TIBCO iWay® Service Manager

XML Archive User's Guide

Version 8.0 and Higher
March 2021
DN3502299.0321

Contents

Chapter 1

Installing the iWay XML Archive

This section discusses the installation and configuration of the iWay XML Archive. It assumes a Windows-based installation, but the iWay XML Archive is also supported on the Linux and UNIX® operating systems. You can use the sample values provided to create a sample configuration.

In this chapter:

iWay XML Archive Installation Requirements

The following software is required:

- iWay Service Manager (iSM) Version 7.0 and higher.

  The iSM installation also requires the iWayXMLArchive.iia binary file to support the iWay XML Archive application. To obtain the required application binary file, contact Information Builders Technical Support at *http://techsupport.iwaysoftware.com*.

- MongoDB® 1.8.0 or higher for a 32-bit or 64-bit environment. You can download mongoDB from:

  http://www.mongodb.com/downloads

- Mongo® Java driver version 2.5, which you can download from:

  https://mvnrepository.com/artifact/org.mongodb/mongo-java-driver/2.5.3.

  Note: There are known issues when using Mongo Java driver version 2.7.3. It is recommended to use Mongo Java driver version 2.5, which has been used for iWay XML Archive certification.

In addition, the end user must agree to the Apache License Version 2.0, found at:

http://apache.org/licenses/LICENSE-2.0

Installing iWay Service Manager

To install iWay Service Manager (iSM):

1. Run the iSM 7.0 installation.
2. During the installation, select the *iWay XML Archive Extensions* category.
3. Copy the mongo-2.5.jar file (or higher) to the *iwayhome*\lib directory.
4. Start the iSM service.

Note: For a complete set of installation instructions, see the *iWay Installation and Configuration Guide*.

Installing mongoDB

To install mongoDB:

1. Create a new directory to store the mongo binary files. For example, c:\mongo.

   Note: It is recommended that you put the mongoDB directories outside of the iSM installation directories. If you need to reinstall iSM, the directories will not be erased.
2. Extract the downloaded mongoDB ZIP file to the c:\mongo directory.
3. Create a new directory to store the mongoDB data. For example, c:\mongo\data\db.
4. As an administrator, open a command prompt, and navigate to the c:\mongo\mongodb-*architecture_version*\bin directory.
5. At the administrator command prompt, type the following command:

   ```
   mongod –-install --dbpath c:\mongo\data\db –logpath c:\mongo\log.txt
   ```
6. Start the mongo service, and verify that it is set to autostart.

   Note: You can start the mongo service from the command line, which is useful for debugging purposes. Type the following at the command prompt:

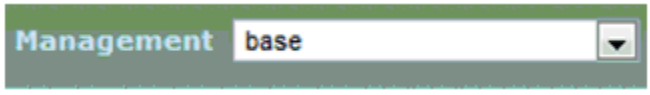
   ```
   mongod --dbpath c:\mongo\data\db
   ```

Configuring the iWay XML Archive Application

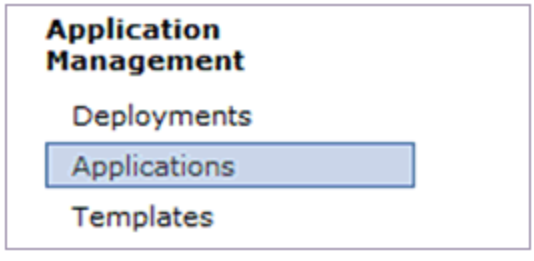
The following procedures describe how to configure the iWay XML Archive Application.

Procedure: How to Configure the iWay XML Archive Application

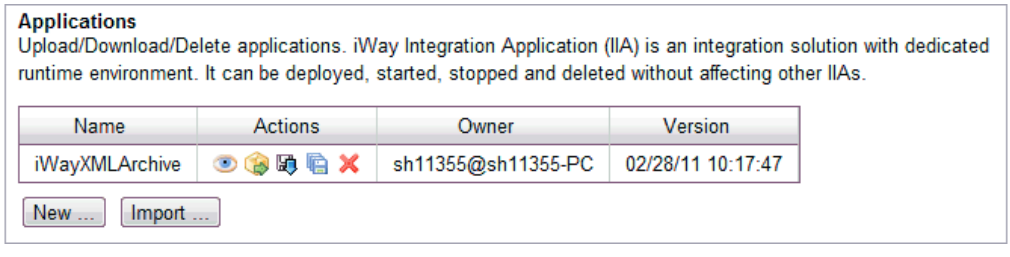
1. From the iSM Console, click the *Management* link, as shown in the following image.



