime>
</xml>
```

We use an xpath query to extract the field and store in the Success variable. Since the result has a namespace, we use the local-name() function to match on the unqualified node name.



Configure Action - Query XML

General

Save Cancel Action Labels Common Variables Help

Commit Settings Variables Help

XML source XML

XML {WorkflowVariable:QdabraResults} Insert Reference

Process using XPath Insert Reference

/xml/*[local-name()='Success']

Return results as ☒ Text ☐ Inner XML ☐ Outer XML

Store result in * Success

Error handling

Figure 9: Query XML Configuration settings

Executing the workflow

The DBXL configuration below shows the doctype with no documents submitted. The next step is to execute the workflow for the selected document.

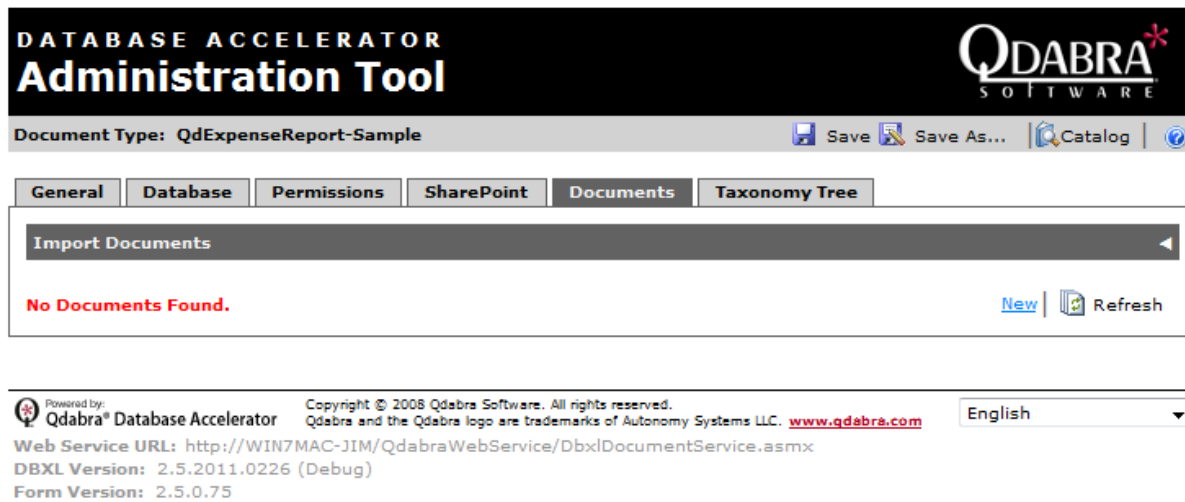


Figure 10: DBXL Admin tool documents

