QDABRA DATABASE ACCELERATOR V2.6

Web Service Developer's Reference

CONTENTS

1 OVERVIEW

This document is meant as an SDK reference for developers that will be integrating DBXL Web Services with their InfoPath forms or any other application. The document describes all of the methods in each of the web services.

2 DEVELOPMENT TIPS

2.1 Web Services with Ex Suffix

There are two versions of some of the web services, one of which has an “Ex” suffix. These “Ex” APIs differ from their “non-Ex” counterparts only in the way errors are handled. Methods defined in the “Ex” pages return a SoapException when an error occurs. This allows InfoPath’s internal mechanism to detect the error and display an error message. The “non-Ex” methods return a StatusInfo object with a *Status* node, of value true or false, and error messages. Calling the non-Ex methods requires form logic to check the Success indicator and handle any errors.

2.2 DbxlSampleDocumentService

This web service provides identical functionality to the DbxlDocumentService web service, except that everything that it returns is derived from the sampledata.xml file present for the form template of the Document Type queried. This is necessary for InfoPath to derive a schema for the form when there are no documents in the Document Type. Also, when there are documents with varying schemas in a particular Document Type (which is considered bad practice), InfoPath cannot derive a schema for the Content node, regardless.

References to this web service are automatically redirected to DbxlDocumentService whenever a form is published in DAT. This way, you can develop a form with the Sample Data, but when it is deployed it will use the actual data.

2.3 QdabraDBXL Processing Instruction

When resubmitting a Document that has already been assigned a Document ID, the QdabraDBXL Processing Instruction must be present in the XML. DBXL uses this PI to determine whether the Document being submitted is new or existing. If it finds the PI with the attributes specified correctly, then it will move the old version of that Document to the archives, and then insert the new Document.

Use the following example to construct the QdbraDBXL PI. The docid and doctype values should come from those defined in the DBXL Admin Tool.

```
<?QdabraDBXL docid="_int_" doctype="_doc_type_name_"
name="_meta_data_" author="_meta_data_" description="_meta_data_"
?>
```

The codeless means to handle the insertion of the QdabraDBXL PI is to open the corresponding Document via Document Linking and Loading. When opening a Document from the web service, the QdabraDBXL PI is automatically injected.

There will be certain situations, however, that will require that you use code to inject the QdabraDBXL PI. There are two main scenarios where you will have to programmatically inject the QdabraDBXL PI. The first is when you want to allow users to intermittently save a form as they are filling it out. The second is when developing a catalog-style form, like the QdCatalogBase form. These two situations are covered later, in the Code Samples section.

3 CODE SAMPLES

It is recommended that you copy-paste the examples below into Visual Studio, as they will be much easier to read in an actual coding environment. These samples were created using Visual Studio 2005 and InfoPath 2003.

3.1 Using a Secondary Data Connection setup to Receive Data

The most common way to interact with the DBXL web services is to receive data from them. The following code sample shows the most common way to do this with code.

```
// Retrieve a handle to the secondary data connection. The string passed to
GetDOM()

// is the name as for the data connection as defined in the InfoPath
Designer.

IXMLDOMDocument3 domGetDocumentsByType =
(IXMLDOMDocument3)thisXDocument.GetDOM("GetDocumentsByType");

// Setup the namespaces for the secondary data connection. For more
information, see

// this HowTo:
http://www.infopathdev.com/blogs/matt/archive/2006/02/02/Setup-Namespaces-
for-a-Secondary-Data-Source.aspx

domGetDocumentsByType.setProperty(

    "SelectionNamespaces",

    "
xmlns:my=\"http://schemas.microsoft.com/office/infopath/2003/myXSD/2007-02-
01T19-18-48\" "

    + "
xmlns:dfs=\"http://schemas.microsoft.com/office/infopath/2003/dataFormSolut
ion\" "

    + " xmlns:ns4=\"http://qdabra.com/webservices/\" ");

// Change the value of a parameter

domGetDocumentsByType.selectSingleNode("/dfs:myFields/dfs:queryFields/ns4:G
etDocumentsByType/ns4:docTypeName").text = "PT_Timecard";

// Query the web service, which will use the new parameter value as defined
above

WebServiceAdapter2 wsaGetDocumentsByType =
(WebServiceAdapter2)thisXDocument.DataAdapters["GetDocumentsByType"];

wsaGetDocumentsByType.Query();

// Make sure the method invocation was successful

string success =
domGetDocumentsByType.selectSingleNode("/dfs:myFields/dfs:dataFields/ns4:Ge
tDocumentsByTypeResponse/ns4:GetDocumentsByTypeResult/ns4:Success").text;
```

```
if (success != "true")

    throw new Exception("Call to GetDocumentsByType failed.");

// Get values from the returned set

string docId =
domGetDocumentsByType.selectSingleNode("/dfs:myFields/dfs:dataFields/ns4:Ge
tDocumentsByTypeResponse/ns4:docInfos/ns4:DocumentInfo/ns4:DocID").text;
```

3.2 Calling SubmitDocument setup as a Receive Data Connection

There are two ways to submit documents into DBXL. The way that is documented in the Getting Started and other documents is to add a secondary data connection that is setup to Submit Data, but there will be situations (such as creating a catalog-type form) where you will want to programmatically work with the SubmitDocument web method. If you setup this web method as a "submit data" connection then it is very difficult to work with in code. The below code sample demonstrates how to use a SubmitDocument data connection as a "receive data" connection. The XML that is assigned to the xml parameter is trivial in this example, but when developing forms it will contain the entire XML of the form to be submitted. If it is a new document, it will not have the QdabraDBXL Processing Instructions, but if it is an update of a form that already has a Document ID, then the QdabraDBXL PI should be added to the top of the submitted XML.

Note that the current recommended approach to submitting documents to DBXL is to use the qRules SubmitToDbxl command, which takes care of checking the submission result and adding the DBXL PI to the main data source.

```
// Retrieve a handle to the secondary data connection.

IXMLDOMDocument3 domSubmitDocument =
(IXMLDOMDocument3)thisXDocument.GetDOM("SubmitDocument");

// Setup namespaces

domSubmitDocument.setProperty(

    "SelectionNamespaces",

    "
xmlns:my=\"http://schemas.microsoft.com/office/infopath/2003/myXSD/
2007-02-01T19-18-48\" "

    + "
xmlns:dfs=\"http://schemas.microsoft.com/office/infopath/2003/dataF
ormSolution\" "

    + "  xmlns:ns4=\"http://qdabra.com/webservices/\" ");

// Alter the query parameter values
```

```
domSubmitDocument.selectSingleNode("/dfs:myFields/dfs:queryFields/n
s4:SubmitDocument/ns4:docTypeName") .text = "PT_Timecard";

domSubmitDocument.selectSingleNode("/dfs:myFields/dfs:queryFields/n
s4:SubmitDocument/ns4:xml").text = "<?QdabraDBXL docId='3'
?><Root><a>hello</a><b><c>1</c><c>2</c></b></Root>";

domSubmitDocument.selectSingleNode("/dfs:myFields/dfs:queryFields/n
s4:SubmitDocument/ns4:name").text = "Matt";

domSubmitDocument.selectSingleNode("/dfs:myFields/dfs:queryFields/n
s4:SubmitDocument/ns4:author").text = "Matt";

domSubmitDocument.selectSingleNode("/dfs:myFields/dfs:queryFields/n
s4:SubmitDocument/ns4:description").text = "Latest timecard.";

// Query the web service

WebServiceAdapter2 wsaSubmitDocument =
(WebServiceAdapter2)thisXDocument.DataAdapters["SubmitDocument"];

wsaSubmitDocument.Query();
```

