MDC Web Service and

eXist XML database installation.

Contents

1. Installing eXist

- Download the latest standard distribution of eXist.
- Start the eXist installation:

  java -jar eXist-<Version>.jar [-p <eXist-Path>]

  Option -p: the path to the directory where you would like to install eXist.

Note: eXist versions 1.2.x requires at least java 1.4.2.

2. Configuring eXist

- <eXist-Path>/**conf.xml** file:

  cacheSize="256M"

  recovery->size="256M"

- <eXist-Path>/**client.properties** file:

  uri=xmldb:exist://localhost:8088/xmlrpc

  #uri=xmldb:exist://localhost:8080/exist/xmlrpc

#alternate_uri_0=xmldb:exist://localhost:8080/exist/xmlrpc

#alternate_uri_1=xmldb:exist://localhost:8088/xmlrpc

#alternate_uri_2=xmldb:exist://

- <eXist-Path>/**backup.properties** file:

  uri=xmldb:exist://localhost:8088/xmlrpc

  #uri=xmldb:exist://localhost:8080/exist/xmlrpc

- <eXist-Path>/bin/functions.d/eXist-settings.sh/**eXist-settings.sh** file:

  JAVA_OPTIONS="${JAVA_OPTIONS} –Xms128m –Xmx512m -Djava.endorsed.dirs=${JAVA_ENDORSED_DIRS}";

3. Running eXist as a Standalone Server

- Starting the stand-alone server:

  <eXist-Path>/bin/**server.sh**

Note: The stand-alone server runs on port 8088 by default. You may specify a different listener port with the -p option.

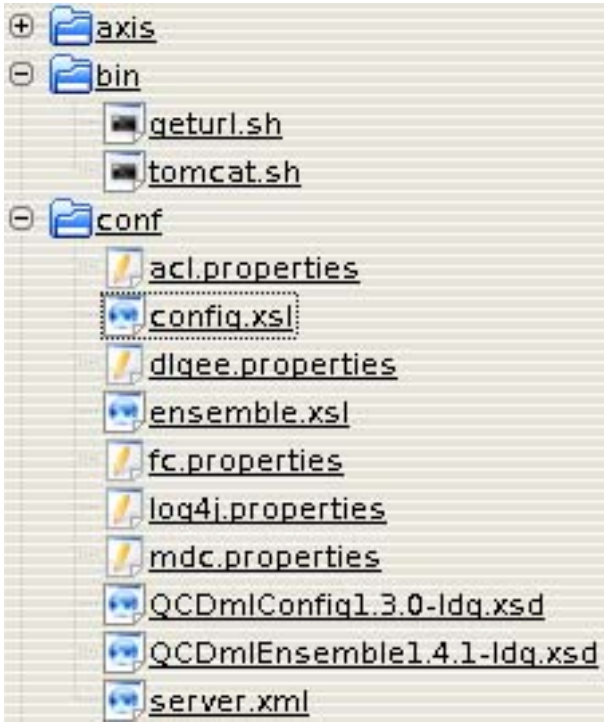
- Database shutdown

  <eXist-Path>/bin/**shutdown.sh**

Note: the shutdown.sh script connects to the default server URI, i.e. xmldb:exist://localhost:8088/xmlrpc

4. Installing and configuring MDC

- Download and install the latest standard distribution of Tomcat5.
- MDC files structure



- copy ‘axis’ directory to ‘webapps’ directory of Tomcat
- copy ‘log4j-1.2.14.jar’, ‘bcprov-jdk15-132.jar’, ‘glite-security-trustmanager.jar’ and ‘glite-security-util-java.jar’ from ‘axis/ WEB-INF/lib’ to ‘server/lib’ directory of Tomcat
- copy ‘config.xsl’, ‘ensemble.xsl’, ‘QCDmlConfigX.X.X.xsd’, ‘QCDmlEnsembleX.X.X.xsd’ files in configuration directory
- copy and configure ‘acl.properties’ configuration file, where:
  - acl.enabled : enabling/disabling ACL for MDC
  - conn.driver : ACL dbase connection Driver
  - conn.url : ACL dbase connection URL
  - conn.user : ACL dbase connection User
  - conn.pswd : ACL dbase connection password

