X A C M Light Reference[i]

---

1. Introduction.

XACMLight is an Axis2 [1] web service that implements a Policy Decision Point (PDP) and a Policy Administration Point (PAP) that are defined in XACML 2.0 [2] specification. This implementation covers all functions that are defined by XAMCL 2.0 (including optional), all mandatory elements and almost all optional features. The current implementation has been successfully tested against XACML 2.0 conformance test suite [10] : all applicable tests have been completed successfully.

---

2. Implemented Features and Limitations.

The following mandatory features defined in XACML 2.0 have been implemented:

1. Mandatory functions
2. All mandatory elements

3. All mandatory data types
4. Intrinsic XACML attributes:
    a. urn:oasis:names:tc:xacml:1.0:environment:current-time
    b. urn:oasis:names:tc:xacml:1.0:environment:current-date
    c. urn:oasis:names:tc:xacml:1.0:environment:current-dateTime

The following optional XACML features have been implemented:

1. Multiple results
2. <MissingAttributeDetail> element
3. <StatusMessage> element
4. <StatusDetail> element
5. <AttributeSelector> element
6. Optional functions

The first optional feature above is very useful from practical point of view because it allows evaluating multiple resources in a single XACML request, the next three are very helpful for understanding what is wrong with authorization request and/or policy when the engine returns a 'Indeterminate' result.

XACMLight uses JDK 1.5 XPath implementation for implementing the following optional functions and elements:

1. <AttributeSelector>
2. xpath-node-count
3. xpath-node-equal
4. xpath-node-match

It means that the version of XPath is determined by the version of this particular implementation and the version provided in <PolicyDefaults> and <PolicySetDefaults> will be ignored.

2.1. Policy, Policy Set References and Local Repo.

XACML 2.0 doesn't define how references to policies and policy sets should be resolved. It says only that if reference is a URL a reference MAY be resolved. XACMLight uses the following strategy to resolve references:

1. Searches a local repository
2. If a policy or policy set was not found in the local repository, it tries to interpret the reference as a URL and retrieve it online (e.g. if it's a real http:// URL it will try to fetch it through HTTP protocol)

3. If it's resolved, its version will be validated against 'EarliestVersion' and 'LatestVersion' attributes using algorithm that is described in XACML's 'VersionMatchType' section.
4. If version is in the range the policy or policy set will be inserted, otherwise they will be ignored.

Local repository must be set in <Repo> section of PDP's initial configuration file. Unbounded sequences of the following elements are allowed in <Repo> section:

1. <Policy>
2. <PolicySet>
3. <Attribute>

The syntax of the first two elements in the list are defined by policy XSD [8], the syntax of the third element is defined by context XSD [9]. Policies and Policy Sets in <Repo> section are used for resolving references. Attributes are used for resolving missing attributes in request.

2.2. Configuration File.

The name of default configuration file is gryb_info_xacml_config.xml. It must be placed to an Axis2 CLASSPATH (e.g. to AXIS2_HOME directory) to be available for XACMLight service. The XSD schema for this file is defined in xacml.wsdl file and contains the following elements:

1. <Config> - the root element of configuration file. It can contain one <Policy> or one <PolicySet> element inside.
2. <Repo> - this element is described in section 2.1

3. Supported Platforms.

Since XACMLight is implemented as an Axis2 service, it can run on native Axis2 Apache server or on any other J2EE compliant container using Axix2 war file that allows deploying aar files to other application servers. The current implementation has been tested with the native Axis2 server only.

---

4. WSDL Operations.

WSDL for XACMLight was built using XML schema definitions provided by OASIS ([8]-[9]). The latter schema definitions use abstract XML types and substitution groups that are not handled well by Apache Data Binding Frameworks ([5]-[6]). The following

WSDL operations have been defined in XACMLight's WSDL and implemented in the scope of this project:

Wsdl Operation	Input	Output	Description	Service
setPolicyRoot	<xl:SetPolicyRoot>	<xl:Result>	Can be used to set up a root policy or policy set	PAP
setRepo	<xl:Repo>	<xl:Result>	Can be used to set up PDP's policy and attribute local repository (see section 2.1)	PAP
getRepo	<xl:GetRepo>	<xl:Repo>	Returns the current local repository	PAP
setPolicyRoot	<xl:SetPolicyRoot>	<xl:Result>	Can be used to set up a root policy or policy set	PAP
getDecision	<ctx:Request>	<ctx:Response>	Gets an authorization decision from PDP engine	PDP
setPolicies	<pol:PolicySet>	<xl:Result>	Overrides the root policy set. Deprecated in version 2.2: use setPolicyRoot instead	PAP
getPolicies	<xl:GetPolicies>	<pol:PolicySet>	Returns the current root policy set	PAP
setPolicy	<pol:Policy>	<xl:Result>	Overrides a policy with a given policy ID	PAP
getPolicy	<xl:GetPolicy>	<pol:Policy>	Returns a policy with a given policy ID	PAP
addPolicy	<xl:AddPolicy>, PolicySetId - parent	<xl:Result>	Adds a policy with a given policy ID to a policy set with a given policy set ID	PAP
deletePolicy	<xl:DeletePolicy>	<xl:Result>	Deletes a policy with a given policy ID	PAP

