

---

# **Mark Logic Content Interaction Server**

---

## **Getting Started with the XDBC Java API**

Version 2.2  
September 1, 2004

---

---

## Table of Contents

---

---

### Getting Started with the XDBC Java API

1.0	Introduction .....	3
1.1	Objectives .....	3
1.2	Audience .....	3
1.3	Scope .....	3
1.4	Requirements .....	3
2.0	Key Concepts .....	4
2.1	Why XDBC? .....	4
2.2	Basic Architecture .....	4
2.3	Configuring an XDBC Server .....	5
3.0	Using the XDBC Java API .....	6
3.1	Platform Support .....	6
3.2	Installation Instructions .....	6
3.3	XDBC Java API Development Kit .....	7
3.4	Building XDBC Java Applications .....	7
4.0	Sample Applications .....	8
4.1	XDMP SAXClient .....	9
4.2	XDMP DOMClient .....	10
4.3	XDMP DocLoader .....	11
4.4	XDMP DirLoader .....	12
4.5	XEval .....	13
5.0	Technical Support .....	15

## 1.0 Introduction

Content Interaction Server is a powerful software solution for harnessing your digital content base. Content Interaction Server enables you to build complex applications that interact with large volumes of XML, SGML, HTML and other popular content formats.

Mark Logic's XDBC Java API enables you to write Java applications that communicate with Mark Logic's Content Interaction Server. With the XDBC Java API, Content Interaction Server can be integrated into application server environments and other complex multi-tier systems.

### 1.1 Objectives

This document introduces key concepts embodied in the XDBC Java API and provides an overview of the XDBC Java API Development Kit.

### 1.2 Audience

This installation guide is intended for a technical audience, specifically an IT staff with experience in installing Java class libraries and in developing Java applications or Java servlets.

### 1.3 Scope

This installation guide provides an introduction to the XDBC Java API Development Kit. This document does not explain how to install the class libraries or how to install the application server with which your Java servlets will communicate.

To see the detailed API documentation or to learn how to configure XDBC servers in Content Interaction Server, refer to the appropriate documents:

- Javadoc methods documentation contained in XDBC Java API Development Kit
- Content Interaction Server *Administrator's Guide*

### 1.4 Requirements

Before using the XDBC Java API, be sure that your system meets the following requirements:

- Java J2SE 1.4.2 installed.

If you want to develop and run Java servlets, make sure an appropriate J2EE application server is installed and running.

**Note:** If you want to use XEval, our sample Java servlet-based web application, you need the JDK (J2EE) version of the Java runtime.

## 2.0 Key Concepts

The objective of Mark Logic's XDBC Java API is to enable enterprises to integrate Content Interaction Server into their existing computing infrastructure and deploy complex multi-tier applications that use Content Interaction Server as the underlying content repository.

### 2.1 Why XDBC?

Mark Logic's products have a strong open standards orientation, conforming to the W3C XML, XPath, XML Schema, and XQuery standards. In embracing Java, Mark Logic's goal is to continue our open standards approach.

Although JDBC is a widely accepted API through which Java applications communicate with relational database systems, the JDBC API is built on a conceptual foundation that closely hews to the relational model. The core aspects of the JDBC API reflect the key relational concepts of tables, rows and fields – concepts that are not applicable to the XML document model supported by Mark Logic's XML Data Management Platform. These fundamental differences make JDBC an inappropriate interface to Mark Logic's products.

Consequently, Mark Logic has decided to develop XDBC, a Java API better-suited to the XML document model. In order to ease the learning curve for developers, we have closely modeled the XDBC API on the existing JDBC standard. We adopted many of the JDBC interfaces without any changes, and where appropriate, transformed the more relationally-oriented interfaces into a more XML-centric approach.

