XMLmind XSL Server - Manual

Hussein Shafie

XMLmind Software
35 rue Louis Leblanc
78120 Rambouillet
France
Phone: +33 (0)9 52 80 80 37
xfc-support@xmlmind.com
www.xmlmind.com/foconverter/

December 27, 2024

Table of Contents

Introduction ii

Part I. Deploying XMLmind XSL Server 1

Chapter 1. Installing XMLmind XML Server 2

Chapter 2. Deploying the WebApp on your existing Servlet Container 3

1. Server parameters 4

2. User authentication 7

Part II. Configuring XMLmind XSL Server 10

Chapter 3. Specifying how to convert an XML document to another format 11

Chapter 4. Customizing a stock conversion 14

Chapter 5. Adding a custom conversion 17

Part III. Using XMLmind XSL Server 20

Chapter 6. User interface (interactive requests) 21

Chapter 7. API (non-interactive requests) 24

Index i

Introduction

XMLmind XSL-FO Converter is available embedded in XMLmind XSL Server, a powerful, production-quality, Servlet which leverages the XSL technology to allow converting XML documents to a variety of formats.

Out of the box, it allows to convert DocBook 4.x, 5.0, 5.1 and 5.2 including assemblies and XHTML documents to PDF, RTF (can be opened in Word 2000+), WordprocessingML (can be opened in Word 2003+), Office Open XML (`.docx`, can be opened in Word 2007+) and OpenOffice (`.odt`, can be opened in OpenOffice/LibreOffice 2+) formats. DITA 1.0, 1.1, 1.2 and 1.3 documents can be converted to even more formats: XHTML 1.0, XHTML 1.1, HTML 4.01, XHTML 5, Web Help, HTML Help, Eclipse Help, EPUB 2, EPUB 3, PDF, RTF, WordprocessingML, Office Open XML, OpenDocument.

XMLmind XSL Server targets the Web developer (JavaScript, Ajax, PHP, etc) and the system integrator.

Part I. Deploying XMLmind XSL Server

Deploying XMLmind XSL Server involves:

- Installing its software distribution.
- Deploying the XMLmind XSL Server WebApp on a Servlet Container (Apache Tomcat, Caucho Resin, Eclipse Jetty, etc).

Chapter 1. Installing XMLmind XML Server

XMLmind XML Server is installed by unzipping the `.zip` archive containing its distribution anywhere you want. Doing this creates a `xslsrv_V_N_M/` directory having the following contents:

`doc/`
: Contains the documentation of XMLmind XML Server.

`legal/`
`legal.txt`
: Contains legal information (licenses, notices, etc) about XMLmind XSL Server and the software components used to build it.

`xslsrv.war`
: The `.war` file containing the XMLmind XSL Server WebApp.

`xslsrv/`
: The unpacked `xslsrv.war` (in case you want to rebuild `xslsrv.war` by running a command like: `jar cf xslsrv.war xslsrv`).

Chapter 2. Deploying the WebApp on your existing Servlet Container

By following the procedure below step by step, a member of your IT staff (not an end-user) should be able to easily deploy the XMLmind XSL Server WebApp on a Servlet Container.

Before you begin

Required software:

- Java™ runtime 1.8+ . Both Oracle Java and OpenJDK are officially supported.
- A Servlet Container compatible with the Servlet 2.3+ standard.

Procedure

1. Stop the Servlet Container.
2. Copy *`install_dir`*`/xslsrv.war`, the `.war` file containing the XMLmind XSL Server WebApp, to the WebApp deployment directory of your Servlet Container.

   Important

   About Apache Tomcat version 10 and above

   Beware that there is a *major breaking change* between latest versions of Apache Tomcat (>= 10) and older versions (<= 9). This is documented in this migration article.

   To make a long story short, if you need to deploy the "Word To XML" servlet on Tomcat version 10+, then you first must create a `webapps-javaee/` folder next to `TOMCAT_INSTALL_DIR/webapps/` then copy `xslsrv.war` to this `TOMCAT_INSTALL_DIR/webapps-javaee/`.

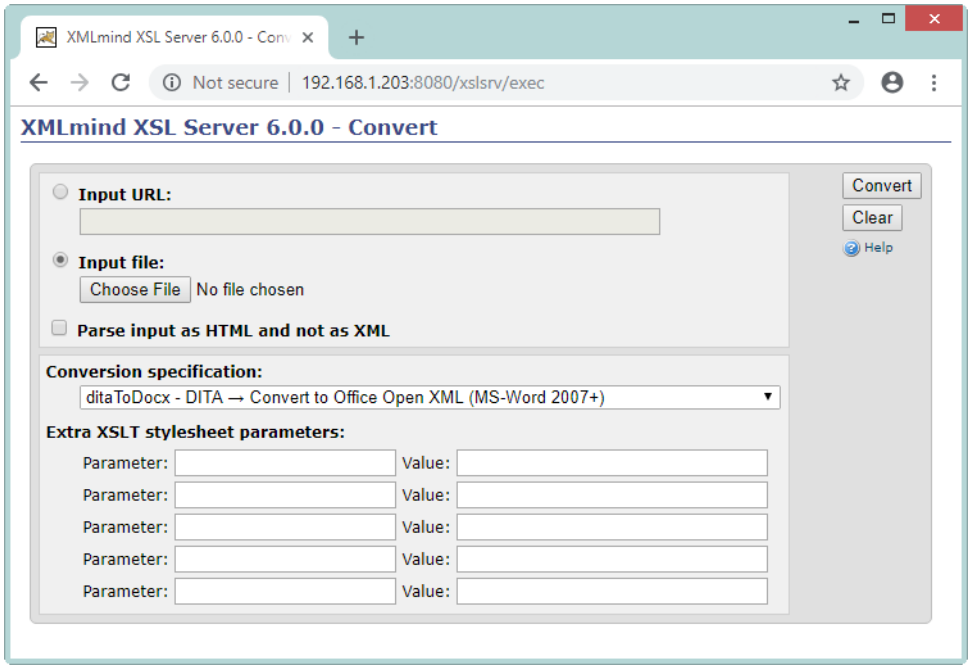
   Tomcat example, assuming that Tomcat has been installed in `/opt/tomcat/`:

   ```
   /# cp /opt/xslsrv/xslsrv.war /opt/tomcat/webapps
   ```

3. Restart the Servlet Container.
4. Test that XMLmind XSL Server is up and running by pointing your Web browser to the `xslsrv/exec` URL.

   For example, let's suppose that the URL of the Servlet Container is `http://localhost:8080/`. Type `http://localhost:8080/xslsrv/exec` in the address bar of your Web browser.

   You should be able to see the following interactive **Convert** form.



What to do next

After deploying XMLmind XSL Server as explained above and testing that it works fine, you may want to:

- Fine tune the behavior of XMLmind XML Server by changing the value of some its parameters.
- Turn user authentication on.

1. Server parameters

The following server parameters allow to fine tune the behavior of XMLmind XML Server. They are declared in `xslsrv/WEB-INF/web.xml` (`xslsrv/` being the unpacked `xslsrv.war`).

Important

Configuring XMLmind XML Server always involves modifying files found in `xslsrv.war`. This implies unpacking[(1)] `xslsrv.war`, modifying some configuration files and then repacking `xslsrv.war`. Fortunately, this is generally done once for all.

`authorization1`

A passphrase which must be passed by the client along all convert requests ("`convert`", "`poll`", "`cancel`", "`result`", "`config`").

Note that this feature is orthogonal to the user authentication. However it is seldom needed when user authentication has been turned on.

Default: none. Convert requests are honored without having to pass any authorization.

`authorization2`

A passphrase which must be passed by the client along all administration requests ("`jobs`", "`clear`").

(1) A `.war` file is just a ZIP archive having a "`.war`" suffix.

Note that this feature is orthogonal to the user authentication. However it is seldom needed when user authentication has been turned on.

Default: none. Administration requests are honored without having to pass any authorization.

