[OpenAz] Version 84 - openaz - partial sunxacml integration

Rich.Levinson rich.levinson at oracle.com
Thu Mar 4 09:58:21 EST 2010

- Previous message: [OpenAz] Agenda for OpenAz Conference Call - Thursday March 4, 1 PM ET
- Next message: [OpenAz] Some thoughts on Mappers and PreDecisionHandler
- **Messages sorted by:** [date] [thread] [subject] [author]

```
  Hi All,

Version 84 is now current. There are some fixes that has the Request
side fully operational now for String and Date attributes and has been
tested with TestAzApi (low-level) and TestStyles (using pep api). Also I
added the sunxacml sample xml files that are used for running the above
and are in test/config, test/policy, and test/request. Otherwise there
are no new files and the request attributes are handled in a new
"internal impl" method:

    openaz.pdp.provider.SimpleConcreteSunXacmlService.getSunXacmlAttributeSet(AzEntity<T>)

which produces a Set of sunxacml Attributes, given an
openaz.azapi.AzEntity, which is a collection of AzAttributes. Note also
that this module is still in "hack mode" where I am experimenting w the
sunxacml capabilities to produce xml, which is not required for current
operations. Note also the sunxacml response side is still not
implemented, but should be in a couple of days.

    Rich

>> http://openaz.svn.sourceforge.net/viewvc/openaz/
>>
>> Last night I updated openaz w version 82 that begins the sunxacml
>> integration.
>>
>> It is going pretty smoothly and is about half-way thru the full cycle
>> of submitting a request and returning a response to an azapi client.
>> The integration strategy is proving to be more straight-forward than
>> I originally thought it would be. Essentially, the "integration" so
>> far is isolated to a single module in the pdp project, which uses the
>> sunxacml jar files in the pdp/lib project that Josh put there back at
>> the start of the project. The original dummy provider was called
>> SimpleConcreteService.
>>
>>     * This is still there but has been renamed to
>>       SimpleConcrete*Dummy*Service and
>>     * the sunxacml integration is done in a parallel module called
>>       SimpleConcrete*SunXacml*Service
>>
>> These modules implement AzService and contain the decide and query
>> methods.
>>
>> What has appeared to emerge as I have done this exercise is a fairly
>> simple conceptualization, which is as follows:
>>
>>     * The azapi interfaces primarily collect all the attributes each
>>       with their xacml metadata (attributeId, issuer, category,
>>       xacmlDataType, assignedJavaValue)
>>     * The "provider" simply operates on the underside of this
>>       interface, and all it needs to do is go thru all the attribute
>>       collections and pull the data it needs to submit the request in
>>       the format expected by the concrete PDP (in the case of
>>        sunxacml, this is a RequestCtx object with 4 sets of
>>        "Attributes" in the sunxacml format, so what you do in the
>>        decide method is go thru each azapi category, pull each
>>        AzAttribute and assign the appropriate data to the
>>        sunxacml.Attribute and AttributeValue and add each
>>        Attribute/AttributeValue to a Set, then create the RequestCtx w
>>        the 4 Sets as parameters)
>>      * The concrete PDP will then return a response, and then you do
>>        the reverse, which is to pull the data from the concrete
>>        response and put it into the AzResponseContext.
>>
>> That's it. All of the above, at least the first 2 bullets, is in
>> openaz.pdp.provider.SimpleConcreteSunXacmlService.(the response is
>> still tbd, but should be straight-forward as well and is expected to
>> go in the same module)
>>
>> A couple additional points:
>>
>>      * This should all work transparently with the azapi.pep package.
>>      * The sunxacml direct path is just Java objects, no xml, however,
>>        you can use the sunxacml impl to translate the azapi into a
>>        XACML XML Request, and it can also generate an XACML XML
>>        Response, so I think we will be able to use it w XML clients as
>>        well, possibly w some JavaObjectMapper that could take the xml
>>        input generate sunxacml RequestCtx and then use the RequestCtx
>>        to drive the building of the AzRequestContext and submit to any
>>        other AzApi front-ended PDP.
>>      * It is beginning to look like all "providers" will need to do is
>>        implement AzService and can use our own impls for all the
>>        pieces of the AzRequestContext and AzResponseContext, which
>>        should make uptake almost a trivial exercise once we have all
>>        the kinks worked out.
>>
>> A final observation is that what this is really all boiling down to
>> is having a "standard structure" which can contain all the attributes
>> w their xacml metadata. Everything else is just writing to and
>> reading from this structure.
>>
>>     Rich
>>
-------------- next part --------------
An HTML attachment was scrubbed...
URL: http://lists.openliberty.org/pipermail/openaz/attachments/20100304/0dad6695/attachment.html
```

- Previous message: [OpenAz] Agenda for OpenAz Conference Call - Thursday March 4, 1 PM ET
- Next message: [OpenAz] Some thoughts on Mappers and PreDecisionHandler
- **Messages sorted by:** [date] [thread] [subject] [author]

More information about the OpenAz mailing list