# ACL in ARC1

Firstly, there is a XML schema about policy and request for ACL in ARC1. In general, this schema borrows some idea from the XACML schema, but tries to do some simplification in order to make the policy administration be easy, meanwhile keeps the expressive ability of XACML. At the same time, the implementation of the policy engine is also easy to extend to adapt XACML schema.
The implementation idea of policy engine is also simple and similar to the existing two java XACML implementations: Sun XACML (http://sunxacml.sourceforge.net), and Ax2e (http://axescon.com/ax2e/), which is to compare (using "function") the two "attribute"s separately from request and policy side, consider the combing "algorithm" between "rule"s inside "policy" as well.

**How to use policy engine?**

1. Firstly, each type of service should have different types of definition about <Subject>, <Action>, <Resource>. For example, a simple WS-wrapped file system which provides remote access, the <Subject> could be the X.509 DN, <Action> could be file operation like "read", "write" etc., the <Resource> could be URI of each file or directory (depends on the access control granularity). So if you would define access control policy for your specific service, you should extract the above kinds of definition from this type of service.
2. Then define the policy according to the arc1 policy schema: (see: http://svn.nordugrid.org/viewvc/nordugrid/arc1/trunk/src/hed/pdc/Policy.xsd?view=markup).
   The policy you defined can be in local file or remote http URL (maybe the xml database will be supported if needed).
3. In your code, you can use the "Evaluator" class, which will parse the configuration file to get the information about **policy** and dynamically-loaded classes, and then you need to feed the "Evaluator" object with your **request** (there is also an xml schema about request, see: http://svn.nordugrid.org/viewvc/nordugrid/arc1/trunk/src/hed/pdc/Request.xsd?view=markup).
   You can feed the evaluator object with an xml file, or an xml structure (XMLNode). For an example about how to use the policy engine, please check the code inside svn about the ArcPDP.cpp file (See http://svn.nordugrid.org/viewvc/nordugrid/arc1/trunk/src/hed/pdc/ArcPDP.cpp?view=markup.
   Here the "ArcPDP" is in charge of collecting some attributes from incoming message, composing these attributes into an xml structure with the Request.xsd format, and feeding it to policy engine (evaluator object).
4. The configuration about the policy engine is like: http://svn.nordugrid.org/viewvc/nordugrid/arc1/trunk/src/tests/echo/service.xml?view=markup

   Specify the policy, and some dynamically-loaded classes:

```
<pdp:PDPConfig>
    <pdp:PolicyStore name="test" location="Policy_Example.xml"/>
    <pdp:AttributeFactory name="attr.factory" />
    <pdp:CombingAlgorithmFactory name="alg.factory" />
    <pdp:FunctionFactory name="fn.factory" />
    <pdp:Evaluator name="arc.evaluator" />
    <pdp:Request name="arc.request" />
</pdp:PDPConfig>
```

   Configure the arc.pdp (which will use the policy engine) into a service:

```
<Service name="echo" id="echo">
    <SecHandler name="simplelist.authz" id="authz" event="incoming">
```

```
            <PDP name="simplelist.pdp" type="plaintxt" location="./accesslist"/>
            <PDP name="arc.pdp"/>
        </SecHandler>
        <next id="echo"/>
        <echo:prefix>[ </echo:prefix>
        <echo:suffix> ]</echo:suffix>
    </Service>
```

There is one thing needs to mention. In the above ArcPDP.cpp source file, attributes are collected and composed by the code. Then there is one problem, each types of service could need different sets of attributes, which means we need to implement different types of implementation for collecting attributes in the service side.