`cacheSize`

Number of job results (that is, PDF, RTF, `.docx`, etc, files created by XMLmind XSL Server) to be cached.

Note that the result of converting an uploaded file is never cached. Only the result of converting an URL can be cached.

Value: Strictly positive integer.

Default: 100.

`credentials`

Accessing some URLs to be converted request user authentication. This parameter specify such user credentials.

Value: Zero or more prompt/username/password triplets separated by whitespace.

A "`*`" prompt matches any prompt. Fields containing whitespace must be quoted using single or double quotes. Example: "`'Document Repository' john foo * admin "bar gee"`".

Default: none. User credentials are not needed to access the URLs to be converted.

`customizeDir`

This directory is intended to contain one or more of the following specifications:

- Customizations of the stock conversion specifications of XMLmind XSL Server.
- User-defined conversion specifications.
- Apache FOP font specifications (e.g. map "`sans-serif`" to "`Verdana`").
- The locations of helper applications such as the HTMLHelp compiler (i.e. "`hhc.exe`").

This directory has the same layout and contents as the user preferences directory of XMLmind XSL Utility.

In fact, XMLmind XSL Server and XMLmind XSL Utility can share the same user preferences directory:

- *`$HOME`*`/.xfc/` on Linux.
- *`$HOME`*`/Library/Application Support/XMLmind/FOConverter/` on the Mac.
- *`%APPDATA%`*`\XMLmind\FOConverter\` on Windows. Example: `C:\Users\joe\AppData\Roaming\XMLmind\FOConverter\`.

OR

XMLmind XSL Server can use its own customization directory initialized and modified using XMLmind XSL Utility. In order to do this, you'll have to run "`xslutil -p` *`customizeDir`*".

No need to stop and restart XMLmind XSL Server when the contents of the *customizeDir* directory has been modified.

Value: this directory and its contents must be readable by the operating system account used to run XMLmind XSL Server.

Default: none. You are limited to using the stock conversion specifications as is.

`keepTimeWhenDone`

After the end of a successful conversion job, keep the job result available to the client for at least this number of seconds.

Value: Strictly positive integer. Unit: second.

Default: 300 (5 minutes).

keepTimeWhenFailed

After the end of a failed conversion job, keep the job status available to the client for at least this number of seconds.

Value: Strictly positive integer. Unit: second.

Default: 60 (1 minute).

maxConversions

The maximum number of concurrent conversions. Requested conversions beyond that number are queued, that is, they are not discarded.

Value: an integer between 1 and 1000 inclusive.

Default: 20.

mimeTypesFile

A plain text file associating a MIME type to one or more filename extensions.

The format is similar to the one used by Apache `httpd`:

```
file -> line*
line -> OPEN_LINE | COMMENT_LINE | spec_line
spec_line -> mime_type (SPACE extension)*
```

Example:

```
# This is a comment line.
application/x-dbm
application/rtf rtf
application/x-compressed-tar tar.gz tgz
```

Value: this file must be readable by the operating system account used to run XMLmind XSL Server.

Default: the "`mime.types`" file found in `xslsrv.war`.

urlPrefixes

Restrict the use of XMLmind XSL Server to converting URLs starting with specified prefixes.

When this parameter has been specified, converting uploaded files is not allowed.

Value: zero or more regular expression patterns matching URL prefixes, separated by whitespace. URL prefixes must be properly escaped (e.g. they cannot contain whitespace).

Example 1: "`http(s)?://www\.acme\.com/`".

Example 2: "`file:/usr/local/httpd/xmlmind/ file:/tmp/`".

Default: none. Allow converting any URL and also allow converting uploaded files.

validateHTTPS

In some cases, the documents to be converted are served by an HTTPS server. If false, do not check the validity of the certicate presented by an HTTPS server. This useful to cope with self-signed certificates.

Value: true or false.

Default: true.

workDir

Uploaded files and conversion results are stored in subdirectories of this directory.

If specified directory does not exist, it will be created.

Value: this directory and its contents must be readable and writable by the operating system account used to run XMLmind XSL Server.

Default: dynamic; supplied by the Servlet Container.

`maxRequestSize` (support of "`multipart/form-data`" requests)
The maximum size allowed for "`multipart/form-data`" requests.

Value: an integer byte count between 1048576 (1Mb) and 2147483647, inclusive.

Default: 20971520 (20Mb).

`maxFileSize` (support of "`multipart/form-data`" requests)
The maximum size allowed for uploaded files.

Value: an integer byte count between 524288 (512Kb) and 2147483647, inclusive.

Default: 10485760 (10Mb).

`fileSizeThreshold` (support of "`multipart/form-data`" requests)
The size threshold after which the file will be written to disk.

Value: an integer byte count between 256 and 2147483647, inclusive.

Default: 16384.

`idXSLTParameterName`
Regular expression matching XSLT parameter names specifying which XSLT parameters are to be checked by XMLmind XSL Server as having values containing syntactically correct XML or HTML IDs. Such XSLT parameters may be found in some conversion specifications. Example of such XSLT parameter: `root-id=xmncontents`.

Default: "", that is, no such XSLT parameter names. However, the sample `web.xml` file contained in shipped `xslsrv.war` specifies "`root\-?id`" because XSLT parameter "`rootid`" is implemented by the DocBook XSL Stylesheets and XSLT parameter "`root-id`" is implemented by the XHTML XSL Stylesheets.

2. User authentication

XMLmind XSL Server has primarily been designed to be used by Web clients and by other server-side applications as a programmable XML conversion service. As such, by default, user authentication is turned off. The following procedure explains how to turn user authentication on.

Important

Configuring XMLmind XML Server always involves modifying files found in `xslsrv.war`. This implies unpacking[(2)] `xslsrv.war`, modifying some configuration files and then repacking `xslsrv.war`. Fortunately, this is generally done once for all.

[(2)] A `.war` file is just a ZIP archive having a "`.war`" suffix.

Procedure

1. Edit `xslsrv/WEB-INF/web.xml` (`xslsrv/` being the unpacked `xslsrv.war`) using a text editor and remove the line starting with "`<!--AUTH`" and also the line ending with "`AUTH-->`".

```
<!--AUTH
<security-constraint>
  <web-resource-collection>
    <web-resource-name>ConvertServlet</web-resource-name>
    <url-pattern>/exec/*</url-pattern>
  </web-resource-collection>

  <auth-constraint>
    <role-name>user</role-name>
  </auth-constraint>
</security-constraint>

<login-config>
  <auth-method>DIGEST</auth-method>
  <realm-name>XSL Server</realm-name>
</login-config>

<security-role>
  <role-name>user</role-name>
</security-role>
AUTH-->
```

2. Specify the authentication scheme and specify how to perform the user authentication.

   a. You may want to change the authentication scheme from `DIGEST` (default value; recommended for production use) to `BASIC` (simpler to configure).

```
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>XSL Server</realm-name>
</login-config>
```

   b. Specify to the Servlet Container how the user authentication is to be performed.

      Simplest Tomcat example, assuming a `BASIC` authentication scheme:

      1. Add directory *`install_dir`*`/doc/manual/tomcat/META-INF/` to `xslsrv.war`.

         Directory `META-INF/` contains file `context.xml`:

```
<Context ... >
  <Realm className="org.apache.catalina.realm.MemoryRealm"
         pathname="conf/tomcat-users.xml"/>
</Context>
```

      2. Copy *`install_dir`*`/doc/manual/tomcat/tomcat-users.xml` to `/opt/tomcat/conf/`.
      3. Edit the contents of `/opt/tomcat/conf/tomcat-users.xml` using a text editor to declare some users:

```
<tomcat-users>
  <role rolename="user"/>
  <user username="john" password="secret" roles="user"/>
  ...
```

Related information

- Apache Tomcat: The Context Container
- Apache Tomcat: Realm Configuration HOW-TO

Part II. Configuring XMLmind XSL Server

In the previous chapter, we have seen that `xslsrv/WEB-INF/web.xml` (`xslsrv/` being the unpacked `xslsrv.war`) contained important settings . However this file does not allow to specify how XML documents are to be converted to other formats.