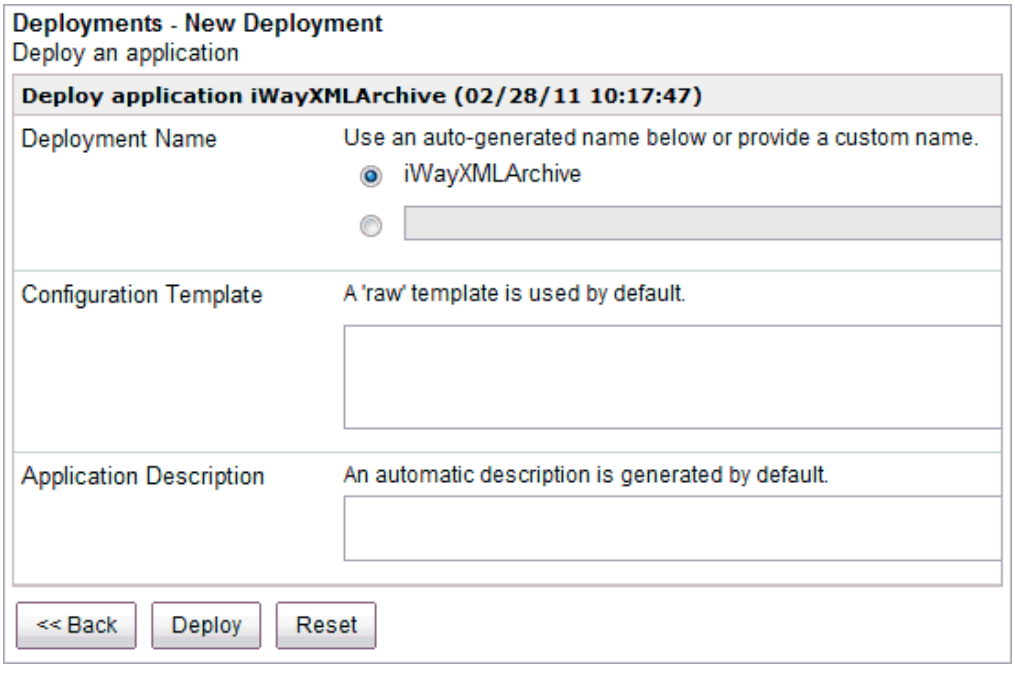
2. Click *Applications*, as shown in the following image.



3. From the Applications list, select *Deploy* for the application named iWayXMLArchive.

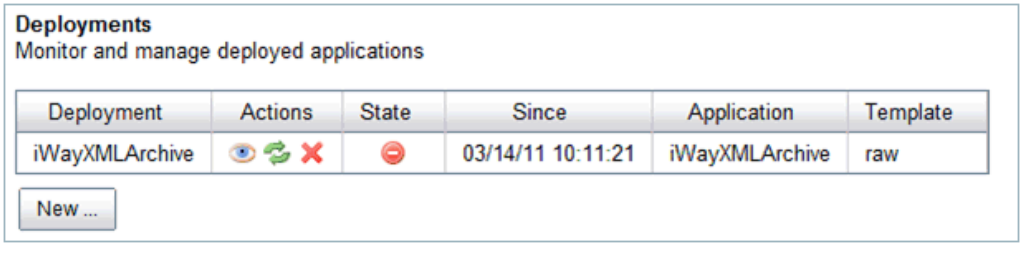


4. Leave iWayXMLArchive selected, and click *Deploy*, as shown in the following image.

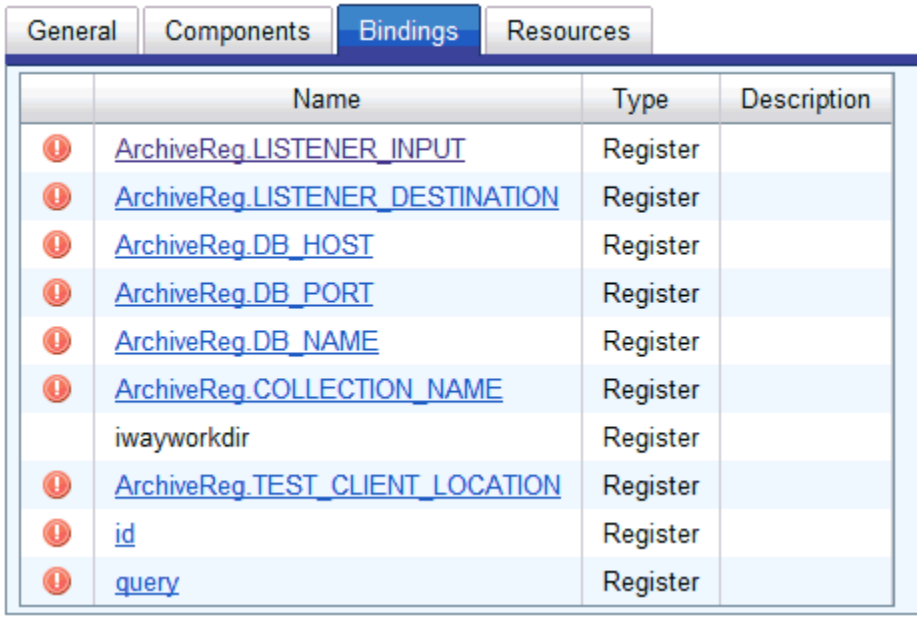


Once the application has been deployed, you are automatically switched to the Deployments link on the left.

5. Click the eye icon next to the iWayXMLArchive deployment, shown in the image below.



6. Click the *Bindings* tab. You will see that each register has a red asterisk next to it.



7. Select each register in order, and type the correct values, as follows:

   - **ArchivReg.LISTENER_INPUT.** The directory on disk from which a file listener reads XML documents to put into the archive. For example,*c:\mongo\input*. You need to create this directory.
   - **ArchiveReg.LISTENER_DESTINATION.** The directory to which the XML files are moved after being processed. For example, *c:\mongo\destination*. You need to create this directory.
   - **ArchiveReg.DB_HOST.** The name of the machine that hosts mongo. For example, *localhost*.
   - **ArchiveReg.DB_PORT.** The port number of the host machine that the mongo instance is listening on. For example, the default value is*27017*.
   - **ArchiveReg.DB_NAME.** The name of the database that stores the XML data. The database is created for you. For example, *FedReg*.
   - **ArchiveReg.COLLECTION_NAME.** The name of the collection that stores the XML data. This is analogous to a table in an RDBMS system. For example, *xml*.

- **ArchiveReg.TEST_CLIENT_LOCATION.** The location of the sample user interface. For example, set it to *_sreg(iwayworkdir)/resource/client*.

Neither id or query should exist. You do not need to set them.