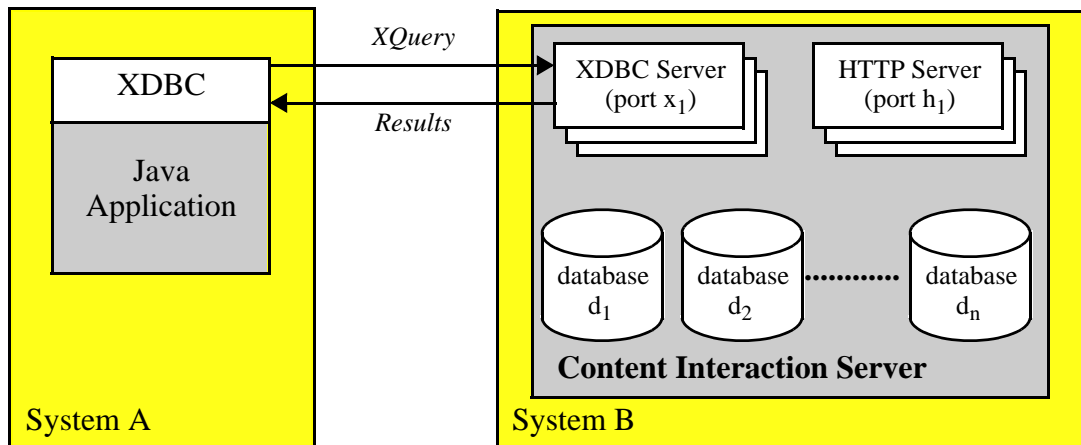
### 2.2 Basic Architecture

The XDBC API is provided as a set of class libraries that can be used by your Java code. Applications can be written as standalone Java programs or as Java servlets that are executed by your application server.

Your XDBC-enabled application connects to a specified port on a system that is running Content Interaction Server, and communicates with by submitting XQuery programs and processing the results returned by those programs. These XQuery programs can incorporate calls to XQuery functions stored at the server, and accessible from any XDBC-enabled application. The XQuery programs can perform the full suite of XQuery functionality, including loading, querying, updating and deleting content from the repository.

XQuery requests submitted via XDBC return results as specified by the XQuery code. These results can include XML and a variety of other datatypes. It is the Java application's responsibility to parse, process and interpret these results in a manner appropriate the variety of datatypes available. There are a number of publicly available libraries for assisting with this task, or you may use your proprietary code. In order to accept connections from XDBC-enabled Java applications, must have been configured with an XDBC Server listening on the designated port. Each XDBC Server connects to a specific database within Content Interaction Server.

The following figure illustrates the high-level architecture:



As shown in the diagram above, the XDBC-enabled application can run on the same system as an instance of Content Interaction Server (a host), or it can run on a completely different system, as long as the two systems are networked together.

In the diagram, the Java application running on System A has opened an XDBC connection to port  $x_1$  on System B. On System B, Content Interaction Server has been configured with an XDBC Server listening to port  $x_1$ , and that XDBC Server has been connected to database  $d_1$ . Consequently, the configuration shown in the diagram above allows the Java application on System A to submit XQuery requests (including query, load, update and delete) for evaluation against database  $d_1$ .

### 2.3 Configuring an XDBC Server

You use the Admin console to set up an XDBC server, specifying a name, port, a database to access, and other configuration parameters. For detailed instructions how to configure an XDBC Server, see the *Administrator's Guide*.

## 3.0 Using the XDBC Java API

The XDBC Java API is available by downloading the XDBC Java API Development Kit from Mark Logic's support website. For detailed instructions, contact [support@marklogic.com](mailto:support@marklogic.com).

### 3.1 Platform Support

The XDBC Java API is supported on the following platforms:

- Microsoft Windows 2000 SP3
- Sun Solaris 8 and 9
- Red Hat Enterprise Linux 2.1 and 3.0

The XDBC driver code is Java-based and should be portable to other platforms where Java is supported.

### 3.2 Installation Instructions

If you have previously installed the XDBC Java API Development Kit on the machine, you may want to uninstall the old version before proceeding with the installation.

The XDBC Java API Development Kit comes as a Zip file. You should unzip the development kit on the platform you will use to develop your Java applications.

On Linux and Solaris systems, issue the following command:

```
% unzip MarkXDBC-2.1.zip
```

On Windows systems, double-click on the Zip file to launch your WinZip or similar application and use the application to extract the contents into a convenient location.

### 3.3 XDBC Java API Development Kit

The development kit expands into a directory structure with the following components:

File or Directory	Command
README.txt	Release notes and other relevant technical information of interest to developers.
classes/	Source code for selected sample XDBC applications included in the development kit.
docs/	Javadoc (HTML-based) documentation of class libraries.
lib/	.jar files used for building XDBC Java client applications.
xeval/	Sample servlet-based XQuery evaluator along with source code.
conf/	Logging configuration files.