In the following chapters, we'll learn

- what is a conversion specification,
- where the stock conversion specifications (`ditaToEPUB`, `db5ToDocx`, `dbToPDF`, etc) are found,
- how to customize a stock conversion specification,
- how to add your own conversion specifications.

Important

Configuring XMLmind XML Server always involves modifying files found in `xslsrv.war`. This implies unpacking[(3)] `xslsrv.war`, modifying some configuration files and then repacking `xslsrv.war`. Fortunately, this is generally done once for all.

[(3)] A `.war` file is just a ZIP archive having a "`.war`" suffix.

Chapter 3. Specifying how to convert an XML document to another format

What is a conversion specification?

How to convert an XML document to another format is specified in XML configuration files called `xslutil.conversions`. Excerpts of `xslsrv/addon/config/xhtml/xslutil.conversions` (`xslsrv/` being the unpacked `xslsrv.war`):

```
<conversions xmlns="http://www.xmlmind.com/xslutil/schema/conversion">
  <conversion name="xhtmlToPDF"
    description="Convert to PDF using Apache FOP"
    icon="icon:acroread"
    category="XHTML"
    categoryIcon="icon:xhtml_file"
    outputExtension="pdf">
    <transform styleSheet="xsl/fo.xsl">
      <parameter name="paper.type">A4</parameter>

      <!-- Needed to convert a remote URL rather than a local file. -->
      <parameter name="img-src-path">%~pi</parameter>
    </transform>

    <processFO processor="FOP">
      <parameter name="renderer">pdf</parameter>
      <parameter name="strict-validation">false</parameter>
    </processFO>
  </conversion>
</conversions>
```

A conversion specification is always referred to by its name, which is unique. In the above example, the conversion specification is called "`xhtmlToPDF`" and allows to convert an XHTML document to PDF using Apache FOP.

The format of `xslutil.conversions` is not yet documented. Fortunately, XMLmind XSL Utility, a user-friendly desktop application allows to create and edit conversion specifications quickly and easily.

Where does XMLmind XSL Server find all its conversion specifications?

During its initialization, that is, the first time the XMLmind XSL Server WebApp is invoked after its Servlet Container has been started, XMLmind XSL Server searches the following directories

1. `xslsrv/addon/`, where `xslsrv/` is the unpacked `xslsrv.war`,
2. *`customize_dir`*`/addon/`, where *`customize_dir`* is the directory pointed to by the `customizeDir` server parameter, if any.

for files called `xslutil.conversions`, which contain conversion specifications and also for files whose basename ends with "`atalog.xml`", which are XML catalogs [(4)]. The conversion specifications collected during these stage

[(4)] Let's suppose you want to convert an XHTML document to RTF. The XHTML file starts with:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xml:lang="en" xmlns="http://www.w3.org/1999/xhtml">
...
```

are immutable. Moreover their number is fixed, that is, adding or removing `xslutil.conversions` files to/from any of the two `addon/` directories will not be detected by XMLmind XSL Server. We'll call this set *the stock conversion specifications*.

During its initialization, XMLmind XSL Server also attempts to load *`customize_dir`*`/xslutil.conversions`, which may contain redefinitions of some of the stock conversion specifications and/or custom conversion specifications.

Specifying the customization directory of XMLmind XSL Server

If you carefully read the above section, you'll understand that, unless you properly define the `customizeDir` server parameter, you'll be limited to using the stock conversion specifications, as is.

Procedure:

1. Download and install XMLmind XSL Utility.
2. Specify the `customizeDir` server parameter in `xslsrv/WEB-INF/web.xml` (`xslsrv/` being the unpacked `xslsrv.war`).

   In the following example, we'll suppose that `customizeDir` points to `/opt/xslsrv/`. Use a text editor and edit `xslsrv/WEB-INF/web.xml`. Give a value to the `customizeDir` `init-param`.

   ```
   <init-param>
   <param-name>customizeDir</param-name><param-value>/opt/xslsrv</param-value>
   </init-param>
   ```

3. Open a terminal or a command prompt and start XMLmind XSL Utility as follows:

   ```
   $ xslutil -p /opt/xslsrv &
   ```

4. Quit XMLmind XSL Utility by clicking its **Quit** button. This is sufficient to create and properly initialize `/opt/xslsrv/`.

   ```
   $ ls /opt/xslsrv
   addon
   xslutil.conversions
   xslutil.properties
   ```

Without an XML catalog, `xhtml1-transitional.dtd` will be downloaded from `http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd` in order to be parsed, which will make the conversion really slow.

With the following XML catalog, `xslsrv/addon/config/catalog.xml`:

```
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
         prefer="public">
  ...
  <public publicId="-//W3C//DTD XHTML 1.0 Transitional//EN"
          uri="xhtml/dtd/1.0/xhtml1-transitional.dtd"/>
  ...
</catalog>
```

It's the local copy of the DTD, `xslsrv/addon/config/xhtml/dtd/1.0/xhtml1-strict.dtd`, which is going to be parsed.

Important

Make sure that the user account used to run the Servlet Container (e.g. user: `tomcat`, group: `tomcat`) has sufficient privileges to read the contents of the directory pointed to by the `customizeDir` server parameter.

Tip

XMLmind XSL Server can directly use the user preferences directory of XMLmind XSL Utility. In other words, no need to create a special purpose directory (`/opt/xslsrv/` in the above examples) for that.

The user preferences directory of XMLmind XSL Utility is:

- *`$HOME`*`/.xfc/` on Linux.
- *`$HOME`*`/Library/Application Support/XMLmind/FOConverter/` on the Mac.
- *`%APPDATA%`*`\XMLmind\FOConverter\` on Windows. Example: `C:\Users\joe\AppData\Roaming\XMLmind\FOConverter\`.

Example (user: `joe` on Linux):