8. Start the iWayXMLArchive configuration by navigating to your*iwayhome* directory and typing the following at a command prompt:

```
iway7.cmd iWayXMLArchive
```

Verifying the iWay XML Archive Installation

To verify the installation:

1. Download the following ZIP file, which contains large XML documents from the Federal Register:

   http://www.gpo.gov/fdsys/bulkdata/FR/2010/12/FR-2010-12.zip

2. Extract the contents to the c:\mongo\input directory. The listener starts to pick up each file and process it.
3. Start a web browser, and point to http://localhost:2222.
4. Click *Search*, and verify that there are 21 results returned.

Chapter 2

Using the iWay XML Archive

The iWay XML Archive is an XML document storage system that you can use to efficiently store, query, and retrieve a large number of XML documents.

In this chapter:

iWay XML Archive Architecture

For efficient storage and retrieval, the XML documents are first converted to BSON (Binary JSON) and stored in a mongoDB collection. A reference to the original XML document itself is then attached to the mongoDB document, and then stored using the mongoDB GridFS system.

You can then query and retrieve documents using web services hosted in iSM. The queries are Java Script Object Notation (JSON) like and hierarchical. They return either the JSON representation or the full XML document. In addition to retrieving a full XML document, you can apply an XQuery to a document and retrieve a part of it.

XML and JSON

The conversion to JSON from XML changes the structure of the document. Take this into account when making hierarchical queries. The primary structural change is that attributes become elements in JSON, and the content of a mixed element is set in a content key.

For example, consider the following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>testA</b>
      <c>12345</c>
   </a>
```

After the conversion, the following corresponding JSON document is created:

```
{
    "a":{
        "b":"testA",
        "c":"12345",
        "name":"bob"
    }
}
```

Another example in which the <a> element has content is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>testA</b>
      <c>12345</c>
      Test
   </a>
```

After the conversion, the following corresponding JSON document is created:

```
{
    "a":{
        "content":"\nTest",
        "b":"testA",
        "c":"12345",
        "name":"bob"
    }
}
```

Loading Documents

In the default configuration, documents are loaded using a file listener listening to a folder in the local file system. An XML document of any schema can be dropped into this folder and loaded into the archive.

For details on how to configure this location, see *Configuring the iWay XML Archive Application* on page 6.

Making Requests

The iWay XML Archive handles requests through an HTTP listener on port 2222 in the default configuration. The requests are made using a properly encoded URL of the following format:

```
http://host:port/?querystring
```

where:

`host`
Is the name of the machine on which the iWay XML Archive is installed and configured. The default value is localhost.

`port`
Is the number of the port that the iWay XML Archive is listening on. The default value is 2222.

`querystring`
Is the query.

The sample queries are encoded and can be executed from any web browser by cutting and pasting them into the address bar. The results are then displayed in the browser. If you are using Internet Explorer® or Mozilla Firefox, you will be asked to either save or open the resulting document. You can then use a text editor to view the results.

The Query String

The query string contains variables that are used to specify query arguments or indicate the type of query. The primary query variable is the word *query*. All queries use a JSON-type structure.

A basic example that returns the _id key value pairs for every record in the database is:

```
http://localhost:2222/?query={},{%22_id%22:1}
```

If you cut and paste this URL into the Google Chrome browser, you will see a result similar to the following. The number of results varies, depending on the number of records in your data store.

```
{ "retval" : [ { "_id" : { "$oid" : "4d068547e818f7cba1b63688"}} ,
 { "_id" : { "$oid" : "4d06852ee818f7cb95b63688"}} ,
 { "_id" : { "$oid" : "4d068532e818f7cb97b63688"}} ,
 { "_id" : { "$oid" : "4d068535e818f7cb98b63688"}} ,
 { "_id" : { "$oid" : "4d068545e818f7cba0b63688"}} ,
 { "_id" : { "$oid" : "4d068550e818f7cba6b63688"}}] ,
 "ok" : 1.0}
```

Adding the {"_id":1} limit to the query is a good practice if you want to retrieve the XML in a subsequent query. If this limit is omitted from the query, the full JSON representation will be returned for each result. If the records are large, this could significantly decrease performance.

Query Grammar

The contents of a query are a set of key-value pairs. You can join multiple pairs together to form *and* or *or*type queries. Furthermore, you can also use regular expressions for the value part of a pair.

Searching Elements

Consider the following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
   <a>
      <b>test</b>
      <c>12345</c>
   </a>
```

To find all document ids for which the value of <b> is equal to *test*, use the search string {"a.b":"test"} ,{"_id":1}. The HTTP request is as follows:

```
http://localhost:2222/?query={%22a.b%22:%22test%22},{%22_id%22:1}
```

Searching Attributes

In the iWay XML Archive, XML documents are first converted to JSON before they are loaded into the archive. This notation is used for storage and retrieval efficiency. Changes that convert attributes to child elements are made to the XML structure to support this notation.

Example: Attribute Search

Consider the following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>test</b>
      <c>12345</c>
   </a>
```

To find all document ids for which the value of the name attribute is equal to *bob*, use the search string {"a.name":"bob"} ,{"_id":1}. The HTTP request is as follows:

```
http://localhost:2222/?query={%22a.name%22:%22bob%22},{%22_id%22:1}
```

Searching on Multiple Keys (and)

To search for a document based on multiple keys, type comma-delimited, key-value pairs in the search box. This is similar to an *and* in a SQL where clause.