3.3 Working with a Secondary Data Connection setup to Submit Data

When a secondary data connection is set up to submit data, the object model only supports a limited amount of access. The submission process can be initiated, and then the results can be viewed, but other than that there is not much else available.

```
// Submit the data connection as defined in the Data Connection
wizard

WebServiceAdapter2 wsaSubmit =
(WebServiceAdapter2)thisXDocument.DataAdapters["Submit"];

wsaSubmit.Submit();

// Inspect what was returned by the submit

IXMLDOMDocument2 domSubmitOutput =
(IXMLDOMDocument2)thisXDocument.CreateDOM();

domSubmitOutput.async = false;

domSubmitOutput.validateOnParse = false;

domSubmitOutput.loadXML(wsaSubmit.OutputLocation.xml);

domSubmitOutput.setProperty( "SelectionNamespaces",
    "
xmlns:dfs=\"http://schemas.microsoft.com/office/infopath/2003/dataF
ormSolution\" "

    + "  xmlns:ns4=\"http://qdabra.com/webservices/\" ");

string submitSuccess = domSubmitOutput.selectSingleNode(
"/dfs:dataFields/ns4:SubmitDocumentResponse/ns4:SubmitDocumentResul
t/ns4:Success").text;

thisXDocument.UI.Alert( submitSuccess );
```

3.4 Adding Exception Handling when Working with Ex methods

When using the Ex web methods, adding exception handling will allow you to handle exceptions gracefully when they are returned from the web service.

```
// Submit the data connection as defined in the Data Connection
wizard

WebServiceAdapter2 wsaSubmit =
(WebServiceAdapter2)thisXDocument.DataAdapters["SubmitEx"];

try

{

    wsaSubmit.Submit();

}

catch (Exception ex)

{

    thisXDocument.UI.Alert("SubmitEx method failed. " +
ex.Message);

}
```

3.5 Client-side Injection of the QdabraDBXL Processing Instruction

The normal scenario when working with DBXL enabled forms is to open the form template, fill out the form, submit to DBXL, and then immediately close the form. If updates need to be made, the form is reopened from the web service (which injects the QdabraDBXL PI), edited, and then resubmitted. However, it is not always desirable to close the form after initial submission, and then reopen. To remedy this, you can inject the QdabraDBXL PI on the client with code. When doing this, it is best practice to also set the dirty flag to false immediately after a successful submission to the web service so that the user is not confused when they try to close the form and are prompted to save their form, even though it has been successfully saved to the DBXL web service.

1. Open the template.xml file for the form template you are developing.
2. At the very top of the XML, insert a blank QdabraDBXL PI between the other PIs and the root node of the form.

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

<?mso-infoPathSolution name="urn:schemas-microsoft-
com:office:infopath:ProjectTracker_Timecard:-myXSD-2006-06-
09T06-04-22" href="manifest.xsf" solutionVersion="1.0.0.881"
productVersion="11.0.6565" PIVersion="1.0.0.0" ?>

<?mso-application progid="InfoPath.Document"?>

<?QdabraDBXL?>

<my:Timecard><!-- omitted --></my:Timecard>
```

3. In the code that handles the submission of the form to the web service, add code that updates the PI of the current form to include the docId.

```
WebServiceAdapter2 wsAdapter =
(WebServiceAdapter2)thisXDocument.DataAdapters["SubmitDocumen
t"];

    wsAdapter.Submit();



    string docId =
wsAdapter.OutputLocation.selectSingleNode("//ns2:docId").text
;


thisXDocument.DOM.selectSingleNode(String.Format("/processing
-instruction()[local-name(.) = '{0}']", "QdabraDBXL")).text =
String.Format(" docid=\"{0}\"", docId);