```
<init-param>
<param-name>customizeDir</param-name><param-value>/home/joe/.xfc</param-value>
</init-param>
```

Tip

XMLmind XSL Server can directly use an existing Apache FOP configuration file.

The location of an existing FOP configuration file may be specified to XSL Server by the means of Java™ system property or environment variable *`XXE_FOP_CONFIG`*. The value of this variable is an URL or an absolute file path. Examples: `-DXXE_FOP_CONFIG=http://localhost/~john/fop/fop.conf` (Java™ system property), `set XXE_FOP_CONFIG=C:\Users\john\misc\fop\fop.conf` (Windows environment variable).

You'll need to restart your Servlet container after setting system property or environment variable *`XXE_FOP_CONFIG`*.

Chapter 4. Customizing a stock conversion

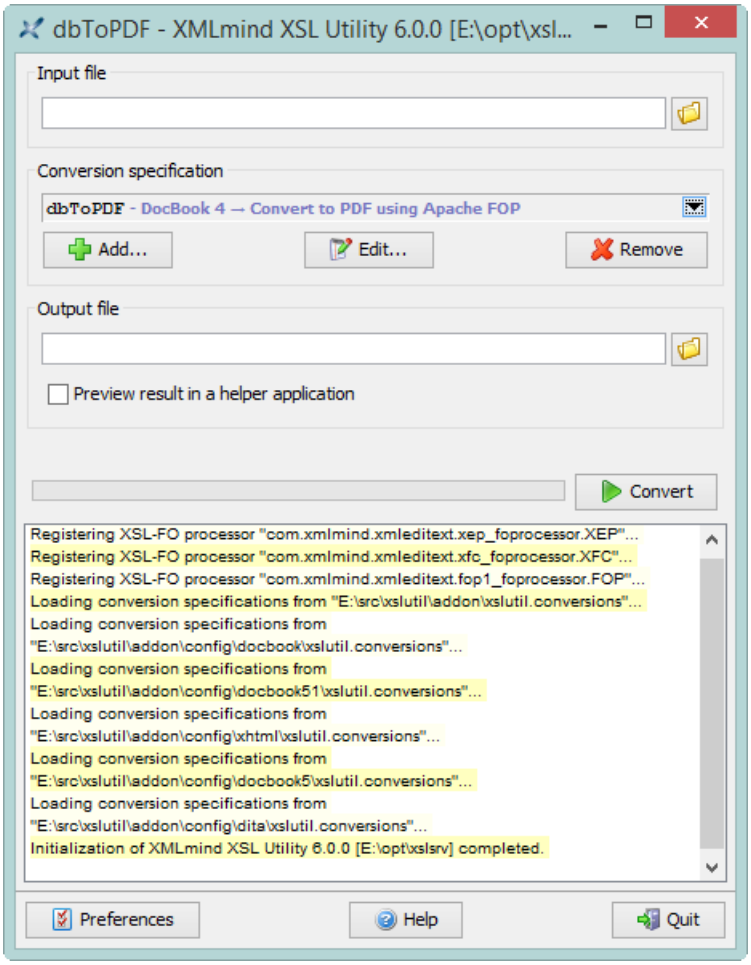
We'll explain how to customize a stock conversion by using the following example: change the paper type of conversion specification `dbToPDF` — convert a DocBook 4 document to PDF using Apache FOP — from the "`A4`" default value to "`USLetter`".

Before you begin

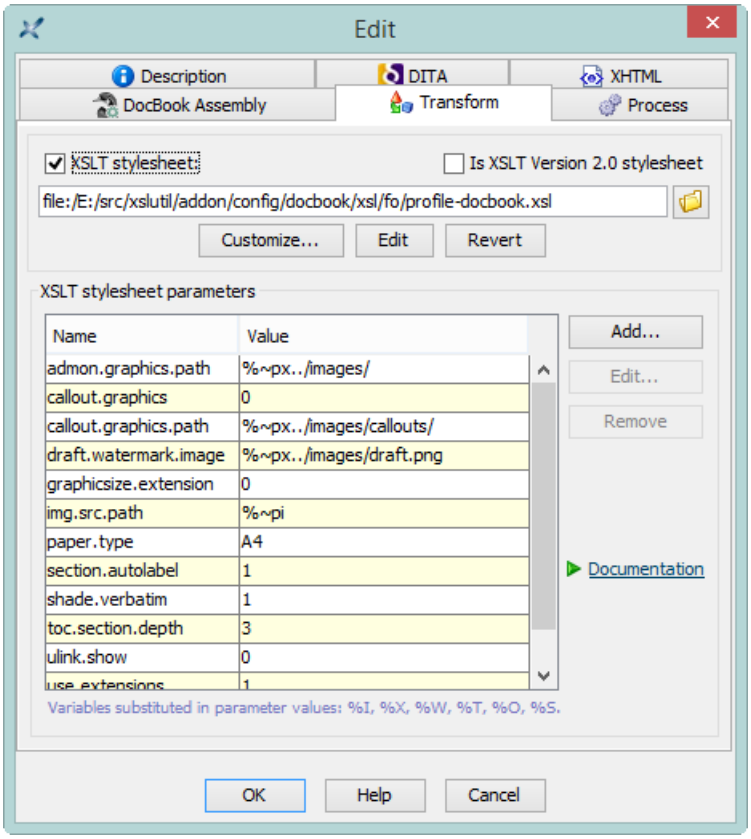
XMLmind XSL Server is assumed to have been configured in order to have a customization directory as explained in Specifying the customization directory of XMLmind XSL Server. In the following example, we'll suppose that this customization directory is `E:\opt\xslsrv/`.

Procedure

1. Start XMLmind XSL Utility by executing `xslutil -p E:\opt\xslsrv &`.
2. Select `dbToPDF`, the conversion specification to be modified.

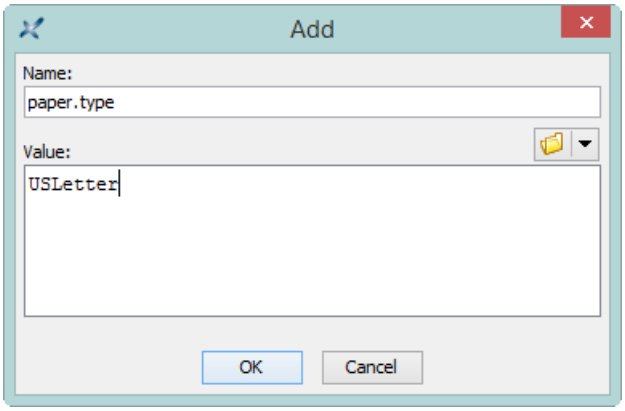


3. Click the **Edit** button.
4. Select the **Transform** tab.



5. Click the **Add** button to add XSLT stylesheet parameter `paper.type=USLetter`.

   The DocBook XSL stylesheets parameters are documented in DocBook XSL Stylesheet Reference Documentation. Clicking the **Documentation** link allows to directly open the aforementioned reference manual in your Web browser.



6. Click **OK** twice to close all the dialog boxes.
7. Click **Quit** to quit XMLmind XSL Utility.

   Next time you'll convert a DocBook 4 document to PDF, XMLmind XSL Server will automatically use your updated `dbToPDF` conversion specification.

   Notice that stopping and then restarting XMLmind XSL Server was not needed in order to customize a stock conversion.

Remember

XMLmind XSL Utility has other uses related to XMLmind XSL Server.

a. It allows to specify a custom font map for Apache FOP. For example, use `Verdana` instead of the `sans-serif` logical font family.