- copy and configure ‘mdc.properties’ configuration file, where:
  - conn.driver : eXist dbase connection Driver
  - conn.url : eXist dbase connection URL
  - conn.user : eXist dbase connection User
  - conn.pswd : eXist dbase connection password
  - ensemble.xsl.file : location of ‘ensemble.xsl’ stylesheet file
  - ensemble.xsd.file : location of ‘QCDmlEnsembleX.X.X.xsd’ schema file
  - ensemble.dump.file : location of ensemble dump/restore data file
    ensemble.xpath.uri : XPath expression of Ensemble XML id
  - ensemble.xpath.namespace : Ensemble namespace definition
  - config.xsl.file : location of ‘config.xsl’ stylesheet file
  - config.xsd.file : location of ‘QCDmlConfigX.X.X.xsd’ schema file
  - config.dump.file : location of metadata dump/restore data file
  - config.xpath.uri : XPath expression of Configutation XML id
  - config.xpath.lfn : XPath of Ensemble XML id in Config XML
  - config.xpath.namespace : Congiruration namespace definition
  - validation.warn.enabled : Warning message level on validating XML
  - data.write.enabled : enabling/disabling MDC data writing
  - data.read.enabled : enabling/Disabling MDC data reading
  - restore.validate.enabled : en/disabling data validation in restore operation
  - xpath.service.name : XPath query service name
  - xpath.service.version : XPath query service version

- copy and configure ‘log4j.properties’ configuration file, where:
  - log4j.appender.file.File : location of logging file
- insert a secure-connector description in server.xml configuration file:

```
<Connector port="8443"
    maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
      enableLookups="false" disableUploadTimeout="true" acceptCount="100"
      debug="0" scheme="https" secure="true" sSLImplementation =
      "org.glite.security.trustmanager.tomcat.TMSSLImplementation"
       sslCAFiles="<certificates>/*.0" crlFiles="<certificates>/*.r0"
       sslCertFile="<hostcert>" sslKey="<hostkey>" clientAuth="true" sslProtocol="TLS"
/>
```

- specify JAVA_OPTS=’-Xms16m -Xmx64m -Dlog4j.configuration=file:<location>/log4j.properties -Dlog4j.debug=false -D mdc.configuration=<location>/mdc.properties –D acl.configuration=<location>/acl.properties –D fc.configuration=<location>/fc.properties –D GLITE_DLGEE_PROPERTY=<location>/dlgee.properties –D com.mchange.v2.log.MLog=com.mchange.v2.log.log4j.Log4jMLog’

5. Administrative tools of MDC

The MDC Administration list of services lets you configure MDC remotely.

web-service	arguments	access
doSetConfig	(name, value)	secure
doGetConfig	-	non- secure
doDump	-	secure
doRestore	-	secure

- **doSetConfig** web-service: sets a specified configuration a specified value. The accepted parameters are:

configuration	values	default
data.read.enabled	{true, false}	true
data.write.enabled	{true, false}	true

Note: Changes made to MDC Core using doSetConfig service can be applied to the current session only.

- **doGetConfig** web-service: gets a list of all configurations with specified values.

- **doDump** web-service : create a dump containing all XML documents stored in the MDC. The output filenames is defined by *ensemble.dump.file/config.dump.file* properties.

  Operations required to dump MDC content:

  1. call doGetConfig if response says *data.write.enabled=false* then 3
  2. call doSetConfig(*data.write.enabled, false*) and go to point 1
  3. call doDump()
  4. periodically call doGetConfig() while currentState in StatusDumpRestore will not be equal to "idle"
  5. call doSetConfig(data.write.enabled, true)

- **doRestore()** web-service: restore XML documents from previous dump files. The output filenames is defined by *ensemble.dump.file/config.dump.filename* properties.

  Operations required to restore MDC content:

  1. call doGetConfig() if response says *data.write.enabled = false* then go to 3
  2. call doSetConfig(*data.write.enabled, false*) and go to point 1
  3. call doSetConfig(mdc.read.enabled, false). Go to point 1
  4. call doRestore()
  5. periodically call doGetConfig() while currentState in StatusDumpRestore will not be equal to "idle"
  6. call doSetConfig(data.write.enabled, true)

6. Migration to a new schema

The script of use:

1. call doGetConfig() if response says *data.write.enabled = false* then go to 3
2. call doSetConfig(*data.write.enabled, false*) and go to point 1
3. call doDump()
4. periodically call doGetConfig() while currentState in StatusDumpRestore will not be equal to “idle”
5. adapt dump files to an updated XML Schema(s).
6. stop MDC
7. update XML Schema(s)
8. set data.write.enabled = false in mdc.properties file
9. start MDC
10. call doRestore()
11. periodically call doGetConfig() while currentState in StatusDumpRestore will not be equal to “idle”
12. call doSetConfig(data.write.enabled, true)
13. set data.write.enabled = true in mdc.properties file