---

where XML namespaces are defined as follows:

Short Name	Full Name	Description
ctx	urn:oasis:names:tc:xacml:2.0:context:schema:os	XACML's 'context' namespace: used for defining authorization request

pol	urn:oasis:names:tc:xacml:2.0:policy:schema:os	XACML's 'policy' namespace: used for defining authorization policies
xl	http://gryb.info/schemas/xacml/common	XACMLight's 'common' namespace: used for defining XACMLight-specific types

5. Binary Installation.

It would be sufficient to copy the authz.aar file to $AXIS2_HOME/repository/services directory to make the service working. However, it's recommended to validate the deployment by running a test that is included to the installation. The following steps will conduct a shake-up test and validate the service:

1. Install Axis2 1.4-RC4 or later (e.g. from http://people.apache.org/~dims/axis2-1.4/RC4/axis2-1.4-RC4-bin.zip)
2. Setup $AXIS2_HOME environment variable to point to Axis2 home directory
3. Make sure that you have curl utility in your PATH - it's required for testing
4. Unpack the binary distribution (see "Source and Binary Locations").
5. cd xacmllight-<version>
6. cp info_gryb_xacml_config.xml $AXIS2_HOME -- This file contains XACML policy that would allow testing all mandatory function and many elements.
7. cp log4j.properties $AXIS2_HOME -- It will enable debug logging. You can change the location of log file by editing log4j.appender.AUTHZ.File property. By default the log file will be in 'current' directory, which differs depending on platform.
8. cp authz.aar $AXIS2_HOME/repository/services
9. Restart (or start) Axis2 server
10. cd test
11. chmod +x *.sh
12. Restart (or start) Axis2 server if you on Windows. On UNIX send.sh will restart the server.
13. send.sh authz1.xml (or send.bat authz1.xml on Windows) -- It will send a XACML request to PDP engine.
14. Verify the output XML: it should contain 10 'Permit' decisions.
15. For testing PAP engine use send.sh or test.pl scripts:

```
      send.sh <file> http://localhost:8080/axis2/services/PapService
<wsdl-operation>
      or
      perl test.pl <XACML-conformance-test-dir> 2>test.txt
      (see "Test Cases" for details)
```

1. For testing PAP engine use the following command:

6. Building from Sources.

XACMLight is a Maven2 [3] project and Maven2 must be installed before the service could be built. The building machine should have access to Internet because Maven will download dependency from online repositories. The following steps will allow to build the project:

1. Install Axis2 1.4-RC4 or later (e.g. from http://people.apache.org/~dims/axis2-1.4/RC4/axis2-1.4-RC4-bin.zip)
2. Setup $AXIS2_HOME environment variable to point to Axis2 home directory
3. Install Maven 2
4. Setup $M2_REPO environment variable to point to Maven repo directory
5. Make sure that 'curl' utility is in the PATH - it's required for testing
6. If you build on Windows platform make sure that 'sed', 'tar', 'gzip' utilities are available and are in the PATH
7. Download XACMLight sources (see "Source and Binary Locations") to your local directory (<xacmllight>)
8. cd <xacmllight>/bin
9. chmod +x *.sh
10. w2j.sh && build.sh && copy_aar.sh - it will build the authz.aar file and copy it to Axis2 repository
11. send.sh will send the authorization request to PDP and print out the result
12. On Windows platform use *.bat scripts instead of *.sh scripts

6.1. Useful Scripts.

The following useful scripts could be found in <xacmllight>/bin directory of source distribution. Please check the source to find out what the parameters and their default values are.

Script	Decsription	Purpose
build	Cleans and rebuilds everything including Eclipse [4] projects and settings	Build
clean	Cleans temporary files including files generated by w2j	Build
copy_aar	Builds and deploys authz.aar to $AXIS2_HOME	Build, Deploy
send	Restarts Axis2 service and sends a message (on Windows only sends without restart)	Test
simple_send	Sends a message without restarting Axis2 server	Test
test.pl	Runs tests from XACML 2.0 Conformance Suite [10]	Test
w2j	Generates proxy classes from WSDL	Build

6.2. Axis2 Limitations and Bugs.

The following problems have been found in Axis2 framework in the process of XACMLight implementation:

Problem	Decsription	Work around
axis2-wsdl2code plugin can not be used	It looks like this plugin generates ADB [5] classes only that don't handle XML abstract types well	Use wsdl2java with -d xmlbeans option instead
SystemTypeHolder.class	wsdl2java generates the class and puts it to src dir	The class needs to be moved to target dir. It's done by 'build' script
ADB classes can't be used	Substitution groups and abstract XML types implementations have bugs	Use xmlbeans
xmlbeans bug	Axis2 server creates 'XmlAnyTypeImpl' instead of XACML XSD type 'ExpressionType'	See XElementImpl.java for workaround
Hot deploy doesn't work	It looks like Axis2 needs to be restarted after each deploy	send script takes care of it
XmlObject.validate throws exceptions	validate throws exception` for valid XMLs	see Helper.validate
XmlObject.validate "no wildcards" error	Validate throws "no wildcards" error for 'anyAttribute'/lax elements	see Helper.validate