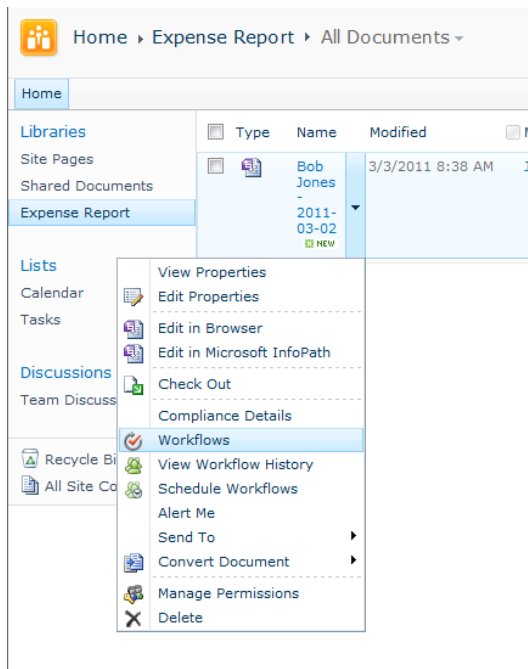


Figure 11: Select Workflows for document.

When the form is submitted to DBXL, the template href is automatically updated to reflect the template stored in DBXL. This allows the form to be opened from DBXL without needing access to the InfoPath template on the extranet server. Figure 9 shows the form opened in the browser on the SharePoint site.

Start a New Workflow



Approve Expense Report

Approve expense report and submit to DBXL for processing by the Accounting department.

Workflows

Select a workflow for more details on the current status or history. [Show my workflows only.](#)

Name

Started

Running Workflows

There are no currently running workflows on this item.

Completed Workflows

There are no completed workflows on this item.

Figure 13: Start a New Workflow page

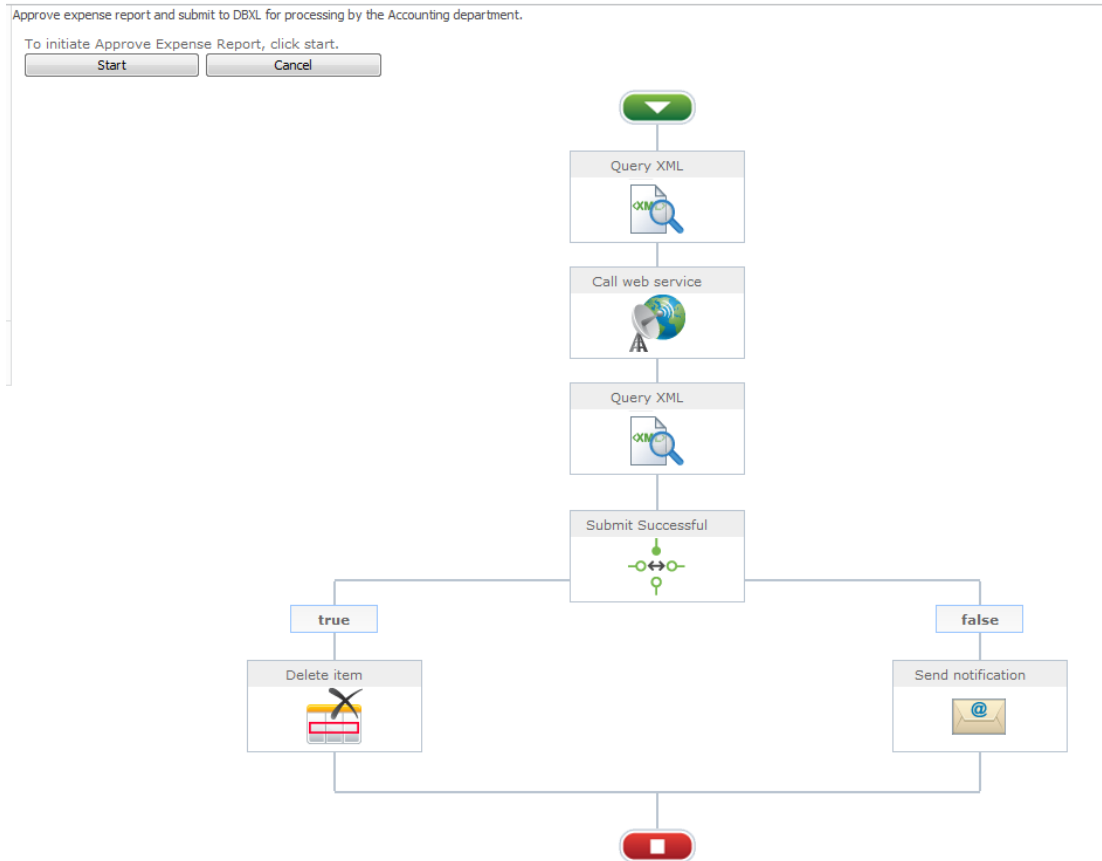