    // Make sure the user does not see the save dialog when
closing the form

   thisXDocument.SetDirty(false);
```

4 DBXLDOCUMENTSERVICE

This is the primary web service that will be used when developing applications that utilize the core functionality of DBXL.

4.1 ChangeDocumentTypeOfDocument

Move a DBXL document from one document type to another.

- **Parameters**:
  - **docId**: The document ID of the document to relocate.
  - **oldDocType**: The name of the document type where the document currently resides. This is a failsafe to prevent against accidentally moving the wrong document.
  - **newDocType**: The name of the document type to which the document should be moved.
- **Return:** A StatusInfo object indicating whether the operation succeeded and listing any errors.

4.2 EnumerateDocumentTypeNames

Enumerate all DocumentTypes, returning only their names.

- **Parameters:** none
- **Return**: A simple string list of all Document Types currently defined in DBXL.

4.3 EnumerateDocumentTypes

Enumerate all DocumentTypes.

- **Parameters:** none
- **Return**: A detailed representation of all Document Types currently defined in DBXL. This includes all of the form template binaries, which could make this a very large download.

4.4 GetAllDocumentTypeSummaries

Returns the summary information of all DocumentTypes in DBXL.

- **Parameters:** none
- **Return**: A simplified version of EnumerateDocumentTypes. This method returns only the top-level metadata about each Document Type and excludes the form template binaries, making it a much smaller download. This method also returns extra data:
  - **DocumentCount** - the current number of documents in the Document Type
  - **Link** - the URL that can be used to access the form template for the Document Type

4.5 GetDocument

Returns the Document with the specified DocID.

- **Parameters:**
  - **DocID** - The Document ID of the Document you wish to retrieve.
- **Return**: A detailed representation of the Document specified by the DocID parameter. This includes all metadata and the XML that was originally submitted, minus the processing instructions.
- **Additional Information:** The InfoPath data connection wizard will attempt to derive a schema from the returned data. The *Content* node returned contains the XML schema of the submitted document. If no forms have been submitted, or if InfoPath cannot infer the schema correctly, the *Content* node will be empty. Using GetDocument from the DbxlSampleDocumentService will populate the *Content* node with the SampleData.xml contained in your form template. InfoPath is then able to derive the correct schema representation.

4.6 GetDocumentFromRefId

Returns the Document with the specified RefId.

- **Parameters:**
  - **RefId** - The RefID of the Document you wish to retrieve.
- **Return**: A detailed representation of the Document specified by the RefID parameter. This includes all metadata and the XML that was originally submitted, minus the processing instructions.

4.7 GetDocumentHistory

Returns an array of Documents which correspond to the history of the DocID specified.

- **Parameters:**
  - **DocID** - The Document ID of the Document you wish to retrieve history of
- **Return**: An array of completely detailed representations of all versions of the Document specified by the docId parameter.

4.8 GetDocumentString

Same as **GetDocument**, but returns the document XML as an escaped string instead of XML nodes. This allows preserving the document's Processing Instructions, which is not possible with GetDocument.

- **Parameters:**
  - **DocID** - The Document ID of the Document you wish to retrieve.
- **Return**: A detailed representation of the Document specified by the DocID parameter. This includes all metadata and the document's XML in string form.

4.9 GetDocumentSummaryByType

Returns information about each Document for the specified DocumentType. This method should be used when querying a large number of Documents. The summary returned includes a Link property that may be used for Document Linking and Loading, the promoted properties of the Document, and several other metadata nodes.

- **Parameters:**
  - **DocTypeName** - The name of the Document Type that you wish to retrieve Documents from.
- **Return**: An array of simplified representations of all the Documents in the Document Type specified by DocTypeName. These representations do not include the full XML content of the Document and will likely be much faster to download.

4.10 GetDocumentType

Returns the DocumentType specified by the DocumentType name. This method is used by DAT to administrate the DocumentTypes and their associated mappings.

- **Parameters:**
  - **docTypeName** - The name of the Document Type that you wish to retrieve.
- **Return**: A detailed representation of the Document Type, including the XSN, specified by the docTypeName parameter.

4.11 GetDocumentTypeId

Allows retrieving the numeric ID for a DBXL document type.

- **Parameters:**
  - **documentTypeName:** The name of the document type to look up.
  - **domain:** The name of the domain for the document type to look up. If it is not in a domain, this parameter should be left blank or null.
- **Returns:** A StatusInfo object indicating the operation's success or errors, and an integer with the numeric ID.

4.12 GetDocumentTypePermissionInfo

Returns the definition of the permissions defined for a particular document type.

- **Parameters:**
  - **documentType:** The name of the document type to look up.
- **Returns:** A StatusInfo object indicating the operation's success or errors, and an array of DocumentTypePermissionInfo objects that define the document type's permissions.

4.13 GetDocumentTypeSummariesByGroup

Returns metadata about all of the document types in a particular solution group or domain.

- **Parameters:**
  - **groupName:** The name of a solution group or ##ID, where ID is the ID of a domain.
- **Returns:** A StatusInfo object describing the result of the operation, and an array of DocumentTypeSummaryInfo objects with metadata about the document types in the group.

4.14 GetDocumentTypeSummary

Returns the summary information of the DocumentType specified by the DocumentType name.

- **Parameters:**
  - **DocTypeName** - The name of the Document Type that you wish to retrieve.
- **Return**: A leaner representation of the Document Type specified by the DocTypeName parameter. The XSN is not returned by this method. This is similar to GetAllDocumentTypeSummaries, except that it only returns the summary for one Document Type at a time.

4.15 GetDocumentVersion

Returns the Document specified by the DocID and version number.

- **Parameters:**
  - **DocID** – The document ID you wish to query.
  - **Version -** The version you wish to retrieve
- **Return**: A detailed representation of the version specified for the given DocID.

4.16 GetDocumentsByType

Returns all Documents for the specifed DocumentType. GetDocumentByType is restricted to document type admins.

- **Parameters:**
  - **docTypeName** - The name of the Document Type whose documents you wish to retrieve.
- **Return**: A detailed representation of all documents in the Document Type specified by docTypeName. Since the XML document content is returned, this is likely to be a large download and is restricted to document type admins only. Use GetDocumentSummaryByType for non-administrator access. This method is also used by the My Forms web part to retrieve data to be shown in the view.

4.17 GetListItems

Replicates the Microsoft SharePoint GetListItems web method.

- **Parameters:**
  - **listName** - The name of the Document Type to query.
  - **viewName** - Reserved for future use.
  - **query** - Reserved for future use.
  - **viewFields** - The list of fields to return.
  - **rowLimit** - The maximum number of rows to return.
  - **queryOptions** – Reserved for future use.
- **Return**: Returns rows containing promoted properties of documents for the requested Document Type.
- **Additional Information:** The query and queryOptions parameters are not currently supported, but there are plans in the future to provide rich querying functionality (much like the QueryDocuments web method) through these parameters. This method was provided to maintain compatibility with web services provided by Microsoft SharePoint. The input and output of this method are the same as the web method of the same name in SharePoint Services. Therefore, if you already have an application built around SharePoint web services, you can seamlessly switch to DBXL. More information about how SharePoint handles this web method can be found here: http://msdn2.microsoft.com/en-us/library/lists.lists.getlistitems.aspx.
- **Sample Output:** Note that because of the way the response is formatted InfoPath isn't able to easily figure out the format of the data. Therefore, when this method is used within an InfoPath form, you will only be able to retrieve the item count.

```
<dfs:myFields
xmlns:dfs="http://schemas.microsoft.com/office/infopath/2003/dataFo
rmSolution" xmlns:tns="http://qdabra.com/webservices/"

    xmlns:s1="http://schemas.microsoft.com/sharepoint/soap/"
xmlns:s2="urn:schemas-microsoft-com:rowset"

    xmlns:s3="#RowsetSchema"
xmlns:my="http://schemas.microsoft.com/office/infopath/2003/myXSD/2
007-03-29T06:18:09"

    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"
xmlns:s="uuid:BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882"

    xmlns:dt="uuid:C2F41010-65B3-11d1-A29F-00AA00C14882">
```

```
  <dfs:queryFields>

    <tns:GetListItems>

      <tns:listName>Ramana_Test_29</tns:listName>

      <tns:viewName></tns:viewName>

      <tns:query></tns:query>

      <tns:viewFields></tns:viewFields>

      <tns:rowLimit></tns:rowLimit>

      <tns:queryOptions></tns:queryOptions>

    </tns:GetListItems>

  </dfs:queryFields>

  <dfs:dataFields>

    <GetListItemsResponse xmlns="http://qdabra.com/webservices/">

      <listitems xmlns:s="uuid:BDC6E3F0-6DA3-11d1-A2A3-
00AA00C14882" xmlns:rs="urn:schemas-microsoft-com:rowset"

          xmlns:z="#RowsetSchema" xmlns:dt="uuid:C2F41010-65B3-
11d1-A29F-00AA00C14882"
xmlns="http://schemas.microsoft.com/sharepoint/soap/">

        <rs:data ItemCount="3">

          <z:row DocID="38"
Link="http://SYSTEM37/QdabraWebService/Documents/38/38.xml"
VersionNo="1"

              RefID="1" IsLocked="False" LockedBy=""
LockExpires="Mon, 01 Jan 0001 00:00:00 GMT"

              LockComments="" Name="" Author="" Description=""
CreatedBy="GGK\\shanthi" Created="Thu, 29 Mar 2007 12:04:45 GMT"

              ModifiedBy="GGK\\shanthi" Modified="Thu, 29 Mar 2007
12:04:45 GMT" field1="promoted property1"

              field2="promoted property2" />

          <z:row DocID="39"
Link="http://SYSTEM37/QdabraWebService/Documents/39/39.xml"
VersionNo="1"
```

```
              RefID="2" IsLocked="False" LockedBy=""
LockExpires="Mon, 01 Jan 0001 00:00:00 GMT"

              LockComments="" Name="" Author="" Description=""
CreatedBy="GGK\\shanthi" Created="Thu, 29 Mar 2007 12:05:08 GMT"

              ModifiedBy="GGK\\shanthi" Modified="Thu, 29 Mar 2007
12:05:08 GMT" field1="promoted property1_1"

              field2="promoted property2_1" />

          <z:row DocID="40"
Link="http://SYSTEM37/QdabraWebService/Documents/40/40.xml"
VersionNo="1"

              RefID="3" IsLocked="False" LockedBy=""
LockExpires="Mon, 01 Jan 0001 00:00:00 GMT"

              LockComments="" Name="" Author="" Description=""
CreatedBy="GGK\\shanthi" Created="Thu, 29 Mar 2007 12:05:23 GMT"

