Web Integration Service

# Advertising Security

With so much focus on security in Testbed 12, it seems only natural that the WIS should also discuss how to deal with this concept. Using the guidelines outlined in the engineering document OGC 15-022, we can easily advertise the security constraints in the WIS GetCapabilities document by using the appropriate OWS Common Constraint element in the GetCapabilities operation as outlined below:

 **<ows:Operation name="GetCapabilities">**

 **<ows:DCP>**

 **<ows:HTTP>**

 **<ows:Get xlink:href="http://ogc-testbed12.compusult.net/wes/wis?request=getCapabilities">**

 **<ows**:Constraint name="urn:ogc:def:tb12:security:authentication">

 <ows:ValuesReference ows:reference="http://tb12.opengis.net/security/authCodeList#HTTP\_BASIC">

 urn:ogc:def:tb12:ietf:2617:basic</ows:ValuesReference>

 </ows:Constraint>

 </ows:Get>

 </ows:HTTP>

 </ows:DCP>

 <ows:Parameter name="Format">

 <ows:AllowedValues>

 <ows:Value>application/xml</ows:Value>

 <ows:Value>application/x-bxml</ows:Value>

 </ows:AllowedValues>

 </ows:Parameter>

 </ows:Operation>

The general idea here is that service providers expose a public capabilities document. Then, within that document they advertise the methods that are available for users and/or clients to authenticate against. Any entity that authenticates successfully would then be given access to other services that would not be accessible otherwise.

In the sample above, the 'GetCapabilities' operation is advertising that users and/or clients can authenticate using 'Basic Authentication' but other methods are also available including: client certificate and user name token. A complete list of the available methods can be found here: http://tb12.opengis.net/security/authCodeList

# Storing Security information in an ebRIM CSW Catalogue.

When security information is advertised in a service's capabilities document it is very important that this information is captured properly so that it can be accessed and/or displayed later.

Typically, when a service capabilities document is published in a CSW catalogue it creates an ebRIM **Service** object that is then associated with various applicable objects within the ebRIM data model.

Because a service's associated security constraints are tied to its operations, it is recommended that a new ebRIM Association object with an association type called **urn:ogc:def:ebRIM-AssociationType:OGC:HasConstraint** be used to associate each ebRIM service binding to its associated constraints.

Each constraint object should be a new ExtrinsicObject with an object type = **urn:ogc:def:ebRIM-ObjectType:OGC:Constraint** that has an associated Slot called **AllowedValues** which contains the list of values permitted for the constraint. Figure ? shows what an ebRIM catalogue might look like after a WIS capabilities document has been successfully harvested.

**\*Show 'The ebRIM Service Security Association' screen shot.**

## Public and Private WIS Service objects

During a typical CSW publishing flow for a Web Integration Service (WIS) it is recommended that the CSW catalogue maintain which of the services listed in the **GetCapabilities** document have **Public** access and which ones have **Private** access. If you recall, the **Private** services are those returned by the service after the user and/or client has been authenticated.

It is recommended that this be accomplished by classifying each service properly using the new **Access Types** classification scheme.

The classification scheme **Access Types** is defined as:

**\*Show ‘ClassificationScheme\_AccessTypes’ screen shot.**

GetAssociations

# Storing the GetAssocations Response in an ebRIM CSW Catalogue.

To properly harvest the **GetAssocations** response it is important that each object returned either as a part of the source or the target also be harvested into the catalogue. This can accomplished in a number of ways but the following basic algorithm below should be used as a guideline.