   This is done very easily by clicking **Preferences** and then selecting **XSL-FO Processor**|**FOP**. See XMLmind XSL Utility - Online Help, FOP preferences.

   However in such case, you'll have to stop and then restart XMLmind XSL Server.

b. It allows to specify helpers applications, for example, the location of the HTML Help compiler, `hhc.exe`, which is needed to convert a DITA document to HTML Help (`.chm` file).

   This is done very easily by clicking **Preferences** and then selecting **Helper Applications**. See XMLmind XSL Utility - Online Help, Helper applications preferences .

   When this is the case, you do *not* need to stop and then restart XMLmind XSL Server.

Chapter 5. Adding a custom conversion

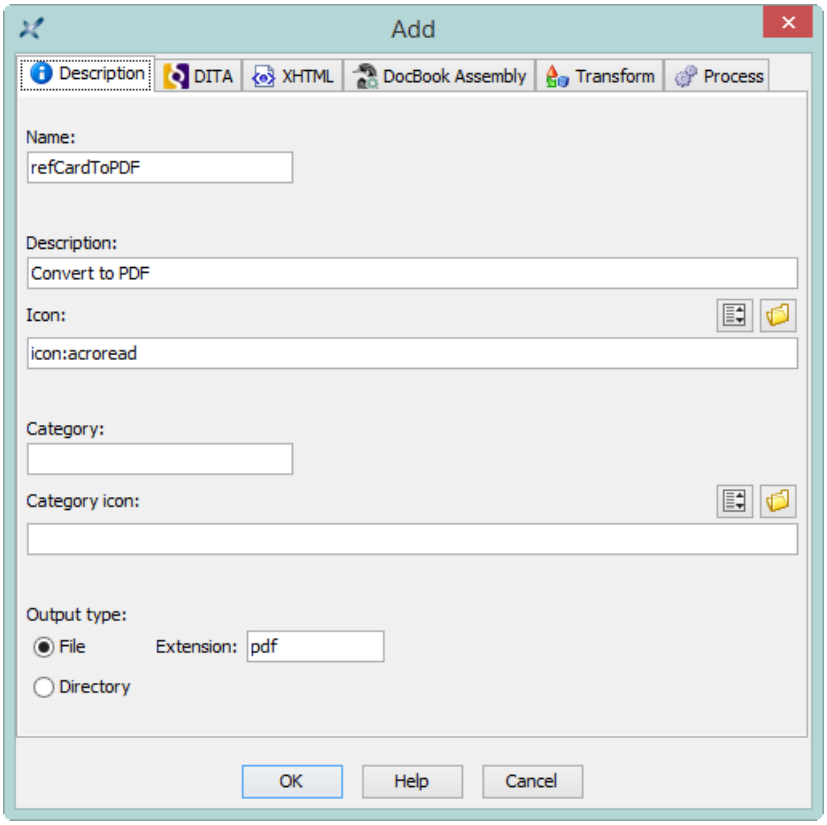
We'll explain how to add a custom conversion by using the following example: define a conversion called "`refCardToPS`" which allows to convert a quick reference card to PostScript using Apache FOP. The XSLT stylesheets needed to perform that conversion are found in `/opt/quickrefcard/`.

Before you begin

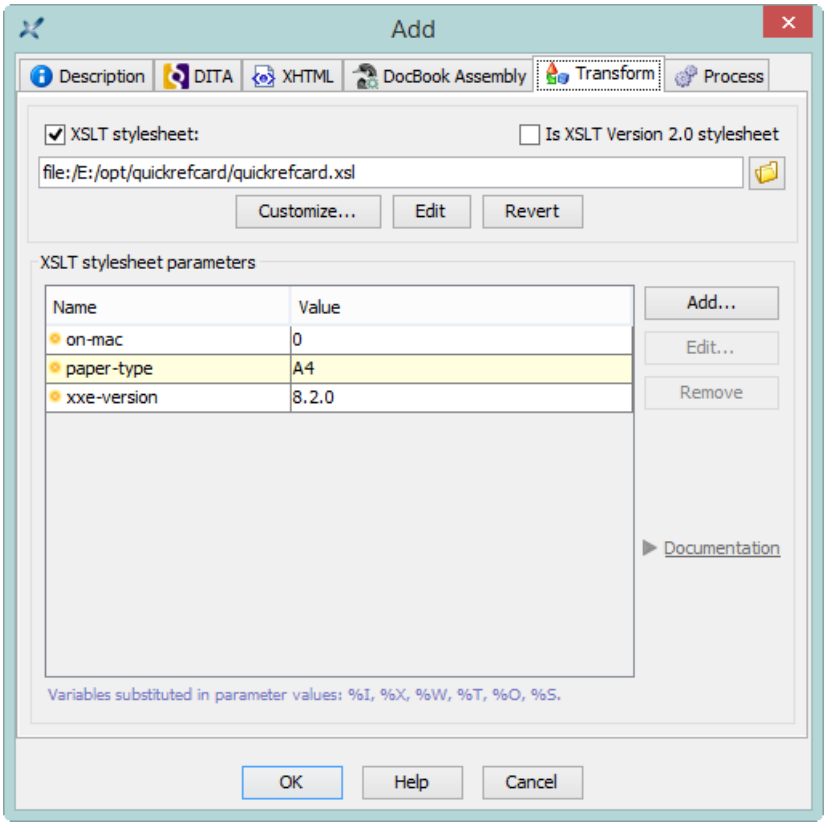
XMLmind XSL Server is assumed to have been configured in order to have a customization directory as explained in Specifying the customization directory of XMLmind XSL Server. In the following example, we'll suppose that this customization directory is `E:\opt\xslsrv/`.

Procedure

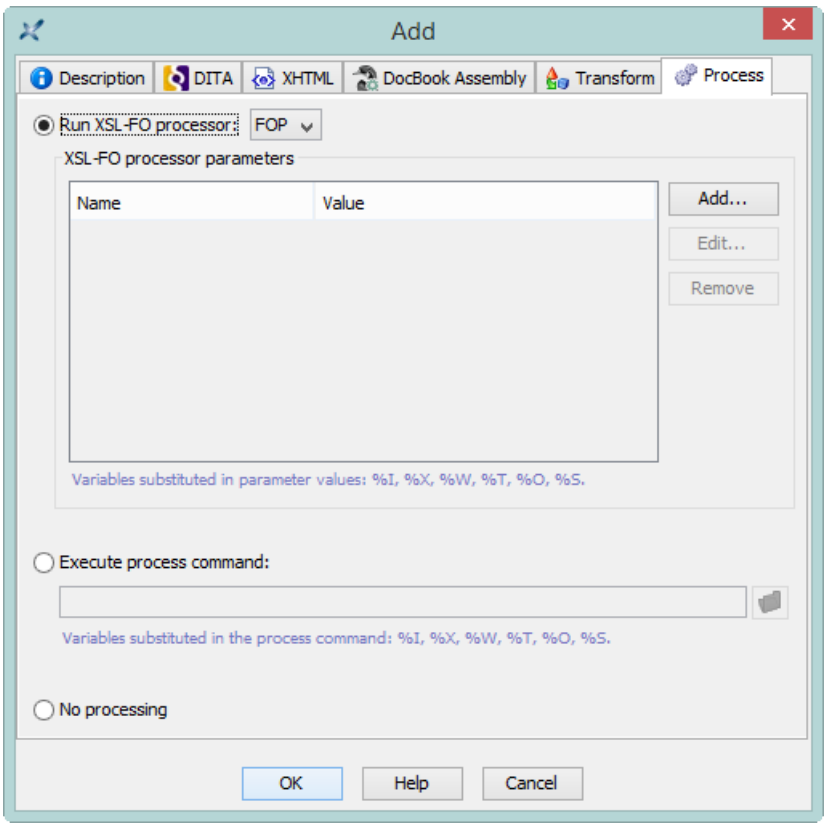
1. Start XMLmind XSL Utility by executing `xslutil -p E:\opt\xslsrv &`.
2. Click **Add**.
3. Click the **Description** tab and describe your conversion as follows.



4. Click the **Transform** tab and specify the location of the main XSLT stylesheet. While at it, add one or more XSLT stylesheet parameters.



5. Click the **Process** tab and choose FOP as your XSL-FO processor.



6. Click **OK** to close the dialog box.
7. Click **Quit** to quit XMLmind XSL Utility.

   From now, you'll be able to perform a `refCardToPDF` conversion in addition to all the stock conversions.

   Notice that stopping and then restarting XMLmind XSL Server was not needed in order to add a custom conversion.

Attention

May be you'll have to slightly modify your custom XSLT stylesheets in order to use them in XMLmind XSL Server. Let's use an example to explain a possible issue.

When you use XMLmind XSL Server to convert a remote document such as https://www.xmlmind.com/xmleditor/_distrib/demo/docbook/docbook-image.xml to PDF or RTF , the server generates a temporary XSL-FO file on its host. Let's call this temporary file `/tmp/xslu12345.fo`.

By default, an XML element such as:

```
<imagedata fileref="graphics/fish1.png"
           format="PNG"/>
```

is translated to something like:

```
<fo:external-graphic src="graphics/fish1.png"
                     content-type="content-type:image/png"/>
```

Unfortunately this cannot work because there is no `/tmp/graphics/fish1.png` file on the host running XMLmind XSL Server.

Instead, the above XML element should be translated to:

```
<fo:external-graphic
src="https://www.xmlmind.com/xmleditor/_distrib/demo/graphics/fish1.png"
content-type="content-type:image/png"/>
```

The `dbTo`*XXX* and `db5To`*XXX* conversion specifications pass a number of parameters such as `img.src.path` to their XSLT stylesheets in order to get absolute URLs in the generated XSL-FO file. May be you'll have to add a similar facility to your own XSL stylesheets.[(5)].

[(5)] The `xhtmlTo`*XXX* conversion specifications pass `img-src-path` to their XSLT stylesheets.

The `ditaTo`*XXX* conversion specifications, which use XMLmind DITA Converter (ditac), have no such issue.

Part III. Using XMLmind XSL Server

XMLmind XSL Server has primarily been designed to be used by Web clients (JavaScript, Ajax, Flex, etc) and by other server-side applications (PHP, ASP, etc) as a ``behind-the-scene'', programmable XML conversion service.

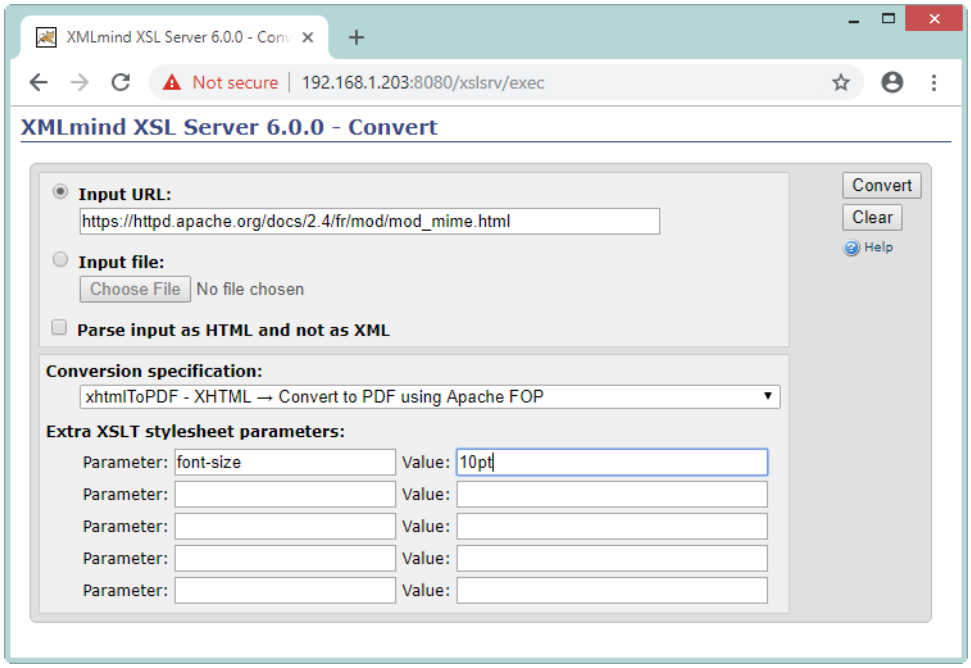
However, XMLmind XSL Server also has a comprehensive user-interface.

Chapter 6. User interface (interactive requests)

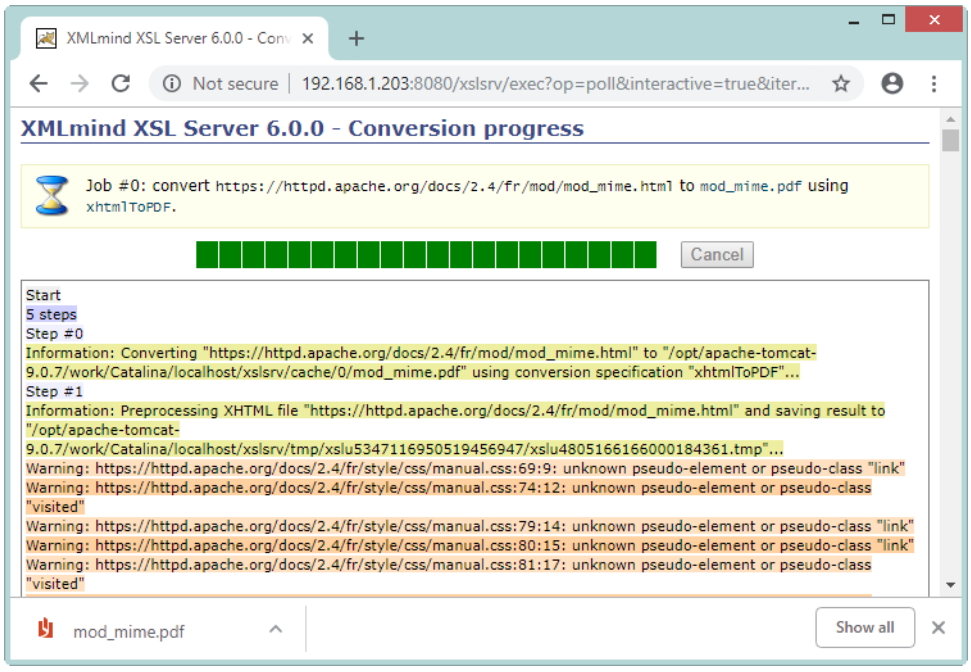
The user interface of XMLmind XSL server is displayed by the Web browser if you open in it any of the following 3 URLs.

Request `xslsrv/exec` (shorthand for `xslsrv/exec?op=convert&mode=interactive`)

Displays an interactive form allowing to convert an XML document to another format. Click **Help** to learn how to use this form.



During the execution of a convert job, a progress page is displayed:



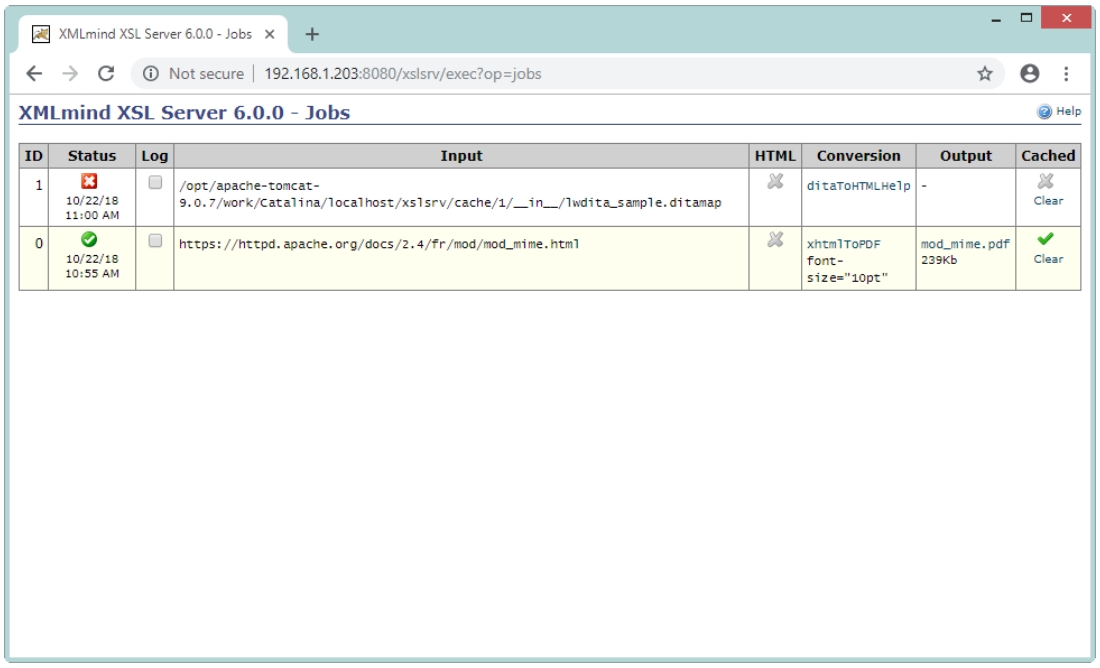
Clicking the **Cancel** button allows to cancel the execution of this job.

Remember

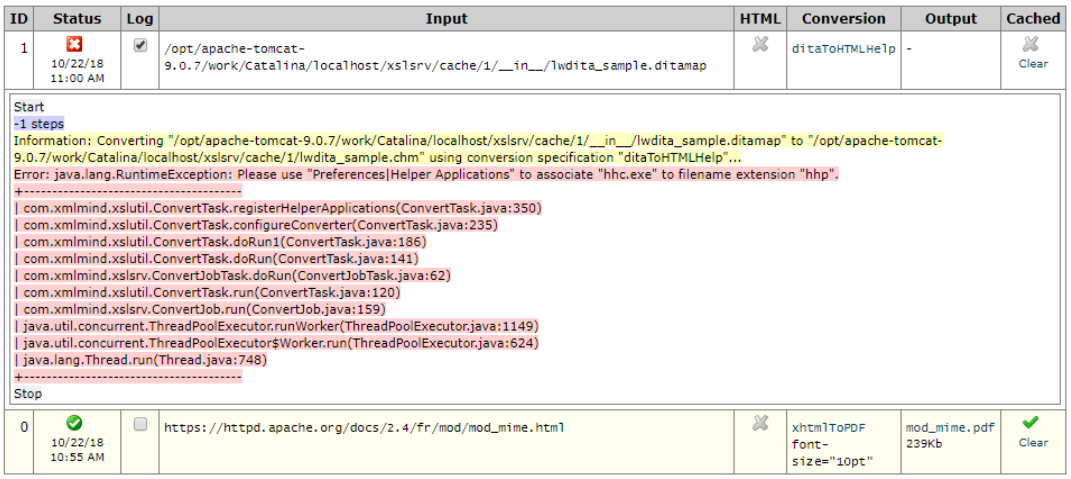
If the `authorization1` server parameter has been specified, you'll need to open `xslsrv/exec?auth=`*`authorization1_token`*.

Request xslsrv/exec?op=jobs

Displays a page listing all the convert jobs still remembered by XMLmind XSL Server, from most recent to least recent. Click **Help** to learn how to use this page.



Clicking the **Log** checkbox expands a section showing the progress log of the corresponding job.

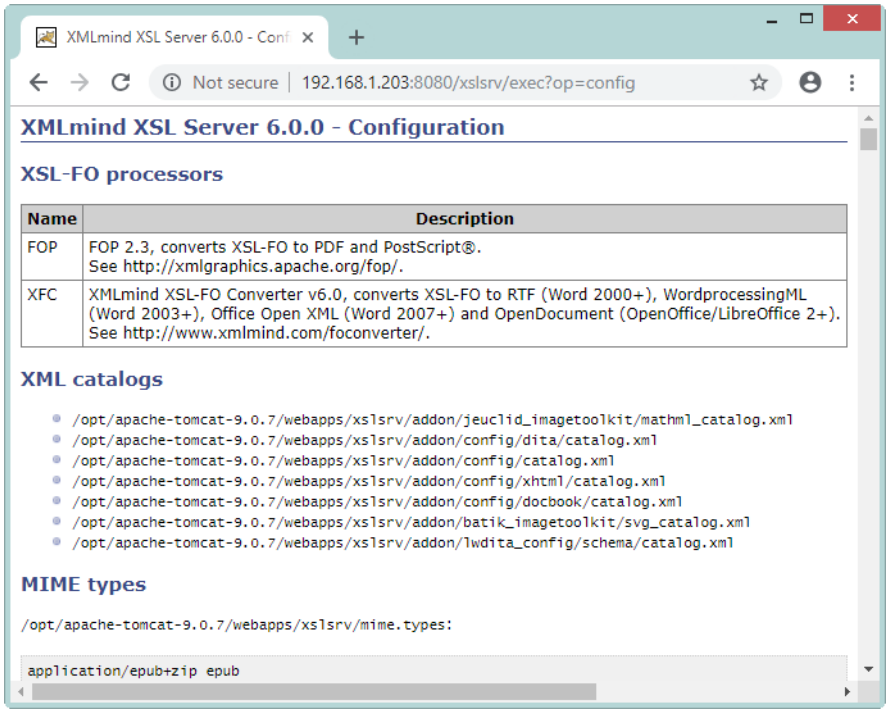


Remember

If the `authorization2` server parameter has been specified, you'll need to open `xslsrv/exec?op=jobs&auth=`*`authorization2_token`*.

Request xslsrv/exec?op=config

Displays a page listing the current configuration of XMLmind XSL Server.



If you have used XMLmind XSL Utility to modify this configuration and now want to see your changes, you'll have to instruct your Web browser to reload this page (generally this is done by clicking the **Refresh** button).

Chapter 7. API (non-interactive requests)

The API of XMLmind XSL Server consists in just 5 requests. These requests may be sent to the server using either `HTTP GET` or `HTTP POST application/x-www-form-urlencoded`.

Request `xslsrv/exec?op=convert`

Description:

Create a new job allowing to convert specified XML document to another format using specified conversion specification.

If a similar job (same `url`, `html`, `conv`, `paramI`/`valueI`, `out` parameters) already exists, no new job is created and the existing job is returned instead.

Parameters other than `op`:

Name	Value	Default value	Description