Figure 14: Workflow

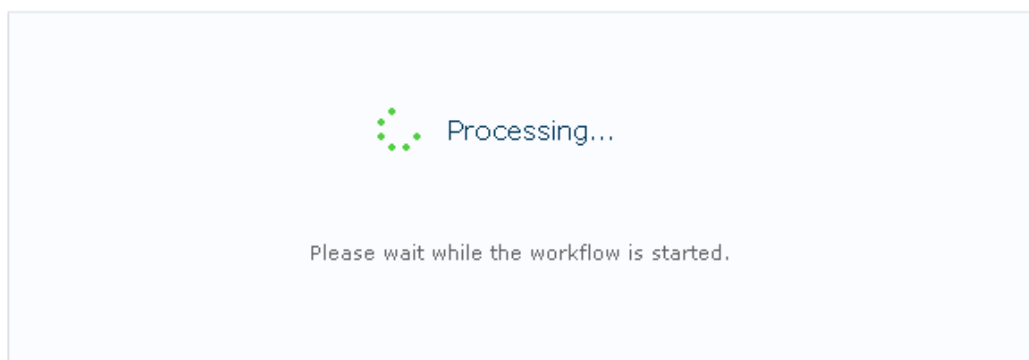


Figure 15: Workflow processing

After the workflow is completed, the document has been removed from the list.

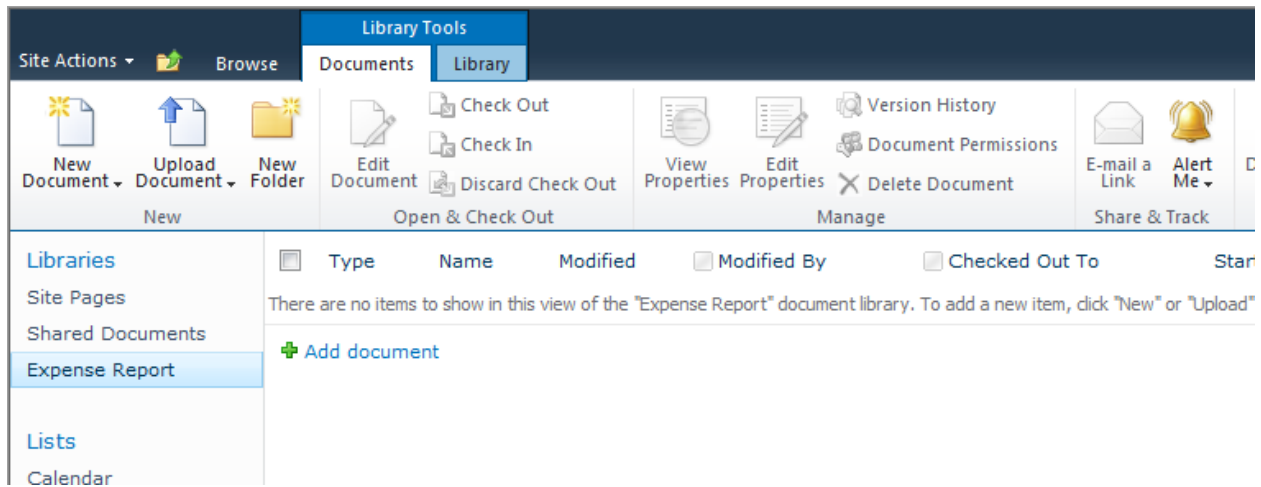


Figure 16 – Document removed after SubmitDocument executes successfully

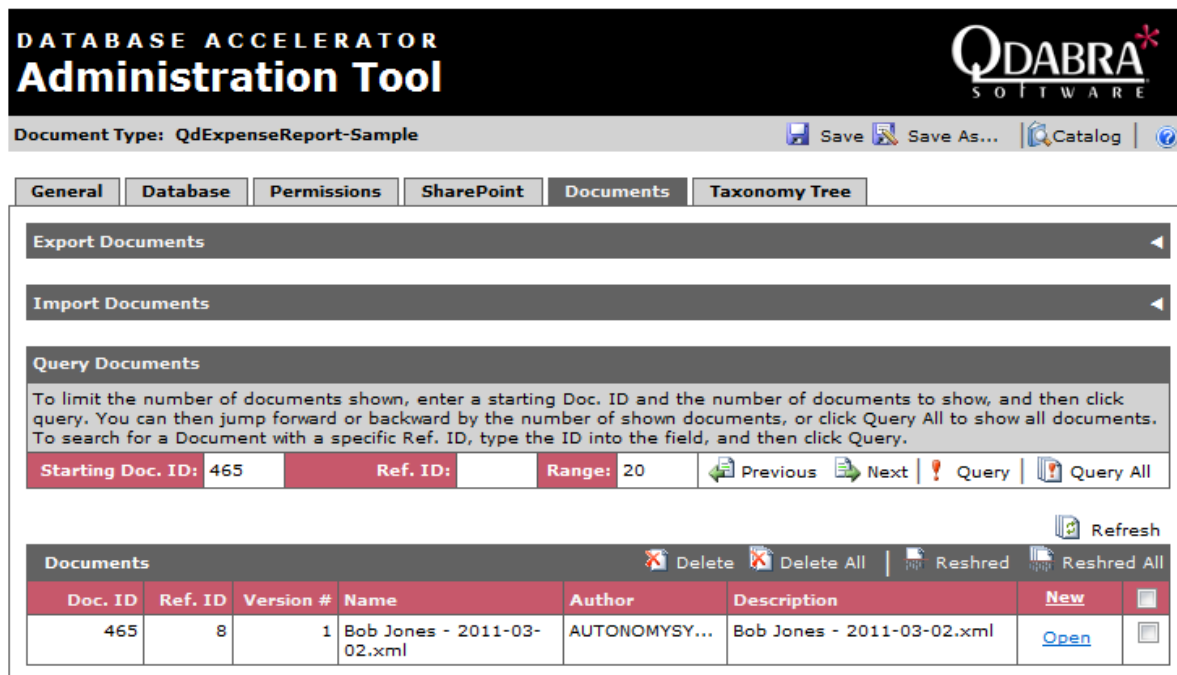


Figure 17: Document now appears in the DBXL admin tool UI.

After submitting to DBXL, the form is accessible through DBXL and can be opened in the InfoPath client.

Bob Jones - 2011-03-02.xml.xml - Microsoft InfoPath

EXPENSE REPORT

Report Date: 3/2/2011 Expense Code: Start Date: 2/28/2011 End Date: 2/28/2011

Business Purpose: Lunch with client

Employee Information

Name: Bob Jones Title: Consultant

Department: ID Number:

E-mail Address: bob@mycompany.com

Manager Information