Example: Multiple Key Search

Consider the following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>test</b>
      <c>12345</c>
   </a>
```

To find all documents in which the name attribute of <a> is equal to *bob* and the value of <b> is equal to*test*, type {"a.name":"bob","a.b":"test"},{"_id":1}. The HTTP request is as follows:

```
http://localhost:2222/
?query={%22a.name%22:%22bob%22,%22a.b%22:%22test%22},{%22_id%22:1}
```

Advanced Queries

The iWay XML Archive query language is based on the mongoDB query language. Many of the advanced queries are supported. Some of the advanced queries require extra operators to complete, such as the *or* query. For more information on the mongoDB advanced queries, see *http://www.mongodb.org/display/DOCS/Advanced+Queries*

Example: Advanced Query

Consider the following XML documents:

```
<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>test</b>
      <c>12345</c>
   </a>

<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>testA</b>
      <c>12345</c>
   </a>
```

To find all document ids for which the value of <b> is equal to *test* or *testA*, use the following syntax:

```
{$or:[{"a.b":"test"},{"a.b":"testA"}]} ,{"_id":1}
```

The HTTP request is as follows:

```
http://localhost:2222/
?query={$or:[{%22a.b%22:%22test%22},{%22a.b%22:%22testA%22}]},
{%22_id%22:1}
```

Using Regular Expressions

You can use Pearl Compatible Regular Expressions (PCRE) for the value of a search expression. The expressions are not enclosed in quotation marks, and are bound between / (slash) characters. For more information on PCRE, see *http://www.cs.tut.fi/~jkorpela/perl/regexp.html*

Example: Regular Expression

Consider the following XML documents:

```
<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>test</b>
      <c>12345</c>
   </a>

<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>testA</b>
      <c>12345</c>
   </a>
```

To find all document ids for which the value of <b> contains the string *test*, type the following syntax

```
{"a.b":/test./},{"_id":1}
```

The HTTP request is as follows:

```
http://localhost:2222/?query={%22a.b%22:/test./},{%22_id%22:1}
```

Searching by _id

The previous examples limited the results of a search to return on the _id key-value pair. If the limitation is removed, the entire JSON document is retrieved for each result. If you search for a single id, you the entire JSON document for that id is retrieved. To execute a search by id, you must use the mongoDB ObjectId class in your query.

For example, to return a document with the id "4d068547e818f7cba1b63688", type the following search term:

```
{"_id":new ObjectId("4d068547e818f7cba1b63688")}
```

The HTTP request is as follows:

```
http://localhost:2222/
?query={%22_id%22:new%20ObjectId(%224d49af7ce5397929bad5421e%22)}
```

Understanding the JSON Results

The following document is returned from a search for a particular id.

```
{
    "retval":[
        {
            "_id":{
                "$oid":"4d49af7ce5397929bad5421e"
            }
            ,
            "a":{
                "b":"test",
                "c":"12345",
                "name":"bob"
            }
            ,
            "xml":{
                "_id":{
                    "$oid":"4d49af7ce5397929b8d5421e"
                }
                ,
                "chunkSize":262144,
                "length":80,
                "md5":"2fd2fece0f2c73ef073436d3e2638922",
                "filename":null,
                "contentType":"application/xml",
                "uploadDate":{
                    "$date":"2011-02-02T19:24:44Z"
                }
                ,
                "aliases":null
            }
        }
    ]
    ,
    "ok":1.0
}
```

The results of a search against the mongoDB are all returned in JSON format and are stored in the retval array. This array will contain multiple result objects if the search was not for a single id.

The JSON result also contains two additional keys. The first is the _id key, which is the unique id assigned by mongoDB to the record. The second is the xml key. The xml key stores reference information in the XML file in the GridFS system.

Returning XML Documents

Once the set of documents of interest has been identified, you can use the _id value to retrieve the XML document associated with the record. The query uses the query variables *query* and *xml*. For example,

```
http://localhost:2222/?query={"_id":new
ObjectId("4d484fcfe7bd792987408a76")}&xml=true
```

The xml=true switch causes the archive to first look up the location of the XML document using the value of _id. It then retrieves the document from the file system.

Returning a Slice of XML With XQuery

Instead of retrieving the full XML document, you can apply an XQuery and retrieve the resulting partial document. To execute an XQuery, you must set the *xquery* and *xml query* variables.

Consider this XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>test</b>
      <c>12345</c>
   </a>
```

The following query,

```
?query={"_id":new ObjectId("4d484fcfe7bd792987408a76"}&xquery=for
$x in /a return $x/b&xml=true
```

returns the following:

```
<?xml version="1.0" encoding="UTF-8" ?>
<XQueryResult>
   <item>
      <b>test</b>
   </item>
</XQueryResult>
```

The HTTP request is as follows:

```
http://localhost:2222/
?query={%22_id%22:new%20ObjectId(%224d49af7ce5397929bad5421e%22)}&xquery=
for%20$x%20in%20/a%20return%20$x/b&xml=true
```

For more information on XQuery, see the XQuery agent documentation.

Chapter 3

Using the iWay XML Archive Test Client

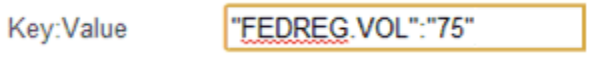
Queries can be made against the iWay XML Archive using an HTTP interface that wraps the mongoDB query language.

Application of an XQuery against a document retrieved in a search is also supported, using some additional options in the HTTP request.

In this chapter:

Test Client Overview

The iWay XML Archive Test Client is a web-based AJAX application used to search the iWay XML Archive and retrieve XML documents. The client interface wraps a mongoDB db.*collection*.find() method. The *Key:Value* field becomes the contents of the find method. Consider the following example, shown in the image below.



This example results in a mongoDB query of db.xml.find({"FEDREG.VOL":"75"}). The additional query options for field selection, sorting, skip, limit, and cursors, are not supported in the HTTP request sent from the Test Client. However, these types of queries can be made from a more intelligent client.