              ModifiedBy="GGK\\shanthi" Modified="Thu, 29 Mar 2007
12:05:23 GMT" field1="promoted property1_2"

              field2="promoted property2_2" />

        </rs:data>

      </listitems>

    </GetListItemsResponse>

  </dfs:dataFields>

</dfs:myFields>
```

4.18 GetMyDocumentSummaryByType

Same as GetDocumentSummaryByType, except it limits the Documents returned to those specified by the permission set of the computer that is currently calling the DBXL web method.

- **Parameters:**
  - **docTypeName** - The name of the Document Type that you wish to retrieve documents from.
- **Return:** The same as GetDocumentSummaryByType, except that it returns only documents that you have been granted Document-level permissions on. If no Document- level permissions have been set, then no documents will be returned. This method is also used by the My Forms web part to query for data data when the Show Only My Documents checkbox is checked. To determine the current user alias, the webservice uses the HttpContext.Current.User.Identity.Name to determine the user alias of whoever is calling this web method.

4.19 GetMyDocumentsByType

Returns all Documents for a specified Document Type that match the permission set of the current user calling the web service method.

- **Parameters:**
  - docTypeName - The name of the Document Type that you wish to retrieve documents from.
- **Return:** The same as GetDocumentsByType, except that it returns only documents that you have been granted Document-level permissions on. The web service uses HttpContext.Current.User.Identity.Name to determine the user alias of whoever is calling this web method.

4.20 GetMyListItems

Replicates the Microsoft SharePoint GetListItems web method.

- **Parameters:**
  - **listName** - The name of the Document Type to query.
  - **viewName** - Reserved for future use.
  - **query** - Reserved for future use.
  - **viewFields** - The list of fields to return.
  - **rowLimit** - The maximum number of rows to return.
  - **queryOptions** - Reserved for future use.
- **Return:** This is like GetListItems, except that it returns rows that the current user has document-level permissions to.
- **Additional Information:** See GetListItems.

4.21 LockDocument

Lock the Document specified by the DocID; optionally add comments that tell why the document was locked.

- **Parameters:**
  - **DocID** - The Document ID of the document you wish to lock.
  - **lockComments** - Any comments that you wish to add as to why the document is being locked.
- **Return:** Success/fail, plus the latest metadata about the document that is locked. When a document is locked, it cannot be edited by any other person than the one who locked it, regardless of the permissions model.

4.22 PrepareTemplateForSigning

Prepares a template for signing by fixing up data connections as would be done during a normal publish to DBXL.

- **Parameters:**
  - **DocType**: The document type you wish to query.
  - **overrideEnumDbDatabase**: - a new database connection string for the Data Source field of the Document Type.

  - **template**: XSN template that will be acted upon.
- **Return:**
  - Success or failure, with any applicable error messages if there is a failure. Also, the statistics on how long the call took.

4.23 QueryDocumentWithKey

Returns information about a document that has been submitted with a key.

- **Parameters:**
  - **key:** The key of the document to look up.
- **Returns:** A StatusInfo object describing the result of the operation, and a DocumentKeyDetails object describing the values relating to submitting documents with keys.

4.24 QueryDocuments

Query all XML Documents for the specified Document Type. Filter values using index query info or XPath query to return those that match. QueryDocuments downloads are limited to IIS maximum size of download and do not support sorting.

- **Parameters:**
  - **DocTypeName** - The name of the Document Type you wish to query.
  - **xpath** - An xpath which will be used for filtering before returning the list. If the xpath returns one or more nodes, the document will be included in the result list.
  - **CompareList** - a collection of comparisons to be combined to form a query
  - **IndexCompareInfo** - a single comparison on an index, as defined in DAT
  - **IndexName** - the name of the index, as defined in DAT
  - **Operand** - one of the following:
    - **LT** - less than
    - **LTE** - less than or equal to
    - **EQ** - equal to
    - **GTE** - greater than or equal to
    - **GT** - greater than
    - **NE** - not equal
    - **Like** - fuzzy comparison
  - **Value** - the value to compare against
- **Return:** A list of Documents that match the given query.

4.25 QueryDocumentsHistory

Query all old versions of XML Documents for specified Document Type, applying the XPath query and returning those that match. If the XPath query is blank, all Documents are returned.

- **Parameters:**
  - **DocTypeName** - The name of the Document Type whose history you wish to query.
  - **xpath** - An xpath which will be used for filtering before returning the list. If the xpath returns one or more nodes, the document will included in the result list.
- **Return**: This is like QueryDocuments, except the query is issued against the history of the Document Type specified.

4.26 QueryDocumentsNodeSet

Queries data from within the documents in a document type and returns the queried data as XML.

- **Parameters:**
  - **query:** The query XML to use to retrieve document data. This can be built using the Report Builder included with DBXL.
- **Returns:** A StatusInfo object describing the outcome of the operation, and an XML node with the results. The returned XML node is of the following format:

```
<Rows>
        <row docId="123">
                <Field1>Value</Field1>
                <Field2>ThatValue</Field2>
        </row>
        <row docId="277">
                <Field1>OtherValue</Field1>
                <Field2>AnotherValue</Field2>
        </row>
</Rows>
```

  Here, Field1 and Field2 correspond to the local names of the fields being queried, or to an alias specified in the query.

4.27 RemoveAllDocumentsForType

Removes all Documents associated with a certain DocumentType. Like RemoveDocument, this will remove the XML from DBXL's database as well as all rows that have been shredded into the target database.

- **Parameters:**
  - **DocTypeName** - the name of the Document Type from which to remove all documents.
- **Return**: Success or failure, with any applicable error messages if there is a failure, and the statistics on how long the call took.

4.28 RemoveDocument

Removes the Document specified by the DocID. This will permanently and irretrievably remove the XML from DBXL's database as well as all rows that have been shredded into the target database.

- **Parameters:**
  - **DocID** - The DocID of the document you wish to remove.
- **Return:** Success or failure, with applicable error data, and the statistics on how long the call took to execute.

4.29 RemoveDocumentType

Removes the DocumentType specified by the name. This method will only successfully complete if the specified DocumentType has no Documents associated with it. To remove Documents, call RemoveDocument, RemoveDocuments, or RemoveAllDocumentsForType.

- **Parameters:**

- **DocTypeName** - The name of the Document Type you wish to remove.

- **Return:** Success or failure, with applicable error data, and the statistics on how long the call took to execute.
- **Additional Information:** You must first remove all documents within a Document Type before you can successfully call this web method.

4.30 RemoveDocuments

Removes all Documents specified by the DocIDs. This will permanently and irretrievably remove the XML from DBXL's database as well as all rows that have been shredded into the target database for each of the DocIDs specified.

- **Parameters:**
  - **DocIDs** - The Document IDs of the documents you wish to remove.
- **Return:** Success or failure, with applicable error data, and the statistics on how long the call took to execute.

4.31 Search

Searches all documents matching the specified document type for the search phrase. SearchType controls what data is searched.

- **Parameters:**
  - **searchPhrase**: String that the service will search for
  - **searchType**: User selects between "All", "Metadata" or "Data"
  - **docType**: Document type name to search within
- **Return:**
  - A summary of the document(s) found.
  - Success or failure, with applicable error data, and the statistics on how long the call took to execute.

4.32 SearchEx

Searches all documents matching the specified document type for the search phrase. **SearchMethod** controls what type of matching is performed. **SearchType** controls what data is searched.

- **Parameters**:
  - **searchPhrase**: string that contains the search term(s).
  - **searchMethod**: Select the method – options are Basic, Contains or Freetext
  - **searchType**: Search Metadata, Data (full form's XML content) or both (All)
  - **docType**: string to indicate the DocType that will be searched
  - **getProperties**: Boolean, determines whether to return properties promoted from the InfoPath form for each record returned.
  - **start** and c**ount**: int. **Start** and **count** can be used for paging of result sets. To return the entire result set, specify -1 for both. To return a portion of the result set, specify the row number of the first row to return for start and specify the maximum number of rows to return for count.
- **Return**: xml schema of the search results

4.33 SetDocumentType

Either creates or edits a DocumentType in DBXL. If DBXL already contains a DocumentType specified by the name then it is edited, if not a new one is created.

- **Parameters:** See the "Overview of DAT Fields" document.
- **Return:** Success or failure, with applicable error data, and the statistics on how long the call took to execute.
- **Additional Information:** This web method will generally only be used by the DBXL Administration Tool (DAT), the DBXL Migration Tool, or any other tool that is created to administrate document types.

4.34 SetDocumentTypeMapping

Updates the mapping for a DocumentType.

- **Parameters**:
  - **docTypeName**: Document Type whose mapping will be updated.
  - **mappingInfo**: A NodeMapInfo object defining the document type's mapping.
- **Return:** Success or failure, with applicable error data. Also, the statistics on how long the call took to execute.

4.35 SubmitDocument

Add or Update the Document designated by the DocID in the Processing Instruction (PI) of the XML being submitted. If the XML being submitted does not have a QdabraDBXL PI, then this method will add the Document to DBXL, therefore creating a new DocID. This new DocID will be injected into the submitted XML upon retrieving the XML from DBXL via Document Linking and Loading. Once the PI has been injected into the XML, this method will update the Document according to the DocID stored in the PI. Returns: int docId and string refId.

- **Parameters:**
  - **DocTypeName** - The name of the Document Type to submit the document to.
  - **xml** - The XML of the document to be submitted. This XML may or may not include the QdabraDBXL Processing Instruction.
  - **name** - Meta data to attach to the document.
  - **author** - Meta data to attach to the document.
  - **description** - Meta data to attach to the document.
- **Return:** Success or failure, with applicable error data, the statistics on how long the call took to execute, the latest Reference ID, and the Document ID that was assigned, if it was a new Document.
- **Additional Information**: This method is used either for adding new documents to a Document Type, or for updating documents that already exist. When submitting a new document, the XML should look exactly how it does when InfoPath saves it. However, when updating an existing document, the QdabraDBXL Processing instruction must be present in the XML so that DBXL will know which document to overwrite. Additionally, when the QdabraDBXL PI is present, the docTypeName, name, author, and description parameters are ignored because these values are pulled from the PI.

4.36 SubmitDocumentWithKey

Submits a document to DBXL along with a unique key to associate with that document, so information about that document can be found using that key.

- **Parameters:**
  - **xml:** The XML of the document to submit, represented as a string.
  - **documentType:** The document type to submit to. (Can be left blank if the document type can be inferred from the document's PIs)
  - **key:** The key to use for this document.
  - **clientDateTime:** The local time at which the document is being submitted.
  - **name:** An optional value to use as the title for the submitted document.
  - **author:** An optional value indicating the user who is submitting the document.
  - **description:** An optional value to use as a description for the document.
- **Returns:** A StatusInfo object indicating the success or failure of the submit.
  Note: This web method will throw an exception upon failure, even in the non-Ex version of the web service. This is because the StatusInfo object cannot be inspected when using a Submit data connection, and the exception will halt any rules from continuing to execute.

4.37 UnlockDocument

Unlocks the Document specified by the DocID.

- **Parameters:**
  - DocID - The DocID for the document that you wish to unlock.
- **Return:** Success or failure, with applicable error data, the statistics on how long the call took to execute, and a summary of with the latest information about the document that was unlocked.

5 DBXLIMPORTEXPORT

This set of web methods allows batch importing or exporting of documents or document type definitions. When inputting a file or folder path, the parameter should specify a path that is accessible from the perspective of the web service.

For instance, if the file path was set to *C:\*, then the web service would read or write into that directory on the server that is hosting the DBXL web services. Additionally, you may specify a network path if you want to save to another machine; the format of these paths will be *\\computer\sharedFolder*. Be sure that the folder has the correct permissions defined so that the web service account has permission to write. When accessing local folders, this will usually be the NETWORK SERVICE account. If accessing a network share, the NETWORK SERVICE account will be validated using the machine account (ex. YOURDOMAIN\MACHINENAME$).

5.1 ExportDocumentType

Exports a document type definition.

- **Parameters:**
  - **docTypeName** - The name of the Document Type that you wish to export.
  - **filePath** - The filePath to save the mapping.xml file to. This parameter must include the filename that will be used, for example, *C:\temp\mapping.xml*.
- **Return:** Success or failure with error message and execution statistics.

5.2 ExportDocuments

Exports documents for a specified type. Each document is written as a separate XML file in the specified folderPath.

- **Parameters:**
  - **docTypeName** - The name of the Document Type that you wish to export all documents of.
  - **folderPath** - The folder path to export the documents to.