Name: David Smith E-mail Address: david@mycompany.com

Itemized Expenses

Date	Description	Category	Cost
2/28/2011	Lunch	Meals	35.88
2/28/2011	Parking	Transportation	5.00
Subtotal			40.88
Less cash advance			0.00
Total expenses			40.88

☒ Add Expense

Item Details

Submit

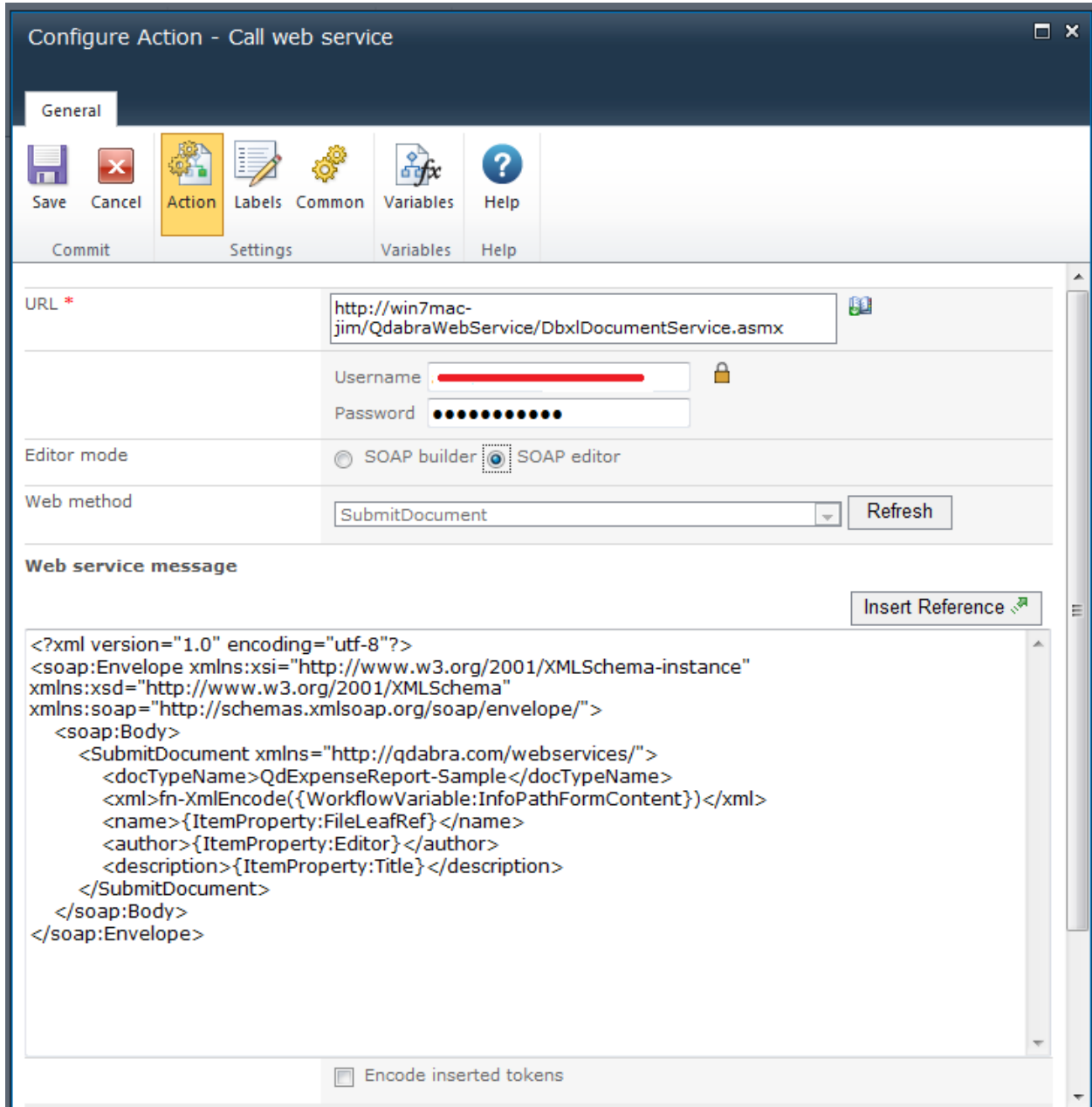
Form template's location: http://win7mac-jim

Figure 12 –Expense report opened from DBXL in the InfoPath client.

Obtaining the DocID of the Submitted Form

The Call Web Service action does not support returning the out parameters docId and refId. The StatusInfo structure is what is stored in the QdabraResults variable. If you need the docId for tracking or use once submitted, you will need to use the Nintex Workflow Web Request action. This will return the full soap message which can be queried using a Query XML action to obtain the docId.