The client also supports applying an XQuery to the set of documents retrieved through the search. This is done by checking the *Apply XQuery:* check box and typing a query string in the *XQuery to apply:* input box as shown in the following image:

The client is an example of one of the many ways in which you can use the archive. In the default installation of the iWay XML Archive, the Test Client can be found at http://localhost:2222.

This section describes the use of the iWay XML Archive Test Client and some basics of the query grammar.

You can find more information about the mongoDB query language at:

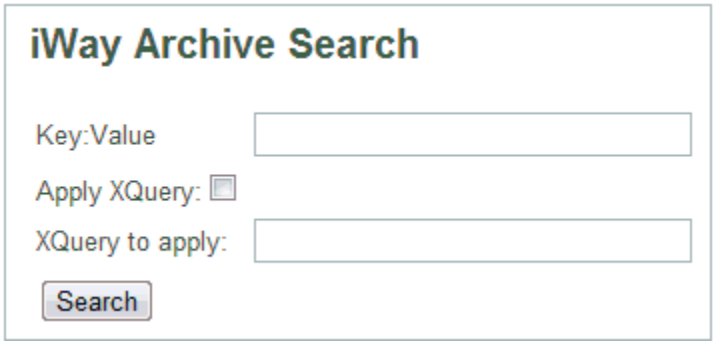
http://www.mongodb.org/display/DOCS/Querying

Query Grammar in the Test Client

The following topics provide the basics of query grammar used in the Test Client.

Searching Elements in the Test Client

The following image shows the iWay Archive Search query page.



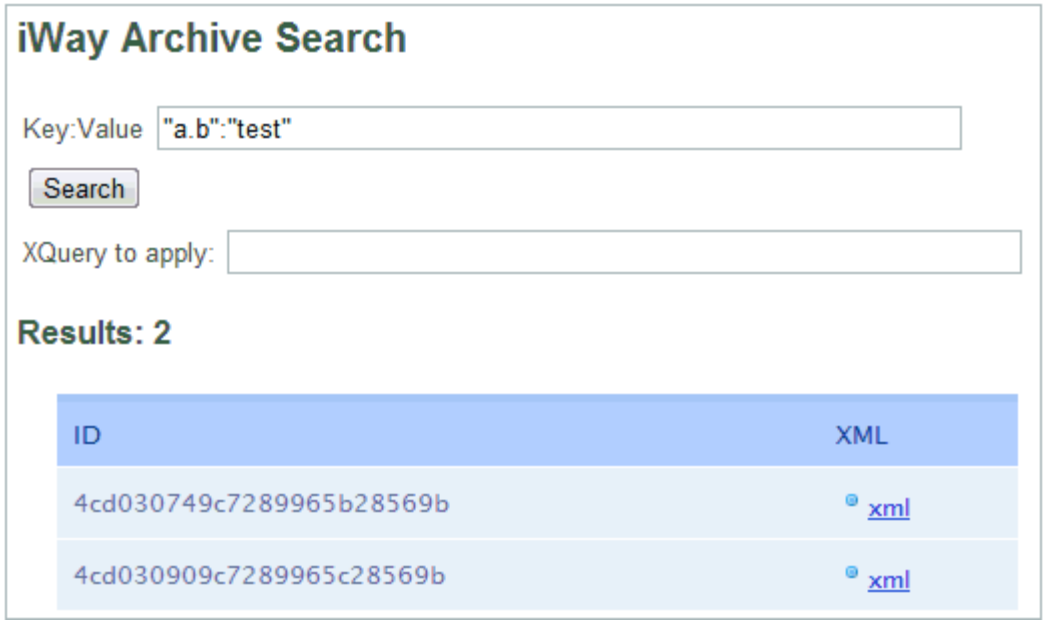
Specify the query in the *Key:Value* search box to search the archive. The Key portion of the query consists of the hierarchical element structure that you wish to query. The Key must be surrounded by double quotation marks. The Value portion consists of the value that the element must match. If the value is a string, it must be surrounded by double quotation marks. If the value is numeric, do not use quotation marks.

Example: Element Search

Consider the following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
   <a>
      <b>test</b>
      <c>12345</c>
   </a>
```

Typing "a.b":"test" retrieves all documents whose <b> element is equal to the string "test". The following image shows the results in the Test Client.



Searching Attributes in the Test Client

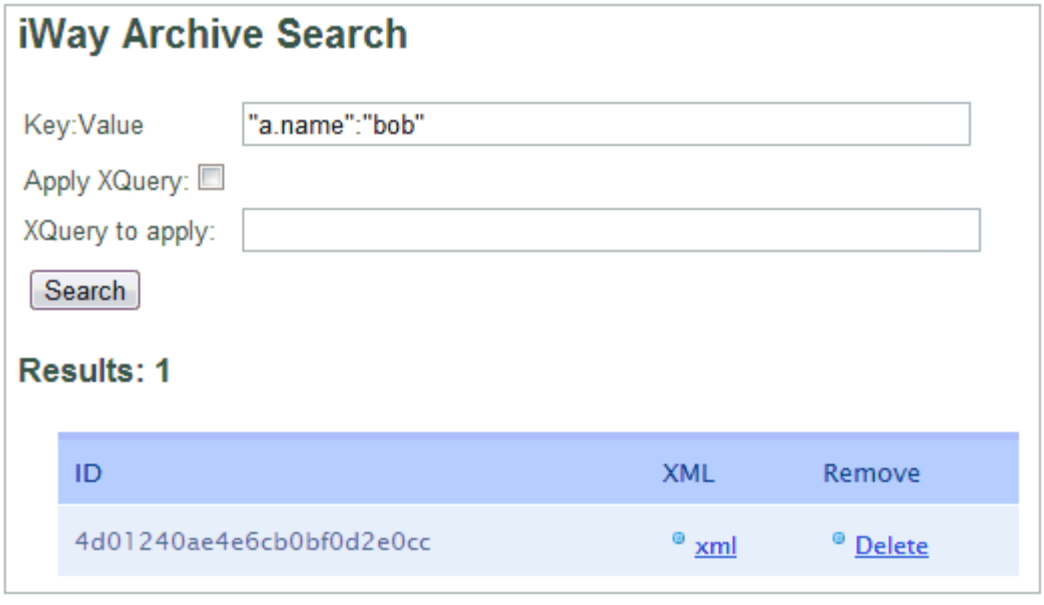
In the iWay XML Archive, XML documents are first converted to JSON (Java Script Object Notation) before they are loaded into the archive. This notation is used for storage and retrieval efficiency. Changes that convert attributes to child elements are made to the XML structure to support this notation.

Example: **Attribute Search**

Consider the following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>test</b>
      <c>12345</c>
   </a>
```