---

7. Source and Binary Locations.

7.1. Version 'SNAPSHOT' Locations.

'SNAPSHOT' is the latest XACMLight code that could be found in SVN [7] repository only at the following URL:
https://xacmllight.svn.sourceforge.net/svnroot/xacmllight/trunk

7.2. Other version locations.

The following versions are available: 2.0, 2.1, 2.2. The latest and recommended stable version is 2.2. The table below provides locations for SVN, source and binary distributions:

Distro Type	Location	Description
Binarie	http://downloads.sourceforge.net/xacmllight/xacmllight-	Binary

s	<ver>.tar.gz	tarball
Sources	http://downloads.sourceforge.net/xacmllight/xacmllight-src-<ver>.tar.gz	Source tarball
Sources	https://xacmllight.svn.sourceforge.net/svnroot/xacmllight/tags/<ver>	SVN [7] repository

* Replace <ver> with a version number to get a real URL.

8. Test Cases.

8.1. Internal Tests

The main test can be performed by calling send.sh authz1.xml (or send.bat authz1.xml) script from <xacmllight>/bin directory. It will send the authorization request for 10 resources to PDP engine. On UNIX/Linux it will also restart the Axis2 server. On Windows starting/stopping of Axis2 server should be done manually.

To test PAP/PDP engines use following commands:

```
send.sh <path-to-file> <service-url> <wsdl-operation>
```

The valid <path-to-file> and <wsdl-operation> values are provided in table below:

File	Wsdl Operation	Description	Service
authz1.xml	getDecision	Gets an authorization decision from PDP engine	PDP
policyset.set.xml	setPolicies	Overrides the root policy set	PAP
policyset.get.xml	getPolicies	Returns the current root policy set to a client	PAP
policy.set.xml	setPolicyy	Overrides a policy with a given policy ID	PAP
policy.get.xml	getPolicyy	Returns a policy with a given policy ID to a client	PAP
policy.del.xml	deletePolicy	Deletes a policy with a given policy ID	PAP
policy.add.xml	addPolicy	Adds a policy with a given policy ID to a policy set with a given policy set ID	PAP

The location of XML sample files:

1. Source distribution: **<xacmllight>/authz/src/main/resources**
2. Binary distribution: **<xacmllight>/test**

The URLs for PAP/PDP engines are provided below (assuming that Axis2 server runs on the local machine):

1. http://localhost:8080/axis2/services/PdpService - PDP engine
2. http://localhost:8080/axis2/services/PapService - PAP engine

---

8.2. XACML 2.0 Conformance Tests.

To test PAP/PDP engines against official XACML conformance tests [10] use the following command:

```
   perl test.pl  <XACML-conformance-test-dir> 2>test.txt
   where <XACML-conformance-test-dir> - a directory with conformance
tests
   text.tx - the results of tests
```

8.2.1 Mandatory Test Results.

The total number of tests was: 333, out of which 328 where successful, 5 were not applicable. The notes on some mandatory tests are provided below:

Test	Status	Note
IIA004	Not Applicable	XACMLight doesn't allow policy or policy set that is not compliant with XSD [8]
IID029.1-2	Not Applicable	XACMLight has only one root policy set or policy
IID030.1-2	Not Applicable	XACMLight has only one root policy set or policy
IIE001	Passed	The Policy document was changed to use XACMLight local repository (see info_gryb_xacml_config.xml for details)
IIE002	Passed	The Policy document was changed to use XACMLight local repository (see info_gryb_xacml_config.xml for details)
IIE003	Passed	The Policy document was changed to use XACMLight local repository (see info_gryb_xacml_config.xml for details)

8.2.2 Optional Test Results.

The total number of optional tests was: 44. All of them have been completed successfully using test.pl utility described above.

9. References.

1. Apache Axis2 - http://ws.apache.org/axis2

2. XACML 2.0 - http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf
3. Maven 2 - http://maven.apache.org/
4. Eclipse - http://www.eclipse.org/
5. Apache ADB - http://ws.apache.org/axis2/1_0/adb/adb-howto.html
6. Apache XMLBeans - http://xmlbeans.apache.org/
7. SVN (Subversion) - http://subversion.tigris.org/
8. XACML 2.0 Policy XSD - http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-cd-04.xsd
9. XACML 2.0 Context XSD - http://docs.oasis-open.org/xacml/access_control-xacml-2.0-context-schema-cd-04.xsd
10. XACML 2.0 Conformance Tests - http://www.oasis-open.org/committees/download.php/14846/xacml2.0-ct-v.0.4.zip

---

Endnotes

i
Based on the URL this document appears to be authored or edited by Oleg Gryb. This information was found for him at LinkedIn:

> **Security Architect/Staff Engineer at Intuit**
>
> Location San Francisco Bay Area
>
> Industry Information Technology and Services

His resume is available from his personal website “gryb.info/resume.”