The Web Request action takes a soap action. The easiest way to get this is to obtain the soap message from the call web service action used previously.



Configure Action - Call web service

General

Save Cancel Action Labels Common Variables Help

Commit Settings Variables Help

URL *

Username

Password

Editor mode ☐ SOAP builder ☒ SOAP editor

Web method Refresh

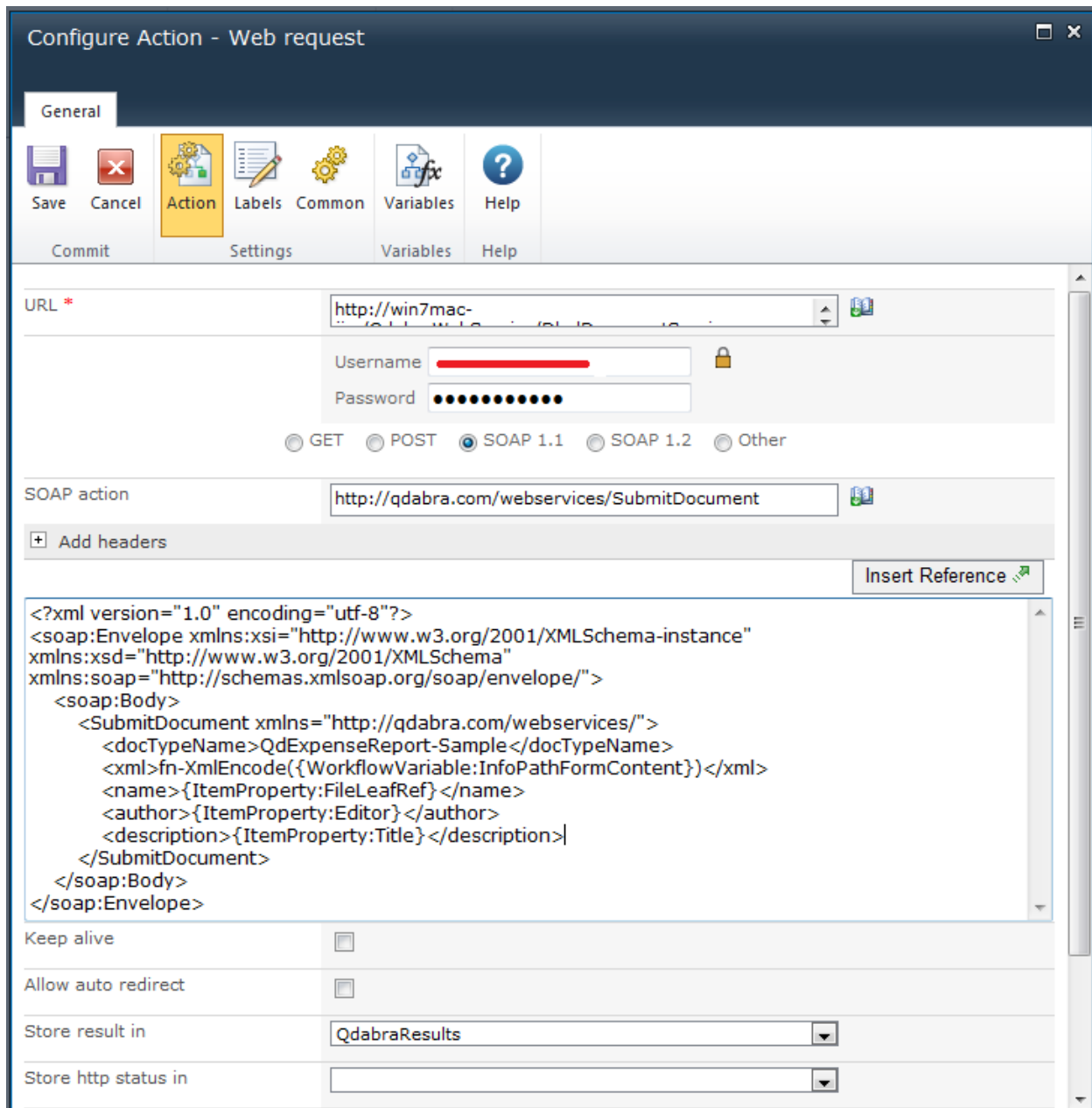
Web service message

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SubmitDocument xmlns="http://qdabra.com/webservices/">
      <docTypeName>QdExpenseReport-Sample</docTypeName>
      <xml>fn-XmlEncode({WorkflowVariable:InfoPathFormContent})</xml>
      <name>{ItemProperty:FileLeafRef}</name>
      <author>{ItemProperty:Editor}</author>
      <description>{ItemProperty:Title}</description>
    </SubmitDocument>
  </soap:Body>
</soap:Envelope>
```