`mode`	`async` | `sync` | `interactive`	`interactive`	`Mode=async` or `mode=sync` must be specified, otherwise this request returns an interactive **Convert** form. The difference between the two modes is explained below.
`conv`	The name of a conversion specification.	Required.	Specifies which conversion is to be applied to the input document. **Tip** Invoke `xslsrv/exec?op=config` to list all the available conversion specifications.
`paramI` (*I* between 0 and 99)	Non-empty string.	None	Pass these extra parameters to the XSLT stylesheets referenced by the conversion specification.
`valueI`	String.	None	Value of XSLTstylesheet parameter `paramI`.
`url`	URL	One of `url` and `file0` is required.	URL of the input document. May not have a fragment identifier (e.g. ends with "`#foo`").
`fileI` (*I* between 0 and 99)	Uploaded file.	One of `url` and `file0` is required.	**Attention** When `fileI` parameters are used, the request must be sent using `HTTP POST multipart/form-data`.

Name	Value	Default value	Description

			The input document is contained in `file0`. Attachments (e.g. graphic files) are contained in `file`*I*, where *I* is an integer between 1 and 99.
			
			Alternatively, all the files may be packed in a Zip archive. However this must be done exactly like in the following example.
			
			Let's suppose you want to convert `userguide.fo` to RTF and `userguide.fo` references several graphic files found in subdirectory `std/`. You'll have to create `userguide.fo.zip` as follows:
			
			`~/tmp$ ls`
			`sdt`
			`userguide.fo`
			
			`~/tmp$ ls sdt`
			`drop-down-list-1.png`
			`drop-down-list-2.png`
			`...`
			
			`~/tmp$ zip -r userguide.fo.zip \`
			`  userguide.fo sdt`
			`~/tmp$ ls`
			`sdt`
			`userguide.fo`
			`userguide.fo.zip`
			
			• The Zip archive must be named after the XML document to be converted. That is, append "`.zip`" to the basename of the XML document to be converted.
			• The Zip archive must directly contain the XML document to be converted. Subdirectories are allowed only for the referenced files.

`html`	`true` | `false`	`false`	The input document must be parsed as HTML, and not as XML or XHTML (XHTML is a special kind of XML).
`out`	String	Same basename as the input file but with the extension specified in the conversion specification (e.g. "`.pdf`" for `db5ToPDF`). If the conversion specification generates a	The basename of the result file.

Name	Value	Default value	Description
		directory (e.g. `ditaToXHTML`), the contents of this directory is packed in a Zip archive having the same basename of the input file but with a "`.zip`" extension.	
`cache`	`true` | `false`	• `true`, if the input document has been specified by its URL. • `false`, if the input document has been uploaded.	If `true`, cache the result of this job. Note that only the result of converting an URL can be cached. The result of converting an uploaded file cannot be cached.
`auth`	String.	None. Required and equal to *authorization1_token* if the `authorization1` server parameter has been specified.	Authorization token specified for conversion operations.

Response:

- If `mode=async`, `plain/text;charset=UTF-8` containing the ID of the job. This job is intended to be polled using `xslsrv/exec?op=poll`.
- If `mode=sync`,
  - If the job has successfully been executed, the file which is the result of the job.
  - Otherwise, a `plain/text;charset=UTF-8` file containing its progress log. This file has the following format.

Examples: Requires cURL, a command-line tool for transferring data using various protocols. (In the examples below, long lines have been folded. In reality, this is not the case.)