- **Return:** Success or failure with error message and execution statistics.
- **Additional Information:** The XML content of all of the documents in the specified Document Type will be saved to the specified folder. Notice that this does not include metadata that might surround the form, such as workflow values, notifications, or the name, author, and description values. The XML files will be named with the name metadata, or if it is blank, their DocID.

5.3 ImportDocumentType

Imports a document type definition.

- **Parameters:**
  - **docTypeName** - The name that the imported Document Type will have in the new DBXL instance.
  - **filePath** - The filePath to the mapping.xml file that will be used to import the Document Type. This parameter must include both the directory path and the filename, for example, *C:\temp\mapping.xml*.
  - **overrideDbConnectionString** - a new database connection string for the Data Source field of the Document Type.
  - **overrideEnumDbDatabase** - the name of the target database in the new DBXL environment that should be used with the Document Type.
- **Return:** Success or failure with error message and execution statistics.
- **Additional Information:** When developing a form template that is heavily integrated with DBXL it is generally assumed that there will be many data connections to the EnumDB web service. When importing a Document Type the name of the database that the EnumDB web service data connection should query will often change. With the overrideEnumDbDatabase parameter you can update all of these fields on import.

5.4 ImportDocuments

Imports documents for a specified type. This method reads all XML files in the folderPath and adds them as new documents to DBXL. The importExistingRefId parameter may take values of 'true' or 'false'. If 'true' is specified, a Ref. ID XPath (on the general tab of DAT) must be defined for the docType.

- **Parameters:**
  - **docTypeName** - The name of the Document Type that you wish to fill with documents from the hard drive.
  - **folderPath** - The folder path to all of the XML files.
- **Return:** Success or failure with error message and execution statistics.

6 DBXLADMIN

The DbxlAdmin.asmx web service contains a collection of methods that can be used for a variety of smaller, diagnostic tasks. Several of these methods are called from the DBXL Admin Tool (DAT).

6.1 BootstrapSystemDocumentTypes

Used by the installer to initialize DBXL's fundamental document types. You should generally not need to invoke this method yourself.

6.2 BulkEditDocuments

Abstracts calls to EditDocuments so that multiple edits may be done with one call.

- **Parameters:**
  - **ID –** A label given to each operation. This can take any value and it's meant to help the developer identify the individual edits.
  - **DocIDs –** The DocID you wish to edit.
  - **XPaths –** The XPath whose value you wish to modify.
  - **newValues –** The new value that will be inserted into the above Xpath.
- **Return:**
  - Success or failure with error message and execution statistics.
  - **TotalDocumentsChanged** – an integer indicating the total number of documents that were modified.
  - **TotalChanges** – an integer indicating the total number of modifications performed.
  - **DocumentsSkipped** – DocIDs of any documents that were skipped in the operation.

6.3 DescribeDatabaseSchema

Returns an enumeration of a database in XML.

- **Parameters:**
  - **docTypeName** - The name of the Document Type whose database schema you wish to view.
- **Return:** An XML representation of the database that the specified Document Type shreds into.
- **Additional Information:** This returns a schema based on the connection string saved for the specified Document Type.

6.4 EditAllDocuments

Bulk edit all documents in a Document Type. Each XPath must be matched by a new value. XPaths may contain any conditional logic so that only certain nodes are updated. To update the Name/Author/Description nodes, you may use the DBXL mapping tokens. DBXL::Name will update all name values. DBXL::Name[. = 'oldValue'] updates all Name values where the old value is equal to oldValue. Only the self value test is allowed for DBXL mapping tokens.

- **Parameters:**
  - **docTypeName** – The name of the Document Type that you wish to modify.
  - **XPaths –** The XPath(s) whose value you wish to modify.
  - **newValues –** The new value(s) that will be inserted into the above Xpath.
- **Return:**
  - Success or failure with error message and execution statistics.

- **TotalDocumentsChanged** – an integer indicating the total number of documents that were modified.
- **TotalChanges** – an integer indicating the total number of modifications performed.
- **DocumentsSkipped** – DocIDs of any documents that were skipped in the operation.

6.5 EditDocuments

Bulk edit a set of documents. Each XPath must be matched by a new value. XPaths may contain any conditional logic so that only certain nodes are updated. To update the Name/Author/Description nodes, you may use the DBXL mapping tokens. DBXL::Name will update all name values. DBXL::Name[. = 'oldValue'] updates all Name values where the old value is equal to oldValue. Only the self value test is allowed for DBXL mapping tokens.

- **Parameters:**
  - **DocIDs –** The DocID you wish to edit.
  - **XPaths –** The XPath whose value you wish to modify.
  - **newValues –** The new value that will be inserted into the above Xpath.
- **Return:**
  - Success or failure with error message and execution statistics.
  - **TotalDocumentsChanged** – an integer indicating the total number of documents that were modified.
  - **TotalChanges** – an integer indicating the total number of modifications performed.
  - **DocumentsSkipped** – DocIDs of any documents that were skipped in the operation.

6.6 ForceUnlockDocument

Removes any lock on the specified document.

- **Parameters:**
  - **DocID** - The DocID of the document to unlock.
- **Return:** Success or failure with error message and execution statistics.

6.7 GetDbxlVersion

Retrieves the DBXL Web service's version number.

- **Parameters:** None.
- **Return:** The version of the DBXL instance which was queried. The format of the version is 2.2.date.revisiontime, where date is the number of days since January 1, 2000. The revisiontime number (last number) is [seconds past midnight / 2].

6.8 InitializeDatabase

Configures database after setup.

- **Parameters:** None.
- **Return:** Success or failure with error message and execution statistics.
- **Additional Information:** This method simply pushes any simple schema changes into the backend DBXL database. It cannot handle large schema changes, such as deletion or rearrangement of hierarchy between the tables, but these changes would result in a new minor release of DBXL. This

method should be called after each upgrade of DBXL to ensure that the backend database is how the web service expects it.

6.9 PublishDATFormTemplate

Publish Database Accelerator Admin Tool form.

- **Parameters:**
  - **templateFilePath** - The path to the template.xsn for the DAT that is to be published.
- **Return:** Success or failure with error message and execution statistics.
- **Additional Information:** This method will take publish the template specified to a Document Type named "DAT". If this Document Type already exists, it will overwrite the template there. Once DAT is published, you should be able to access it from Document Linking and Loading.

6.10 ReshredAllDocuments

Refreshes database-mapped values for a document type.

- **Parameters:**
  - **docTypeName** - The name of the Document Type that contains all of the documents to be reshredded.
- **Return:** Success or failure with error message and execution statistics.
- **Additional Information:** This method can be very useful when debugging the database mapping. This method will effectively reconstitute the target database from all of the XML that is present. So, if the target database becomes slightly corrupted, you can dump all of the data and then reconstitute it with this method. This is the method that is called when the "Reshred All Documents" button is clicked in the "Document Catalog" view of the DBXL Admin Tool.

6.11 ReshredDocument

Refreshes database-mapped values for a document.

- **Parameters:**
  - **DocID** - The DocID of the Document to be reshredded.
- **Return:** Success or failure with error message and execution statistics.

6.12 TestConnectionString

Used in the Database Accelerator Admin Tool to determine whether or not a connection string is functional or not. This method attempts to use the connection string specified to establish a connection to the target database from the perspective of the DBXL Web Service. If a connection is establish this method returns true, otherwise any error messages are returned.

- **Parameters:**
  - **ConnectionString** - The database connection string that is to be tested.
- **Return:** Success or failure with error message and execution statistics.
- **Additional Information:** This method tests the given connection string to make sure that the web service can connect to the database that you wish to shred XML into. If there are any problems with the login, more information can be found in the returned data.

6.13 TestEditAllDocuments

Same as EditAllDocuments, but changes are not committed. The total changes are calculated and returned.

6.14 TestEditDocuments

Same as EditDocuments, but changes are not committed. The total changes are calculated and returned.