☐ Encode inserted tokens

Web Request Action

Configure the Web Request action using the soap message obtained. The SOAP action will be <http://qdabra.com/webservices/SubmitDocument>.



Configure Action - Web request

General

Save Cancel Action Labels Common Variables Help

Commit Settings Variables Help

URL *

Username

Password

☐ GET ☐ POST ☒ SOAP 1.1 ☐ SOAP 1.2 ☐ Other

SOAP action

+ Add headers

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SubmitDocument xmlns="http://qdabra.com/webservices/">
      <docTypeName>QdExpenseReport-Sample</docTypeName>
      <xml>fn-XmlEncode({WorkflowVariable:InfoPathFormContent})</xml>
      <name>{ItemProperty:FileLeafRef}</name>
      <author>{ItemProperty:Editor}</author>
      <description>{ItemProperty:Title}</description>
    </SubmitDocument>
  </soap:Body>
</soap:Envelope>
```

Keep alive ☐

Allow auto redirect ☐

Store result in

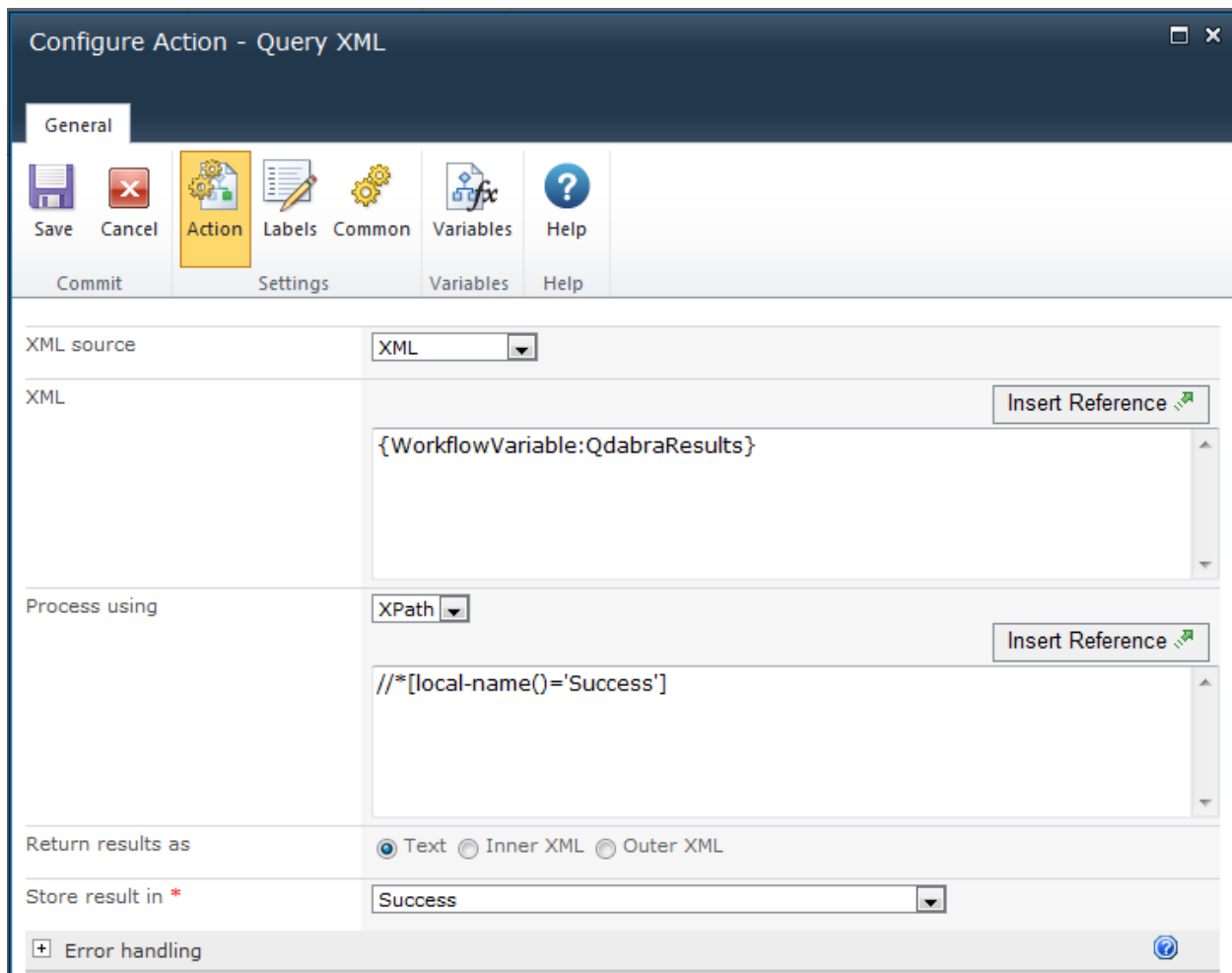
Store http status in

The stored result from the Web request action will be the entire return SOAP message. Below is an example of the returned structure.

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <SubmitDocumentResponse xmlns="http://qdabra.com/webservices/">
      <SubmitDocumentResult>
        <Success>true</Success>
        <Errors />
        <ExecutionTime>00:00:00.7500750</ExecutionTime>