Typing "a.name":"bob" retrieves all documents containing the name attribute of the <a> element that equals the string *bob*. The following image shows the results in the Test Client.



Searching on Multiple Keys (and) in the Test Client

To search for a document based on multiple keys, type comma-delimited, key-value pairs in the *Key:Value* search box. This is similar to an *and* in a SQL where clause.

Example: Multiple Key Search

Consider the following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>test</b>
      <c>12345</c>
   </a>
```

Typing the search term "a.name":"bob","a.b":"test" returns all documents containing the name attribute of the <a> element that equals the string *bob*, and whose <a><b> element value is equal to the string *test*. The following image shows the results in the Test Client.

Advanced Queries in the Test Client

Some queries require extra operators, such as an *or* query. Some of these advanced queries are documented at:

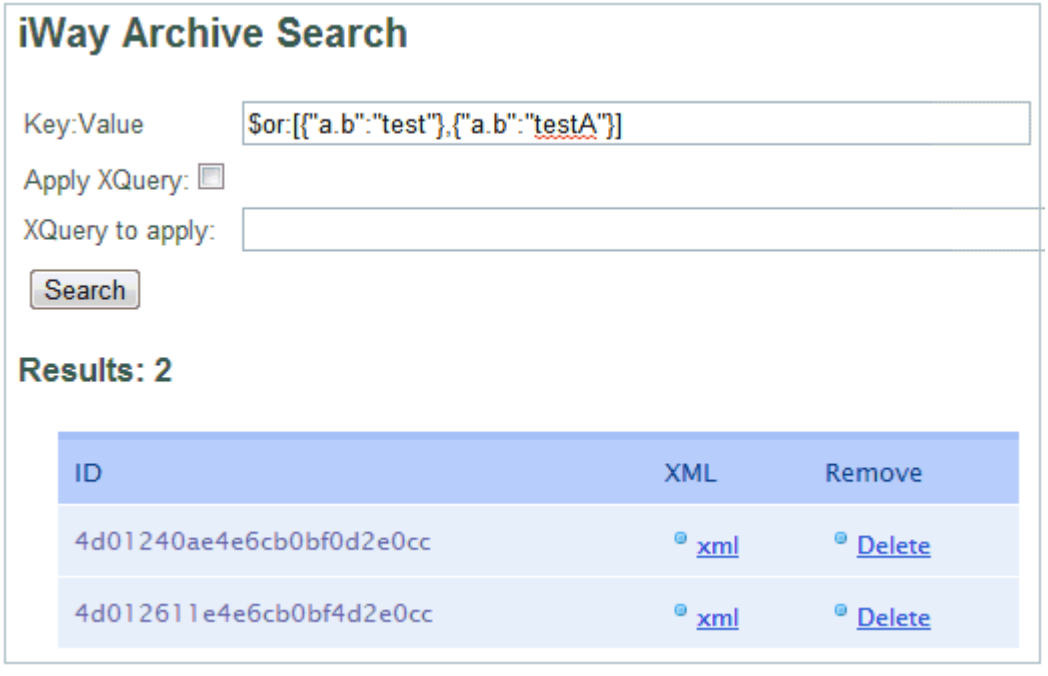
http://www.mongodb.org/display/DOCS/Advanced+Queries

Example: Advanced Query

Consider the following XML documents:

```
<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>test</b>
      <c>12345</c>
   </a>