You should read the `README.txt` file before proceeding to use any other part of the XDBC Java API Development Kit.

### 3.4 Building XDBC Java Applications

To start building Java applications that use the XDBC Java API, follow these steps:

Experiment with the sample applications and review the relevant source code included with the development kit. The sample applications are introduced in the next chapter.

Understand the interfaces, classes and methods by reviewing the XDBC Javadoc documentation.

Before compiling an XDBC Java client application or servlet, ensure that the following two .jar files are included in your CLASSPATH environment variable at both compile-time and run-time:

.jar Files	Description
xdbc.jar	The XDBC interface definitions.
xdmp.jar	The XDMP implementation of the interfaces.

Compile your XDBC Java client application or servlet using your favorite Java environment.

## 4.0 Sample Applications

The XDBC Java API Development Kit contains a number of sample applications. The sample applications are as follows:

Sample	Type	Description
XDMPSAXClient	Application	A sample Java client that evaluates an XQuery expression and parses the results using a SAX parser.
XDMPDOMClient	Application	A sample Java client that evaluates an XQuery expression and creates a DOM tree.
XDMPDocLoader	Application	A sample Java client that loads a single XML file into Content Interaction Server
XDMPDirLoader	Application	A sample Java client that loads a directory of XML files into Content Interaction Server.
XEval	Servlet	A Java servlet-based web application that allows XQuery statements to be evaluated through a web browser interface

Each sample application is provided along with its source code, giving developers a starting point for authoring their own applications.

Be sure to complete the following steps before running the sample applications:

1. Create and configure an XDBC Server using the Admin interface. See the Content Interaction Server *Administrator's Guide* for details on how to create and configure an XDBC Server.
2. Configure a user with for the XDBC Server you created. Add a user to the security database with the username as `user` and the password as `pass`. The sample applications are hardwired to use that username and password. See the Content Interaction Server *Administrator's Guide* for details on adding a user to the security database.

The sample servlet, XEval, will need to be deployed into your specific application server environment before it can be used. Instructions for deploying the application are provided in the section “XEval” on page 13.

## 4.1 XDMP SAX Client

`XDMP SAX Client` is a sample Java client that evaluates an XQuery expression and parses the results using a SAX parser. To execute the application, issue the following command:

```
% java com.marklogic.xdmp.util.XDMP SAX Client <host> <port> <XQuery
expression>
```

For example, the following command:

```
% java com.marklogic.xdmp.util.XDMP SAX Client marklogicserver 8002
//profile[first_name="David"][1]
```

will connect to the XDBC server listening on port 8002 of the system named `marklogicserver`, and return the first `profile` node containing a `first_name` element with value “David”. The returned node will be parsed using a SAX parser and printed to standard output.

**Note:** Consider the following:

- On UNIX, you can escape the entire XQuery expression by surrounding it with single-quotes(') or you may escape special characters individually by using backslashes. For example, the entire XQuery expression above should be surrounded with single quotes to avoid the [ ] characters being interpreted as single expression characters
- On Windows, you may need to surround the XQuery expression with double quotes if it contains spaces.

In UNIX-based command line environments, you can use the back-quote (`) command-line escape to send a more complex XQuery expression that is stored in a file. The following example submits the XQuery described in the file `test.xqy` to the XDBC server listening on port 8002 of the system named `marklogicserver`:

```
% java com.marklogic.xdmp.util.XDMP SAX Client marklogicserver 8002 `cat
test.xqy`
```

To use a different SAX library other than the default, you need to specify an additional system property when running your Java client:

```
% java -Djavax.xml.parsers.SAXParserFactory=<SAX Parser Factory
implementation class>
com.marklogic.xdmp.util.XDMP SAX Client <host> <port> <XQuery
expression>
```

For example, to use the saxon implementation for the profile query shown above:

```
% java -Djavax.xml.parsers.SAXParserFactory=net.sf.saxon.aelfred.SAXParserFactoryImpl
com.marklogic.xdmp.util.XDMPsAXClient marklogicserver 8002
//profile[first_name="David"] [1]
```

Make sure that the relevant .jar files are in your CLASSPATH environment variable before issuing this command.