6.15 UpdateIndexesForDocumentType

Updates the indexes for a DocumentType.

- **Parameters:**
  - **docTypeName** - The name of the Document Type that the indexes should be updated on.
- **Return:** Success or failure with error message and execution statistics.
- **Additional Information:** The indexes for a Document Type are pulled from the XML of the respective Documents. Once the indexes are created, searching through the Document Type should be much faster and more accurate. Note that this method has nothing to do with Smart Search technology.
  If this web method should fail due to a timeout, we recommend using the bulk reshred functionality in the Migration Tool to update a document type's indexes.

7 ADUSERINFO

7.1 FindUsersByAlias

Searches for all users in AD whose username matches a certain search parameter and returns their display name and username.

- **Parameters:**
  - **alias:** The alias or partial alias to search for.
  - **searchType:** The type of search to perform – possible values are StartsWith, EndsWith, Contains, or Exact
- **Returns:** A DropdownList object, containing the display name and username for any users that were found.

7.2 FindUsersByName

Searches for all users in AD whose display name matches a certain search parameter and returns their display name and username.

- **Parameters:**
  - **name:** The name or partial name to search for.
  - **searchType:** The type of search to perform – possible values are StartsWith, EndsWith, Contains, or Exact
- **Returns:** A DropdownList object, containing the display name and username for any users that were found.

7.3 GetAllUsers

Returns a list of all of the users in the AD system.

- **Parameters:**
  - **OU:** The OU (Organizational Unit) to search within. Can be blank to not restrict by OU.
  - **filter:** An LDAP filter to use in finding certain users. Can be blank to apply no filter.
  - **maxCount:** The maximum number of results to return.
- **Returns:** A DropdownList object, containing the display name and username for any users that were found.

7.4 GetEmployeesForManager

Returns a list of all of the users who have a certain manager.

- **Parameters:**
  - **managerAlias:** The username of the manager whose directreports should be found.
- **Returns:** A DropdownList object, containing the display name and username for any users that were found.

7.5 GetGroupsForUser

Returns a list of all active directory groups to which a certain user belongs.

- **Parameters:**
  - **alias:** The username of the user whose groups should be returned.
  - **recursive:** A Boolean value – False to return only groups to which the user directly belongs, and True to return groups that contain the user's groups.
- **Returns:** A DropdownList object, containing the name of each group that was found (as both the value and display value).

7.6 GetManagerAlias

Returns the alias for a user's manager, if one is set.

- **Parameters**: username
- **Returns**: a string with a value equal to the user's manager alias.

7.7 GetMembersOfGroup

Returns a list of all of the users who are in a certain group.

- **Parameters:**
  - **groupAlias:** The name of the group to search for.
  - **maxCount:** The maximum number of results to return.
- **Returns:** An array of DropdownList items, containing the display name and username for any users that were found.

7.8 GetMyDomains

Reserved for future use.

7.9 GetMyInfo

Gives the Active Directory properties of the currently logged-in user.

- **Parameters**: none
- **Returns**: Active Directory Properties table, listing all Keys and Values for the current logged-in user.

7.10 GetUserInfo

Gives the Active Directory properties of the specified user name.

- **Parameters**:
  - **username** – the username you wish to query with Active Directory.
- **Returns**: Active Directory Properties table, listing all Keys and Values for the provided username.

7.11 IsUserMemberOfGroup

Returns true if the user is part of the specified group, false otherwise.

- **Parameters**:
  - **username –** The username you wish to run the method on.
  - **groupAlias –** The group alias you want to query.
- **Returns**:
  - **IsUserMemberOfGroupResult**: Boolean value indicating whether username provided is a member of the groupAlias entered.

8 QUERYDB

For more details on the QueryDB web service, please see the QueryDB User Guide.

8.1 DescribeDatabaseSchema

Scans the schema in the specified connection string and returns an XML Document describing it. The dataSource parameter should be a connection string as passed to the SqlConnection constructor the .NET framework.

- **Parameters**:
  - **dataSource:** connection string parameter
- **Returns**:
  - XML Document that fully describes the database schema

8.2 DescribeDatabaseSchemaFromDatabaseName

Scans the schema in the specified database and returns an XML Document describing it. The database parameter should be a database specified in DBXL's web.config, or a database alias preceded and followed by pound signs.

- **Parameters**:
  - **database:** The name of a database declared in web.config, or a database alias preceded and followed by pound signs.

- **Returns**:
  - XML Document that fully describes the database schema

8.3 GetColumnsXMLQuery

Returns the result of the query specified by the queryXml paramenter. This method accepts un-escaped XML for the queryXml parameter. This method is meant for use in code, as its parameter is XML that must be submitted via code.

8.4 GetColumnsXMLStrQuery

Returns the result of the query specified by the queryXml paramenter. The XML representing the queryXml parameter must be escaped for this method.

- **Parameters**:
  - **queryXml**: query parameter, as generated by Qdabra's Query Builder.
- **Returns**:
  - XML containing the returned values requested via the xml query.

8.5 GetDropdownEntries

Returns a list of values and optionally display values to easily bind with a dropdown control. Default sort is ascending by valueColumn (displayColumn if specified).

- **Parameters**:
  - **database**: database for connection
  - **table**: table for connection
  - **valueColumn**: table column
  - **displayColumn**: table column that will be displayed
  - **sortColumn**: column to sort by
  - **sortOrder**: ASC or DESC
  - **filterXml**: <filter> node from query generated by Query Builder.
- **Returns**:
  - Success or failure, with applicable error data. Also, the statistics on how long the call took to execute.
  - Table containing two columns, valueColumn and displayColumn, which can be used to populate a dropdown control.

8.6 GetLicenseInfo

Returns information about the current QueryDB licenses installed and in use.

- **Parameters:** None
- **Returns:** Integer values indicating the number of QueryDB licenses installed, the number in use, and the number still available.

9 DBXLXSNSERVICE

9.1 GetSampleDataXml

Returns the sampledata.xml from the Form's XSN file. This describes the schema of the form. Create a data connection to this if you need to load the XML data of another document type into your form and you want to bind elements in your view to the elements of this other document type.

- **Parameters**:
  - **docTypeName**: Document Type Name to extract Sample Data from.
- **Returns**:
  - Success or failure, with applicable error data. Also, the statistics on how long the call took to execute.
  - **sampleDataXml** from the form's xsn file

10 DBXLLICENSING

10.1 AddLicense

Adds a DBXL license. Assumes that the licenseBlob comes from a File Attachment control in InfoPath and includes the IFA header.

- **Parameters**:
  - **contactEmail**: Email address associated with the license being attached
  - **licenseBlob**: base64Binary file that contains the license information (file provided by Qdabra)
- **Additional Information:** This method assumes that the licenseBlob comes from a File Attachment control in InfoPath. For development, please use **AddLicenseFromFile**.

10.2 AddLicenseFromFile

Adds the DBXL license from a local file.

- **Parameters**:
  - **contactEmail**: Email address associated with the license being attached
  - **filename**: A string that contains the local path to the license file.
- **Returns**: Success or failure, with applicable error data, and the statistics on how long the call took to execute.

10.3 GetLicenseInfo

Returns information about the DBXL licenses installed

- **Parameters**: none.
- **Returns**:
  - Success or failure, with applicable error data, and the statistics on how long the call took to execute.

- The Serial Number, contact name, contact email and expiration date for every license configured on the system.
- License allowed count for Documents (int), Document Type (int), QueryDb databases (int) and Active Directory License (Boolean).

10.4 GetLicenseInfo2

Returns information about the DBXL licenses installed. Similar to GetLicenseInfo, above, but does not return success/failure field, error data or execution time.

- **Parameters**: none.
- **Returns**:
  - The Serial Number, contact name, contact email and expiration date for every license configured on the system.
  - License allowed count for Documents (int), Document Type (int), QueryDb databases (int) and Active Directory License (Boolean).

10.5 RemoveLicense

Uninstalls the specified license.

- **Parameter**: The Serial Number for the license that you wish to remove.
- **Returns**: Success or failure, with applicable error data. Also, the statistics on how long the call took to execute.

11 DBXLRESOURCE

11.1 CreateResource