<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>testA</b>
      <c>12345</c>
   </a>
```

Typing the search term $or:[{"a.b":"test"},{"a.b":"testA"}] results in the return of both documents, as shown in the following image. You can see that the $or key operator has the value of a JSON array with two JSON-formatted, Key:Value pair objects.



Using Regular Expressions in the Test Client

You can use Pearl Compatible Regular Expressions (PCRE) for the value of a search expression. The expressions are not enclosed in quotation marks, and are bound between / (slash) characters. For more information on PCRE, see *http://www.cs.tut.fi/~jkorpela/perl/regexp.html*.

Example: Regular Expression

Consider the following XML documents:

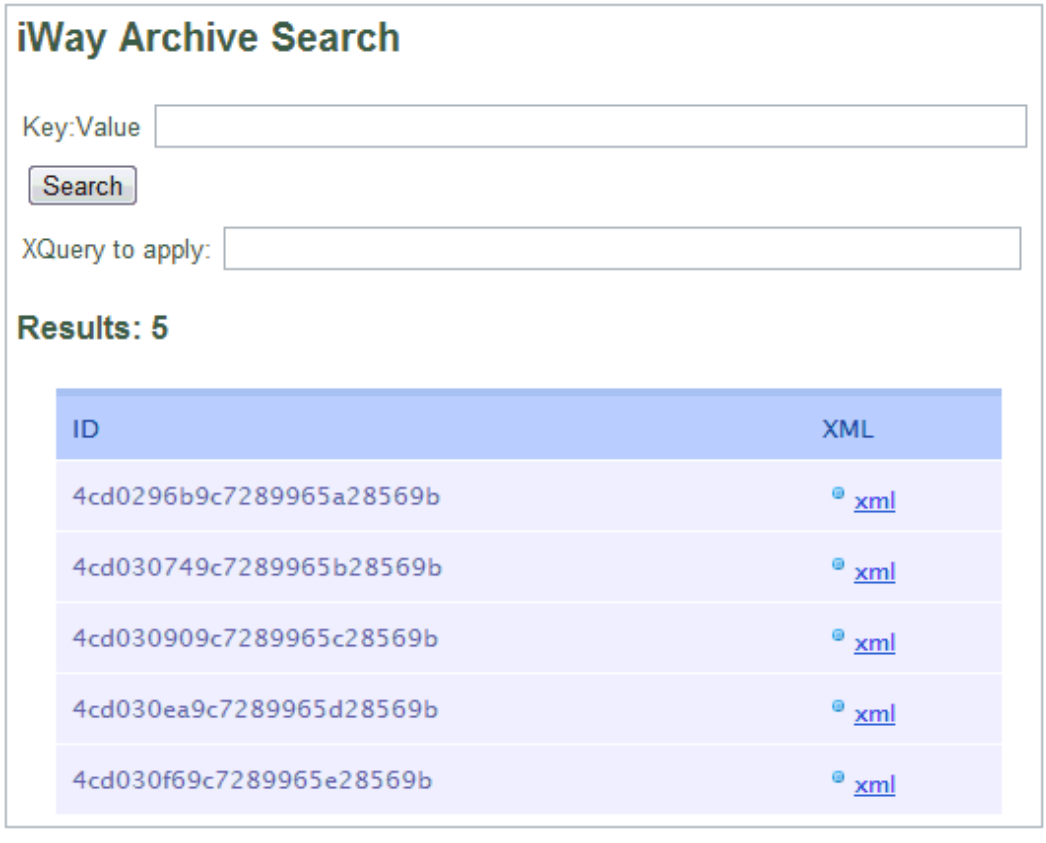
```
<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>test</b>
      <c>12345</c>
   </a>
```

```
<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>testA</b>
      <c>12345</c>
   </a>
```

Typing "a.b.":/test.*/ returns all documents whose <a><b> element values contain the string *test*, even if there is text following *test*. The period (.) and asterisk (*) operators match anything after the string *test*, 0 or more times.

Understanding the Results

The following image shows the results of an empty query, which returns all documents in the archive.



The ID column of the result set contains the key for the record returned. This key can be used in a query to retrieve one particular document. For more information, see *Special Queries* on page 27. The XML column contains a link that returns the full XML document for the record from the archive.

Applying an XQuery

You can apply an XQuery to a set of documents matching the mongoDB Key:Value query by selecting the *Apply XQuery* check box and typing a query in the *XQuery to apply:* field.

You can also apply an XQuery to an individual document in a result set by selecting the *Apply XQuery* check box, and then clicking the xml link in the result set.

Information on XQuery is available at:

http://www.w3schools.com/xquery/default.asp

Example: XQuery

Consider the following XML documents:

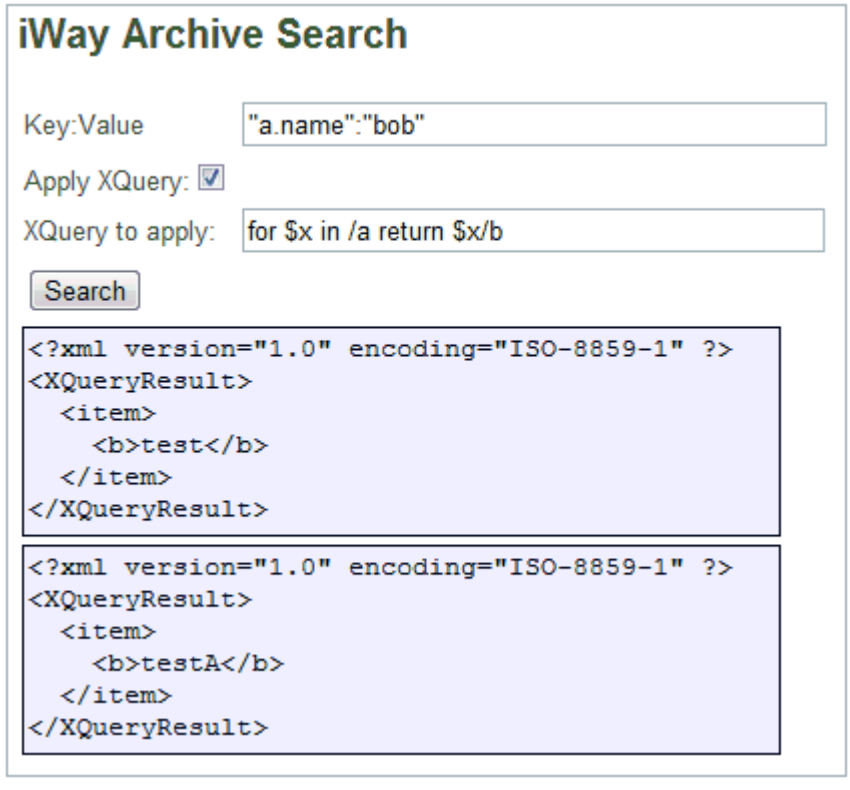
```
<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>test</b>
      <c>12345</c>
   </a>

<?xml version="1.0" encoding="UTF-8"?>
   <a name="bob">
      <b>testA</b>
      <c>12345</c>
   </a>
```

With a Key:Value query of "a.name":"bob" and an XQuery of *for $x in /a return $x/b*, two XML documents are returned. The returned documents have the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<XQueryResult>
   <item>Return xml</item>
</XQueryResult>
```

The results are wrapped in an <XQueryResult><item> element, in case the query has multiple items, and the root has been removed.