## 4.2 XDMPDOMClient

`XDMPDOMClient` is a sample Java client that evaluates an XQuery expression and returns the results as a DOM tree that is printed to standard output. To execute the application, issue the following command:

```
% java com.marklogic.xdmp.util.XDMPDOMClient <host> <port> <XQuery
expression>
```

For example, the following command:

```
% java com.marklogic.xdmp.util.XDMPDOMClient marklogicserver 8002
//profile[first_name="David"] [1]
```

will connect to the XDBC server listening on port 8002 of the system named `marklogicserver`, and return the first `profile` node containing a `first_name` element with value “David”. The node will be returned as a DOM tree and printed to standard output.

**Note:** Consider the following:

- On UNIX, you can escape the entire XQuery expression by surrounding it with single-quotes(‘) or you may escape special characters individually by using backslashes. For example, the entire XQuery expression above should be surrounded with single quotes to avoid the [ ] characters being interpreted as single expression characters
- On Windows, you may need to surround the XQuery expression with double quotes if it contains spaces.

In UNIX-based command line environments, you can use the back-quote ( ` ) command-line escape to send a more complex XQuery expression that is stored in a file. The following example submits the XQuery described in the file `test.xqy` to the XDBC server listening on port 8002 of the system named `marklogicserver`:

```
% java com.marklogic.xdmp.util.XDMPDOMClient marklogicserver 8002
`cat test.xqy`
```

To use a different DOM library other than the default, you need to specify an additional system property when running your Java client:

```
% java -Djavax.xml.parsers.DocumentBuilderFactory=<Document Builder
Factory impl. class>
      com.marklogic.xdmp.util.XDMPDOMClient <host> <port> <XQuery
expression>
```

For example, to use Oracle's DOM implementation for the profile query shown above:

```
% java -Djavax.xml.parsers.DocumentBuilderFactory=oracle/xml/jaxp/
JXDDocumentBuilderFactory
      com.marklogic.xdmp.util.XDMPDOMClient marklogicserver 8002
      //profile[first_name="David"] [1]
```

Make sure that the relevant `.jar` files are in your `CLASSPATH` environment variable before issuing this command.

### 4.3 XDMPDocLoader

`XDMPDocLoader` is a sample Java client that loads a set of XML files into a Content Interaction Server. To execute the application, issue the following command:

```
% java com.marklogic.xdmp.util.XDMPDocLoader <host> <port> <listfile>
```

`<listfile>` is a file that contains a list of line-delimited document descriptors. Each line in `<listfile>` contains the name of an XML document to load and the URI to assign to that document when loading it into the forest.

For example, the following command:

```
% java com.marklogic.xdmp.util.XDMPDocLoader marklogicserver 8002
listOfXMLFiles
```

coupled with the following `listOfXMLFiles` file (which is expected to be located in the working directory):

```
docs/document1.xml doc1.xml
docs/document2.xml doc2.xml
backups/original.xml original.xml
```

will connect to the XDBC server listening on port 8002 of the system named `marklogicserver`, and load the files `docs/document1.xml`, `docs2/document2.xml` and `backups/original.xml` (all relative to the working directory) into the forest connected to that XDBC server, setting the URIs for each of them as `doc1.xml`, `doc2.xml`, and `original.xml`, respectively.

#### 4.4 XDMPDirLoader

`XDMPDirLoader` is a sample Java client that loads a directory of XML files into Content Interaction Server. To execute the application, issue the following command:

```
% java com.marklogic.xdmp.util.XDMPDirLoader <host> <port> <dirpath>
  [<file suffix>]
```

For example, the following command:

```
% java com.marklogic.xdmp.util.XDMPDirLoader marklogicserver 8002 .
xml
```

will connect to the XDBC server listening on port 8002 of the system named `marklogicserver`, and load every file with a name that matches `*.xml` from the current directory into the forest connected to that XDBC server. If the optional file suffix argument is omitted, all files are loaded from the specified directory, regardless of their filename. The URIs for the documents loaded will be the filenames (without path descriptors) of the files found in the directory.

**Note:** `XDMPDirLoader` only loads documents from the specified directory. It does not recursively descend through the directory structure loading XML documents from all of the subdirectories within the specific directory.