* Harvest the service the containing the **GetAssociations** operation first as the generated UUIDs will be needed to create the proper ebRIM Association objects.
* For each **Association** returned from the **GetAssociations** response:
	+ Process the **Source** object:
		- Use the **Identifier** value and the generated ebRIM service id to retrieve the **Identifier's** ebRIM id.
	+ Process the **Target** object:
		- Use the **xlink:href** parameter to retrieve the associated ebRIM service id for the **Target** object in the ebRIM database.
			* If the service does not exist in the catalogue it will need to be harvested.
		- Use the **Identifier** value and **namespace** attribute to retrieve the **Identifier's** ebRIM id.
	+ Using the ebRIM id retrieved for the **Source**, the **Association's** **rel** attribute value and the ebRIM id retrieved for the **Target** object, create an ebRIM **Association** object.
* Once all associationshave been processed the generated ebRIM **Association** objects should submitted has part of an ebRIM transactiont document.

Figure ? presents a typical view of what the various ebRIM **Associations** might look like in an ebRIM database once all the appropriate services and other objects have been successfully harvested.

**\*Show 'The ebRIM Catalogue View' screen shot.**

Compusult WIS Implementation Details

# Web Integration Service Endpoint

The WIS endpoint consists of a WIS servlet endpoint, capabilities harvester, and database cache.

To allow flexibility in the content returned by the WIS, the GetCapabilities response is generated on demand from a list of services. The list of service endpoints known to the WIS is specified in an administrator-configurable property.

The WIS invokes a capabilities harvester to retrieves the capabilities for each known service. Metadata parsed from the capabilities responses are timestamped and cached in a database table. The timestamp is used by the WIS to decide whether to use the cached metadata or retrieve a fresh capabilities response.

The WIS cache also records security information for each service. For the purposes of Testbed 12, the security information simply indicates whether or not the service is exposed in the WIS public capabilities. This could be extended in the future to accommodate additional security restrictions.

# Security

Compusult's WIS endpoint has been implemented as a component of Web Enterprise Suite (WES). Access to the WIS endpoint is controlled by WES, requiring authentication via HTTP Basic Auth or PKI client certificate.

In addition to the regular service endpoint, the WIS implementation exposes a publicly-accessible capabilities URL that advertises security annotations as described by OGC 16-048 (OWS Common Security Extension ER).

An unresolved issue is how to reconcile security restrictions for the WIS, its contents, and the catalog. In an operational scenario, the WIS and catalog could have separate authentication credentials. Each service in a WIS could have its own security constraints. Catalog publishing clients will need to be aware of this and handle authentication for each service appropriately. Additionally, a catalog containing anything other than public data will need to enforce security restrictions on the published records.

Compusult's WIS implementation deals with these issues in the following manner:

* With respect to visibility, each service is treated as simply either public or non-public.
* When publishing to the catalog, the same authentication credentials are used for both the catalog and the WIS.
* The public capabilities endpoints are used when publishing services to the catalog.

Future Work

## Extended security for WIS content

Extend the security implementation to determine the visibility of service endpoints in the WIS capabilities based on user authorization. Propose solutions to reconcile authentication and authorization for the WIS, catalog and services.

## Approaches for filtering of WIS content

It may be useful for a content provider to expose subsets of services through the WIS. For example, an environmental data provider may wish to provide separate entry points for oceanographic and atmospheric data, along with an additional entry point to expose both data holdings.

A direct solution might be to stand up a separate WIS instance for each data holding. However, care must be taken to avoid collision between instances, while minimizing redundancy and resource usage.

It may be preferable to extend the specification so that a single instance can support subsets of services. A "passive" approach is for the WIS capabilities response to contain groupings or metadata attributes to categorize services. The client would be responsible for processing the response to identify the groupings. A more active approach is to implement a parameter or operation allowing a client to specify a simple category filter in the service request.

## Software Tools

For a service implementation that does more than simply returning a static capabilities response, software tools may aid in configuration of the service content. It may be useful to implement an interface to manage the list of service endpoints and the metadata cache, including the ability to add/remove endpoints, restrict access to content, and configure cache settings.

Tools to facilitate the creation of service associations may help speed the adoption of the Web Integration Service. A software library could provide developers with an easy means of implementing the GetAssociations operation. A configuration tool for this library could be used for data providers to quickly configure association metadata for their services.