Creates a named resource (Requires administrative permission for the document type)

- **Parameters**:
  - **docTypeName**: Document type for the resource being created
  - **resourceName**: name for the resource
  - **description**: description of the resource being created
  - **docID**: DocID indicating the xml source of the resource. Use 0 for resources that do not use this option.
  - **allowUserVersions**: boolean value to indicate whether the individial Active Directory users are allowed to save customized versions of the XML.
- **Returns**:
  - Success or failure, with applicable error data. Also, the statistics on how long the call took to execute.

11.2 DeleteResource

Deletes the named resource (Requires administrative permission for the document type)

- **Parameters**:
  - **docTypeName**: document type configuration in DAT
  - **resourceName**: name for the resource being deleted

- **Returns**:
    - Success or failure, with applicable error data. Also, the statistics on how long the call took to execute.

11.3 GetResourceXml

Returns the named resource xml

- **Parameters**:
    - **docTypeName**: document type configuration in DAT
    - **resourceName**: name for the resource being queried
- **Returns**:
    - Success or failure, with applicable error data. Also, the statistics on how long the call took to execute.
    - An XML structure for the Resource requested.

11.4 GetUserResourceXml

Returns the calling user's resource xml for the named resource

- **Parameters**:
    - **docTypeName**: document type configuration in DAT
    - **resourceName**: name for the resource being queried
- **Returns**:
    - Success or failure, with applicable error data. Also, the statistics on how long the call took to execute.
    - An XML structure for the Resource requested for the currently logged in user.

11.5 ListResources

Returns a list of the defined resources for the specified doctype

- **Parameters**:
    - **docTypeName**: document type configuration in DAT
- **Returns**:
    - Success or failure, with applicable error data. Also, the statistics on how long the call took to execute.
    - A list of resources with the details CreatedBy, DateCreated, ModifiedBy, DateModified

11.6 ModifyResource

Updates the named resource (Requires administrative permission for the document type)

- **Query Parameters**:
    - **docTypeName**: Document type configuration
    - **resourceName**: name for the resource being modified
- **Modifiable Parameters:**
    - **description**: description of the resource
    - **docID**: DocID indicating the xml source of the resource. Use 0 for resources that do not use this option.

- **allowUserVersions**: boolean value to indicate whether the individial Active Directory users are allowed to save customized versions of the XML.

- **Returns**:
  - Success or failure, with applicable error data. Also, the statistics on how long the call took to execute.

11.7 ResetUserResourceXml

Resets the named per-user resource file to the default

- **Parameters**:
  - **docTypeName**: Document type configuration
  - **resourceName**: name for the resource being reset
- **Returns**:
  - Success or failure, with applicable error data. Also, the statistics on how long the call took to execute.

11.8 SaveResourceXml

Saves the named resource file

- **Parameters**:
  - **docTypeName**: Document type configuration
  - **resourceName**: name for the resource being modified
  - **xml**: xml structure to be set. Cannot be submitted as a string.
- **Returns**:
  - Success or failure, with applicable error data. Also, the statistics on how long the call took to execute.

11.9 SaveUserResourceXml

Saves the named per-user resource file

- **Parameters**:
  - **docTypeName**: Document type configuration
  - **resourceName**: name for the resource being modified
  - **xml**: custom xml (per-user) structure to be set.
- **Returns**:
  - Success or failure, with applicable error data. Also, the statistics on how long the call took to execute.

12 QUERYSHAREPOINT

For more details on the QuerySharePoint web service, please see the QuerySharePoint User Guide.

12.1 DescribeListSchema

Scans the schema for the specified SharePoint list and returns an XML Document describing it. The listUrl parameter should be the root URL of a SharePoint list).

- **Parameters**:
  - **listUrl**: The URL to the SharePoint list
- **Returns**: xml schema for the SharePoint list

12.2 DescribeListSchemaSiteAndGuid

Scans the schema for the specified SharePoint list and returns an XML Document describing it. The siteUrl parameter should be the URL of a SharePoint site. The listId should be the guid uniquely identifying the SharePoint list on the site. Use this method if the list may be moved or renamed in the future since the list name guid is less likely to change.

- **Parameters**:
  - **siteUrl**: The URL to the SharePoint site.
  - **listId**: the guid uniquely identifying the SharePoint list on the site
- **Returns**: xml schema for the SharePoint list

12.3 GetDropdownEntries

Returns a list of values and optionally display values to easily bind with a dropdown control. Default sort is ascending by valueColumn (displayColumn if specified).

- **Parameters**:
  - **listUrl:** root URL of the SharePoint list
  - **valueColumn**: column
  - **displayColumn**: column that will be displayed
  - **sortColumn**: column to sort by
  - **sortOrder**: ASC or DESC
  - **filterXml**: <filter> node from query generated by Query Builder.
- **Returns**:
  - Success or failure, with applicable error data. Also, the statistics on how long the call took to execute.
  - Table containing two columns, valueColumn and displayColumn, which can be used to populate a dropdown control.

12.4 GetDropdownEntriesSiteAndGuid

Returns a list of values and optionally display values to easily bind with a dropdown control. Default sort is ascending by valueColumn (displayColumn if specified).

- **Parameters**:
  - **siteUrl**: The URL to the SharePoint site.
  - **listId**: the guid uniquely identifying the SharePoint list on the site
  - **webId:** the guid that uniquely identifies the site (web) in a SharePoint site collection. WSS 3.0 requires this in the API in addition to the listUrl.
  - **valueColumn**: column
  - **displayColumn**: column that will be displayed
  - **sortColumn**: column to sort by
  - **sortOrder**: ASC or DESC
  - **filterXml**: <filter> node from query generated by Query Builder.
- **Returns**:

- Success or failure, with applicable error data. Also, the statistics on how long the call took to execute.
- Table containing two columns, valueColumn and displayColumn, which can be used to populate a dropdown control.

12.5 GetListItems

Perform GetListItems on the SharePoint list and return the resulting rowset. The listUrl is the root URL of the SharePoint list. The following arguments match the functionality of the arguments to the SharePoint Lists.asmx method GetListItems.

- **Parameters**:
  - **listUrl**: root URL of the SharePoint list
  - **query** (xml): This query is built using Qdabra's Query Builder tool and is used for filtering results.
  - **queryOptions** (xml): matches the functionality of the same argument in the SharePoint Lists.asmx method GetListItems
- **Returns**: xml for the items being returned

12.6 GetListItemsSiteAndGuid

Perform GetListItems on the SharePoint list and return the resulting rowset. The siteUrl is the URL of the SharePoint site containing the list. The following arguments match the functionality of the arguments to the SharePoint Lists.asmx method GetListItems. Use this method if the list may be moved or renamed in the future since the list name guid is less likely to change.

- **Parameters**:
  - **siteUrl**: The URL to the SharePoint site.
  - **listId**: the guid uniquely identifying the SharePoint list on the site
  - **query** (xml): This query is built using Qdabra's Query Builder tool and is used for filtering results.
  - **queryOptions** (xml): matches the functionality of the same argument in the SharePoint Lists.asmx method GetListItems
- **Returns**: xml for the items being returned

12.7 GetListItemsStr

Perform GetListItems on the SharePoint list and return the resulting rowset. The listUrl is the root URL of the SharePoint list. The following arguments match the functionality of the arguments to the SharePoint Lists.asmx method GetListItems.

- **Parameters**:
  - **listUrl**: root URL of the SharePoint list
  - **query** (xml): This query is built using Qdabra's Query Builder tool and is used for filtering results.
  - **queryOptions** (xml): matches the functionality of the same argument in the SharePoint Lists.asmx method GetListItems
- **Returns**: xml for the items being returned

12.8 GetListItemsStrSiteAndGuid

Perform GetListItems on the SharePoint list and return the resulting rowset. The siteUrl is the URL of the SharePoint site containing the list. The following arguments match the functionality of the arguments to the SharePoint Lists.asmx method GetListItems. Use this method if the list may be moved or renamed in the future since the list name guid is less likely to change.

- **Parameters**:
  - **siteUrl**: The URL to the SharePoint site.
  - **listId**: the guid uniquely identifying the SharePoint list on the site
  - **query** (xml): This query is built using Qdabra's Query Builder tool and is used for filtering results.
  - **queryOptions** (xml): matches the functionality of the same argument in the SharePoint Lists.asmx method GetListItems
- **Returns**: xml for the items being returned