```
$ curl 'http://localhost:8080/xslsrv/exec?op=convert&mode=async¬
&conv=xhtmlToPDF&param0=root-id&value0=contents-box¬
&out=xfc.pdf¬
&html=true&url=http://www.xmlmind.com/foconverter/what_is_xfc.html'
7

$ curl -o xfc.pdf 'http://localhost:8080/xslsrv/exec/xfc.pdf?op=result&job=7'

$ acroread xfc.pdf &

$curl -o tutorial.zip 'http://localhost:8080/xslsrv/exec?op=convert&mode=sync¬
&conv=ditaToXHTML¬
&url=http://www.xmlmind.com/tutorials/DITA/src/tutorial.ditamap'
```

```
$ unzip -v tutorial.zip
```

Request `xslsrv/exec?op=poll`

Description:

If specified job has successfully been executed, download its result file, otherwise download a plain text file containing its progress log.

Parameters other than `op`:

Name	Value	Default value	Description
`job`	Integer larger than or equal to 0.	Required.	The ID of the job which is to be polled.
`from`	Integer	0.	If specified job has successfully been executed, this parameter is ignored. Otherwise this parameter specifies the index of the first progress message that should appear in the downloaded log file. The first index is 0. A negative index means: return the end of the log file, that is, count messages starting from the end of the log file rather than from its beginning.
`auth`	String.	None. Required and equal to *authorization1_token* if the `authorization1` server parameter has been specified.	Authorization token specified for conversion operations.

Response:

If specified job has successfully been executed, the file which is the result of the job. Otherwise a `plain/text;charset=UTF-8` log file having this format:

```
file -> job_status
        '\n' index_of_first_message
        '\n' message_count
        [ '\n' message ]*

job_status -> CREATED|RUNNING|CANCELED|FAILED|DONE|CANCELING|CRASHED

index_of_first_message -> Positive_Integer

message_count -> Positive_Integer

message -> start_message |
           step_count_message |
           step_message |
           progress_message |
```

```
           stop_message

start_message -> START ' ' -

step_count_message -> STEP_COUNT ' ' -1|Positive_Integer

step_message -> STEP ' ' Positive_Integer

progress_message -> MESSAGE ' ' ERROR|WARNING|INFO|VERBOSE|DEBUG ': ' Text_Line

stop_message -> STOP ' ' -
```

- The very first message of the progress log is guaranteed to be *start_message*.
- *Step_count_message* is guaranteed to occur before *step_message* (if any *step_message*).
- *STEP_COUNT* -1 means: undetermined number of steps.
- In *progress_message*, the backslash character is escaped as "\\" and the newline character is escaped as "\n".
- The very last message of the progress log is guaranteed to be *stop_message*.

Examples: (In the examples below, long lines have been folded. In reality, this is not the case.)

```
$ curl -o 'http://localhost:8080/xslsrv/exec?op=poll&job=12345'
UNKNOWN
0
0

$ curl -o 'http://localhost:8080/xslsrv/exec?op=poll&job=5'
RUNNING
0
9
START -
STEP_COUNT 4
STEP 0
MESSAGE INFO: Converting¬
"http://www.xmlmind.com/xmleditor/_distrib/demo/docbook/docbook-image.xml"¬
to "/opt/tomcat2/work/Catalina/localhost/xslsrv/cache/6/docbook-image.pdf"¬
using conversion specification "dbToPDF"...
STEP 1
MESSAGE INFO: Compiling XSLT stylesheet¬
"/opt/tomcat2/webapps/xslsrv/addon/config/docbook/xsl/fo/docbook.xsl"...¬
MESSAGE INFO: XSLT stylesheet compiled in 0.0s.
STEP 2
MESSAGE INFO: Transforming¬
"http://www.xmlmind.com/xmleditor/_distrib/doc/demo/docbook/docbook-image.xml"¬
to "/opt/tomcat2/work/Catalina/localhost/xslsrv/tmp/xslu8379/xslu5120.tmp"¬
using XSLT stylesheet¬
"/opt/tomcat2/webapps/xslsrv/addon/config/docbook/xsl/fo/docbook.xsl"...

$ curl -o 'http://localhost:8080/xslsrv/exec?op=poll&job=5&from=-3'
STEP 3
MESSAGE INFO: Converting
"/opt/tomcat2/work/Catalina/localhost/xslsrv/tmp/xslu8379/xslu5120.tmp"¬
to "/opt/tomcat2/work/Catalina/localhost/xslsrv/cache/6/docbook-image.pdf"¬
```

```
using XSL-FO processor "FOP"...
MESSAGE WARNING: [FOP WARNING] Font 'Symbol,normal,700' not found.¬
Substituting with 'Symbol,normal,400'.
```

Request `xslsrv/exec?op=cancel`

Description:

Cancel the execution of specified job. The execution of a job can be canceled only when this job is in the **Running** state.

Parameters other than `op`:

Name	Value	Default value	Description
`job`	Integer larger than or equal to 0.	Required.	The ID of the job whose execution is to be canceled.
`auth`	String.	None. Required and equal to *authorization1_token* if the `authorization1` server parameter has been specified.	Authorization token specified for conversion operations.

Response:

`Plain/text;charset=UTF-8` containing `true` if the job has successfully been switched from the **Running** to the **Canceling** state or is already in the **Canceling** state; `false` otherwise.

Examples:

```
$ curl -'http://localhost:8080/xslsrv/exec?op=cancel&job=4'
true

$ curl -'http://localhost:8080/xslsrv/exec?op=cancel&job=4'
true

$ curl -'http://localhost:8080/xslsrv/exec?op=cancel&job=4'
false
```

Request `xslsrv/exec?op=result`

Description:

Download the result of specified job.

Important

As a simple security check, the basename of the file which is the result of the job must be passed in the request URL just after `xslsrv/exec`. See examples below.

Parameters other than `op`:

Name	Value	Default value	Description
`job`	Integer larger than or equal to 0.	Required.	The ID of the job whose result is to be downloaded.
`auth`	String.	None. Required and equal to *authorization1_token* if the `authorization1` server parameter has been specified.	Authorization token specified for conversion operations.

Response:

The file which is the result of the job. If, for any reason, this file does not exist, returns HTTP error "`404, Not Found`".

Examples:

```
$ curl -o docbook-table.pdf \
'http://localhost:8080/xslsrv/exec/docbook-table.pdf?op=result&job=1'

$ acroread docbook-table.pdf &

$ curl -o dita-sample.zip \
'http://localhost:8080/xslsrv/exec/dita-sample.zip?op=result&job=3'

$ unzip -v dita-sample.zip

$curl 'http://localhost:8080/xslsrv/exec/foo.bar?op=result&job=1234'
HTTP Status 404 - the output of job #1234 is not available
```

Request `xslsrv/exec?op=clear`

Description:

Discard the result of specified job.

Note that the result of a job cannot be discarded in the following cases:

- Specified job ID is unknown or is no longer known.
- It is too early to discard the result of specified job. See the `keepTimeWhenDone` and `keepTimeWhenFailed` server parameters.

Parameters other than `op`:

Name	Value	Default value	Description
`job`	Integer larger than or equal to -1.	Required.	The ID of the job whose result is to be discarded. -1 means: all jobs.
`auth`	String.	None. Required and equal to *authorization2_token* if the `authorization2`	Authorization token specified for administrative operations.

Name	Value	Default value	Description
		server parameter has been specified.	

Response:

`Plain/text;charset=UTF-8` containing the number of job results which have been discarded.

Examples:

```
~$ curl 'http://localhost:8080/xslsrv/exec?op=clear&job=12'
0

~$ curl 'http://localhost:8080/xslsrv/exec?op=clear&job=-1'
2
```

Index

A

authorization1, server parameter, 4
authorization2, server parameter, 4

C

cacheSize, server parameter, 5
config, request, 23
convert, request, 21
credentials, server parameter, 5
customizeDir, server parameter, 5

F

fileSizeThreshold, server parameter, 7

I

idXSLTParameterName, server parameter, 7

J

jobs, request, 22

K

keepTimeWhenDone, server parameter, 5
keepTimeWhenFailed, server parameter, 6

M

maxConversions, server parameter, 6
maxFileSize, server parameter, 7
maxRequestSize, server parameter, 7
mimeTypesFile, server parameter, 6

U

urlPrefixes, server parameter, 6

V

validateHTTPS, server parameter, 6

W

web.xml, configuration file, 4, 8, 10, 12
workDir, server parameter, 6

X

XML catalog, 11
XMLmind XSL Utility, 5, 11, 12, 14, 16, 17, 23