      </SubmitDocumentResult>
      <docId>467</docId>
      <refId>10</refId>
    </SubmitDocumentResponse>
  </soap:Body>
</soap:Envelope>
```

To extract the docId and Success node values, the Query XML settings will need to be modified slightly to accommodate the change. For the Success indicator, use `//*[local-name()='Success']` and for the docId, use `//*[local-name()='docId']`. Create a new workflow variable of type integer to store the returned docId.



Configure Action - Query XML

General

Save Cancel Action Labels Common Variables Help

Commit Settings Variables Help

XML source XML

XML {WorkflowVariable:QdabraResults} Insert Reference

Process using XPath Insert Reference

Return results as ☒ Text ☐ Inner XML ☐ Outer XML

Store result in * Success

Error handling

Configure Action - Query XML

General

Save

Cancel

Action

Labels

Common

Variables

Help

Commit

Settings

Variables

Help

XML source

XML

XML

Insert Reference

{WorkflowVariable:QdabraResults}

Process using

XPath

Insert Reference

//*[local-name()='docId']

Return results as

☒ Text
☐ Inner XML
☐ Outer XML

Store result in *

DocID

+
Error handling

Summary

Using steps similar to how the SubmitDocument was performed, you can use other Qdabra™ DBXL apis to fully integrate your Nintex Workflow with Qdabra™ DBXL.