The other option is to let the requestor or some third-party collect the attributes and compose the request file, and then we don't need to collect attributes in the service side. There is an example service which can consume soap message (including request xml structure), see: http://svn.nordugrid.org/viewvc/nordugrid/arc1/trunk/src/services/pdp/

**How to develop customized classes?**

For those who would like to implement some customized classes, including policy or request parsing classes (in order to adapt different request or policy schema), "attribute factory" class, "combining algorithm factory" class, "function factory" class, "evaluator" class, etc., there is some interface which you can use to implement (see http://svn.nordugrid.org/viewvc/nordugrid/arc1/trunk/src/hed/libs/security/ArcPDP/).

```
    <pdp:PDPConfig>
        <pdp:PolicyStore name="test" location="Policy_Example.xml"/>
        <pdp:AttributeFactory name="attr.factory" />
        <pdp:CombingAlgorithmFactory name="alg.factory" />
        <pdp:FunctionFactory name="fn.factory" />
        <pdp:Evaluator name="arc.evaluator" />
        <pdp:Request name="arc.request" />
</pdp:PDPConfig>
```

**Comparison between ARC1 policy and XACML policy**

ARC1 policy schema is the simplification of XACML policy schema.

There are some characteristics of XACML which are taken way in ARC1 schema:

| XACML | ARC1 |
|---|---|
| PolicySet->Policy->Rule | The same |
| <Target/> element (includes <Subjects/> <Resources/> <Actions/>) | Replaced by <Subjects/> <Resources/> <Actions/> directly |
| <Target/> for <PolicySet/> <Policy/> <Rule/> | Only for <Rule/> |
| <Condition/> | Simplified, not <Apply/> element; and put together with <Subjects/> <Resources/> <Actions/> |
| <Obligation/> | No <Obligation/> |
| <AttributeDesignator > <AttributeSelector> | No <AttributeDesignator > <AttributeSelector>; Matching is based on local attributes and scanning result from request. |
| <SubjectMatch/>, <ResourceMatch/> | No <SubjectMatch/>, <ResourceMatch/>, |

| | |
|---|---|
| | match function is derived from “Function” attribute inside <Subject/> or <Resource/> |

**ARC1 Policy:**

```
<Policy xmlns="http://www.nordugrid.org/ws/schemas/policy-arc" PolicyId="sm-example:policy1"
CombiningAlg="Deny-Overrides">
    <Rule RuleId="rule1" Effect="Permit">
        <Subjects>
            <Subject Type="string"> /O=Grid/O=Test/CN=CA </Subject>
            <Subject Type="string">127.0.0.1 </Subject>
            <Subject Type="string">/vo.knowarc/usergroupA</Subject>
            <Subject>
                <Element Type="string">/O=Grid/OU=KnowARC/CN=XYZ</Element>
                <Element Type="string">urn:mace:shibboleth:examples</Element>
            </Subject>
            <GroupIdRef Location="./subjectgroup.xml">subgrpexample1</GroupIdRef>
        </Subjects>
        <Resources>
            <Resource Type="string">file://home/test</Resource>
        </Resources>
        <Actions Type="string">
            <Action>read</Action>
            <Action>stat</Action>
            <Action>list</Action>
        </Actions>
        <Conditions>
            <Condition Type="period“ Function=“time-in-range”>2007-09-10T20:30:20/P1Y1M</Condition>
            <GroupIdRef Location="./conditiongroup.xml">normalcondition</GroupIdRef>
        </Conditions>
    </Rule>
</Policy>
```

**XACML PolicySet:**

```
<PolicySet>
  <Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                                                urn:med:example:schema:records </AttributeValue>
          <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:target-names
pace" DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <PolicyIdReference>urn:oasis:names:tc:xacml:2.0:example:policyid:3</PolicyIdReference>
    <Policy                                    PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:2"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
      <Target/>
        <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:1" Effect="Permit"/>
        <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:2" Effect="Permit"/>
        <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:4" Effect="Deny"/>
    </Policy>
</PolicySet>
```

**XACML Policy:**

```
<Policy                                    PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:3"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Target/>
    <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:3" Effect="Permit">
      <Target>
        <Subjects/>
        <Resources/>
        <Actions/>
      </Target>
```

```
            <Condition>
               <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
                     <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:example: attribute:phy
sician-id" DataType="http://www.w3.org/2001/XMLSchema#string"/>
                  </Apply>
                  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
                     <AttributeSelector
RequestContextPath="//xacml-context:Resource/xacml-context:ResourceConten
t/md:record/md:primaryCarePhysician/md:registrationID/text()"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
                  </Apply>
               </Apply>
            </Condition>
      </Rule>
      <Obligations/>
</Policy>
```