[OpenAz] Version 92: proposed azapi callback has been added

Rich.Levinson rich.levinson at oracle.com
Thu May 27 02:36:53 EDT 2010

- Previous message: [OpenAz] openAz conference call, May 27, 1PM Eastern
- Next message: [OpenAz] link to javadoc for v92
- **Messages sorted by:** [date] [thread] [subject] [author]

To OpenAz maillist:

Version 92 has been uploaded to the openaz project (code changes in v91, javadoc in v92).

This version has a proposed implementation of the azapi callback capabilities. The design takes into consideration the following:

* it is expected that callback at the azapi layer will be invoked by a callback from the pdp layer, and an example sunxacml callback was implemented to demo this, which revealed a number of concerns:
    - o the sunxacml pdp only has context pertaining to a single resource-action pair, so when the azapi callback is invoked it is necessary to identify exactly what resource or action or other AzEntity collection of attributes is the intended destination of the attribute that is being found.
    - o in order to make this happen effectively, what appears to be required is to have an identity attribute for each AzEntity, and when the sunxacml RequestCtx is submitted a means needs to be provided to include this info in the context. The proposed resolution to this is to have special Attributes added to the RequestCtx which identify the AzEntity objects, so that when the finder is invoked, that it can find those AzEntity objects in the AzRequestContext.
    - o this raises the question of how the sunxacml finder knows about the AzRequestContext. The proposed resolution to this is that when the azapi layer registers its sunxacml AttributeFinder, that it initialize the finder with a reference to a List of AzRequestContext objects, that will also have ids that can be referenced by the same mechanism as the AzEntity objects. i.e. when the sunxacml AttributeFinder is invoked, it will have a reference to the list of AzRequestContext objects from configuration, and the specific RequestCtx currently being processed will have attributes of pre-defined names in the subject-codebase category that identify the ids of the AzEntity objects and the AzRequestContext object that the finder needs.
* the signature of the azapi.AttributeFinder was determined to be the following to meet a variety of requirements:
    - o Collection<AzAttribute<T>> findAzAttribute(
      AzRequestContext azReqCtx,
      AzEntity<T> azEntity,
      AzAttribute<T> azAttribute)
* each of the 4 objects above will be described here:
    - o Collection<AzAttribute<T>>: what is returned is a Collection (unordered, duplicates allowed) of AzAttributes of a specific AzCategoryId T. For XACML, the basic paradigm is that a "bag" of AttributeValues can generally be processed by the Policy for any given Attribute in the Policy, so a Collection allows the finder to provide multiple values.
    - o AzRequestContext azReqCtx: the AzRequestContext is made available to the AzAttributeFinder so that it may use the information therein to assist it in obtaining the required attributes. This is consistent with the sunxacml implementation and appears to be a reasonable approach to providing the context getting additional information.

```
          o AzEntity<T> azEntity: the requested attribute from the PDP
            is being requested for a specific AzEntity. Because the
            sunxacml AttributeFinder was provided the AzEntity ids it is
            able to get the specific AzEntity to supply in the
            findAzAttribute call. It is important for the
            AzAttributeFinder to know which AzEntity the attribute is
            being requested for, because that may be a factor in the
            process of determining how to obtain the desired attribute.
          o AzAttribute<T> azAttribute: Finally, the AzAttribute that is
            passed, has a null value, but all the necessary metadata:
            AttributeId, Issuer, DataType, Category is in this null
            AzAttribute, which may be considered to be a template.

There were a few minor changes to the azapi interface to support this
functionality:

    * a new AzAttributeFinder.java interface was added that contains the
      above-described findAzAttribute method.
    * a getId() method was added to AzRequestContext to enable
      identification of the specific context needed
    * a setAzEntityId(String azEntityId) method was added to AzEntity,
      which actually is to catch up with the Obligation implementation
      which needs to be able to set this value, especially in PepApi
      handlers that work on the AzResponseContext. (this is unrelated to
      the callback functionality, and is just fixing a previous omission)
    * a very simple sample AzAttributeFinder has been provided as
      openaz.pep.provider.SimpleDummyAzAttributeFinder.java
    * a sample sunxacml AttributeFinder that invokes the
      AzAttributeFinder has been provided:
      openaz.pdp.resources.OpenAzSunXacmlAttributeFinderModule.java
    * finally, the test policy has been update to include a
      MustBePresent attribute that triggers all the callback machinery
      to be invoked, and the finders are hardcoded to provide a specific
      attribute in return to satisfy the policy. This limitation is easy
      to be removed, time permitting, but was provided to demonstrate
      the overall feasibility of the approach. Also a new method.
      testStyleMustBePresent, was added to TestStyles.java to drive the
      callback functionality.

This submission will be on the agenda for discussion at the 5/27 openaz
meeting.

    Thanks,
    Rich

-------------- next part --------------
An HTML attachment was scrubbed...
URL: http://lists.openliberty.org/pipermail/openaz/attachments/20100527/0712107a/attachment-0001.html
```

- Previous message: [OpenAz] openAz conference call, May 27, 1PM Eastern
- Next message: [OpenAz] link to javadoc for v92
- **Messages sorted by:** [date] [thread] [subject] [author]

More information about the OpenAz mailing list