## 4.5 XEval

`xEval` is a sample Java servlet-based web application that allows XQuery statements to be evaluated through a web browser interface.

`xEval` has been tested under Tomcat 4.1 but should be portable to most other J2EE application servers. Tomcat 4.1 can be downloaded from <http://jakarta.apache.org/tomcat/index.html>.

A script for deploying `xEval` as a Java web application is provided. To deploy `xEval`, perform the following steps:

1. Open the `<xdbc_dir>/xeval/conf/web.xml` file in a text editor.
2. Find the `serverhost` and `serverport` variables and set them to the same XDBCServer host and port as specified in the Admin interface.
3. Make sure that the `username` and `password` variables are set to correct values for your configuration.
4. Save the changes to your `web.xml` file.
5. Open the `<xdbc_dir>/xeval/deploy.bat` (if you are using Windows) or `<xdbc_dir>/xeval/deploy.sh` (if you are using UNIX) file in a text editor.
6. Set the `DEFAULT_XDBC` variable to the directory in which you unzipped the Mark Logic XDBC release.
7. Set the `DEFAULT_APPSERV` variable to the location of your Tomcat server.

**Note:** On Windows, pathnames with spaces may need to be surrounded by double quotes in the deploy script. However, do not use double quotes to surround pathnames on UNIX systems.

8. Modify the Tomcat startup scripts by adding the following JAR files to the `CLASSPATH` set in the `TOMCAT_HOME/bin/setclasspath` script:
  - `<xdbc_dir>/lib/xeval.jar`
  - `<xdbc_dir>/lib/xdmp.jar`
  - `<xdbc_dir>/lib/xdbc.jar`
  - `TOMCAT_HOME/common/lib/servlet.jar`

where `xdbc_dir` is the directory in which you unzipped the Mark Logic XDBC release and `TOMCAT_HOME` is the directory in which Tomcat is installed.

For example, add a line similar to the following to the `TOMCAT_HOME/bin/setclasspath.bat` (`setclasspath.sh` on UNIX systems) file:

```
set CLASSPATH=%CLASSPATH%;c:\MarkXDBC-2.1\lib\xeval.jar;C:\MarkXDBC-2.1\lib\xdmp.jar;c:\MarkXDBC-2.1\lib\xdbc.jar;c:\tomcat\jakarta-tomcat-4.1.30\common\lib\servlet.jar
```

**Note:** Add this line after the other `set CLASSPATH` lines but before the `set _RUN*` lines in the `setclasspath.bat` or `.sh` file.

9. Change to the `xeval` subdirectory and issue the following command:

Windows:

```
deploy.bat
```

UNIX:

```
./deploy.sh
```

This will copy the `xEval` files into your Tomcat `webapps` directory.

10. Make sure your environment is set up to use the J2EE version of Java, not the J2SE version. You can set the `JAVA_HOME` variable in the `TOMCAT_HOME/bin/setclasspath.bat` or `setclasspath.sh` script (at the beginning of the script) as in the following examples:

Windows (`setclasspath.bat`):

```
set JAVA_HOME=<j2ee_install_dir>
```

UNIX (`setclasspath.sh`):

```
set JAVA_HOME=<j2ee_install_dir>
```

11. Restart Tomcat and open the `XEval` page in a browser:

```
http://localhost:8080/xeval/index.jsp
```

You can submit queries in the `XQuery` evaluator and the results will display in the bottom part of the browser.

**Note:** Consider the following about running the `xEval` Java application:

- Tomcat uses any of the following ports: 8005, 8007, 8008 and 8080. You may have a port conflict if another application uses any of the ports above.
- If you make changes to the `web.xml` file or the `xEval` code, you will need to run the deploy script again and then restart Tomcat.
- When you install Tomcat, make sure Tomcat is configured to use a full J2EE version of Java, not a J2SE version of Java.

## 5.0 Technical Support

Mark Logic provides technical support according to the terms detailed in your Software License Agreement. For evaluation licenses, Mark Logic may provide support on an “as possible” basis.

We invite you to visit our support website at <http://support.marklogic.com> to access our full suite of documentation and help materials.

If you have questions or comments, you may contact Mark Logic Technical Support at the following email address:

[support@marklogic.com](mailto:support@marklogic.com)

If reporting a query evaluation problem, please be sure to include the sample XQuery code.