An example of the query performed in the Test Client is shown in the following image.

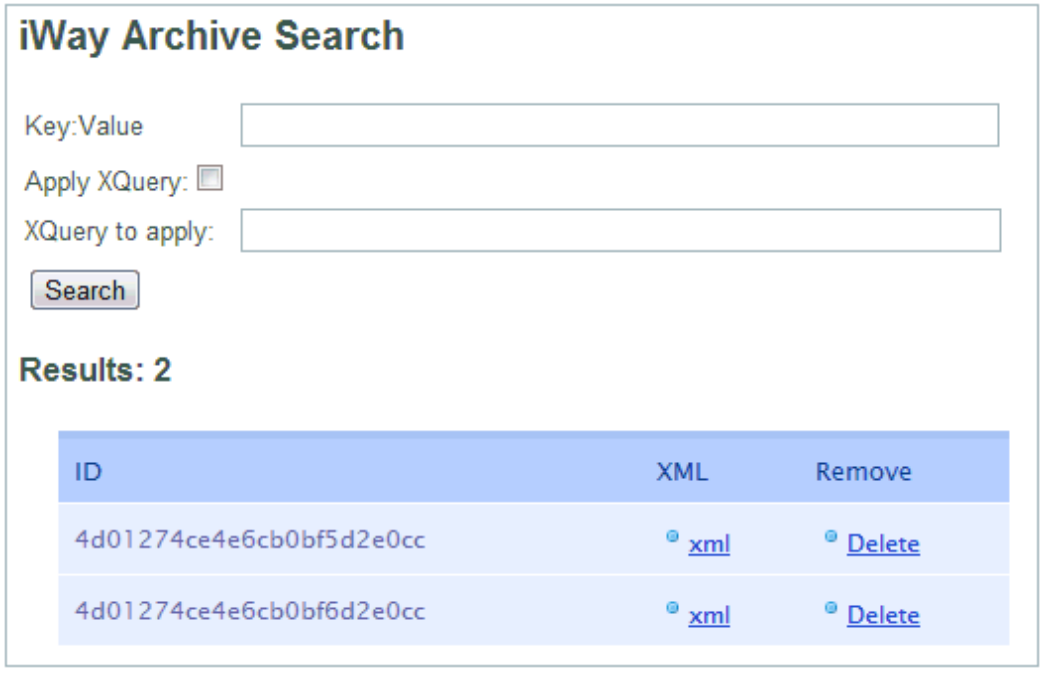


Special Queries

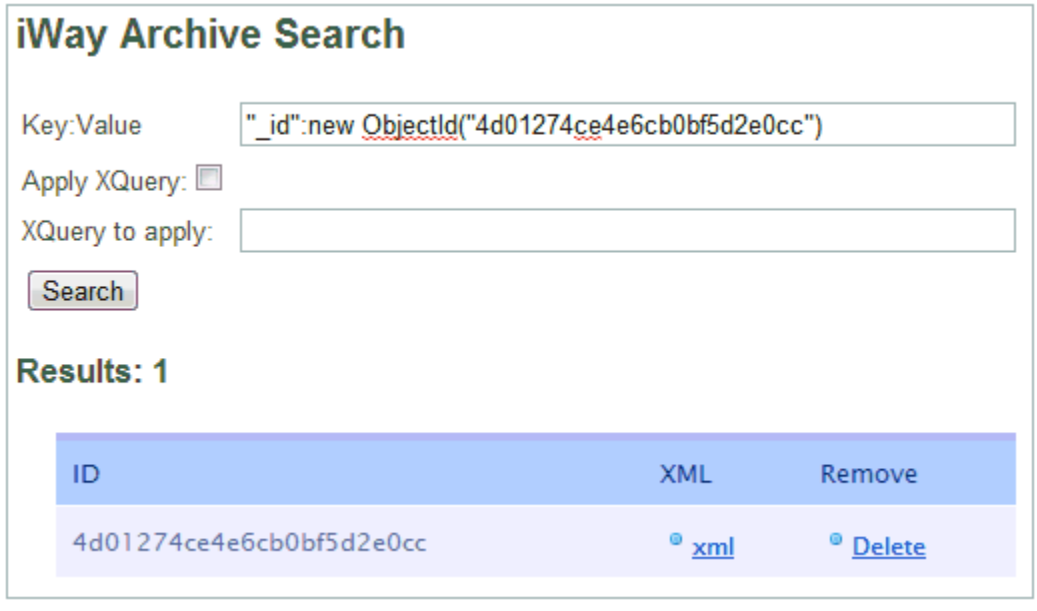
The archive query language uses a special notation for querying ID or date fields. Date fields are a special case and must be defined to the archive prior to loading.

Querying an ID

The following image shows a search result of two records.



Given the preceding search results, to retrieve just the third record in the list, type "_id":new ObjectId("4d01274ce4e6cb0bf5d2e0cc") in the Key:Value field. The following image shows the results.



SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

TIBCO, the TIBCO logo, the TIBCO O logo, FOCUS, iWay, Omni-Gen, Omni-HealthData, and WebFOCUS are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle Corporation and/or its affiliates.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the readme file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.