EML - Bug #1132

fix access control rule ambiguities

08/15/2003 10:06 AM - Matt Jones

Status:	Resolved	**Start date:**	08/15/2003
Priority:	Immediate	**Due date:**	
Assignee:	Matt Jones	**% Done:**	0%
Category:	eml - general bugs	**Estimated time:**	0.00 hour
Target version:	EML2.1.0	**Spent time:**	0.00 hour
Bugzilla-Id:	1132		

Description

We found some issues that need to be discussed regarding access control in the EML2 specification. We have run into major problems while trying to implement the specified access control procedures for Metacat and suspect that these problems are not fixable without a change in the EML2 access control specification. Though we are having these problems within Metacat, we believe them to be general to any system that is trying to be EML2 compliant.

In EML2 there are two possible places where a processor may encounter access control: one is at the resource level and the other is at the additionalMetadata level. According to the EML spec, resource level access control applies to the whole document and additionalMetadata rules apply to a specific subtree for finer grained access control of EML subtrees. This allows one to have a general access policy and then make specific exceptions or changes for particular subtrees.

The problems arise when a processor must remove a controlled subtree and deliver it to the user. Once the user changes the document and resubmits it, the subtree that was removed must be put back in its valid and correct location.

1) Take this document for instance:
<a>
b
<d>d</d>

If a user has permission to write to the whole document (permission comes from top level access control) and doesn't have permission to read subtree d (restriction comes from addtionalmetadata access control) when he tries to download the document he will get part of the document like:
<a>
b

The user adds the elements c and e to the document.
<a>
b
<c>c</c>
<e>e</e>

Once the document is submitted back to the processor, the processor must figure out that element d (that was removed before) must fit in between c and e like so:
<a>
b
<c>c</c>
<d>d</d>
<e>e</e>

This may seem simple, but first of all, the only way to know where d is

supposed to go when you remove it is to store its parent id and its most immediate sibling(s) id(s). In this case d's parent is the same (a) but in the original document b was it's most immediate sibling. If d is inserted below b, the document becomes invalid. The only way to possibly know where d is allowed to be reinserted is to parse the schema which could still fail because element d could be legally allowed in many different locations (ie, it is not necessarily deterministic wrt node placement).

2) Nested subtrees also present a problem.

<c id="200">c</c>
<d>d</d>

An access module in additionalMetadata could specify that a user has read access to c but not a. If the processor simply returns c but not a or sub-elements (besides c) of a, the resulting document makes no sense. We need some sort of cascade rule that says that once read has been taken away for a node, none of its children can be made 'readable'.

3) Previously we stated that there are two palaces for access information to exist. This is actually not quite correct. In EML2 each of the four resource level modules (dataset, software, citation and protocol) have their own embedded access module. Even though a document has only one resource level module, the other resource level modules are embedded in each other. For example, you can have a citation within a dataset. That citation has its own access module. We have not defined in the EML spec how that is to be handled by a processor. Should the top-level resource access description take precedence? Probably. Should the lower level elements be ignored, or used in a manner similar to additionalMetadata? If the latter, to what do they apply, themselves, or their parent resource (unlike additionalMetadata, there is no describes element here to clarify the situtation)?

Proposed solution:

Changing EML at this late date is hugely problematic. We feel that we should maintain our commitment to make changes in EML backwards compatible (ie, EML 2.0.0 docs would be valid 2.0.1 docs). However, we feel that this is an important bug that compromises the usefulness of EML, and so fixing it now is the right thing to do. Nevertheless, we should minimize the disruptiveness of the change by 1) trying not to change the schema structure, and 2) redefining semantics of access control in a more tractable way.

We propose to alter EML to allow only two levels of access control. The first would be document wide control, accomplished by a new "access" element on the root "eml" document. The second would be data control for specific data files, accomplished by an optional "access" element in the physical distribution module that applies to the data object being described. We should remove access from the eml-resource module (now that it is in the eml module itself), although this would be an incompatible schema change. Alternatively we could simply define in the spec that access elements on the "resource" module are to be ignored. Restricting access control to the metadata and data respectively would greatly simplify the processing of EML, although it would limit the granularity of access control within the EML document.

Here's a fragment that shows what this new model might look like:

<eml>
...
<access>...</access> <-- defines overall access to
all metadata
...
<dataset>
<access>...</access> <-- this is ignored
<dataTable>

<physical>
<distribution>
<access>...</access> <-- defines access to the data object
in inline, online, or offline
elements (ie, not the metadata
itself, just the data)
<inline>...</inline>
</distribution>
</physical>
</dataTable>
</dataset>
</eml>

Of course, these changes would make an access element that is present in the schema (under dataset, for example) be ignored. Which is certainly confusing. We have to choose the lesser of two evils: 1) keep it and ignore it, which is confusing but allows schema compatibility with 2.0.0, or 2) delete it, which is clearer but makes all 2.0.0 documents that use it invalid and must be transformed to become valid EML 2.0.1 documents. This is a tough choice.

We also need to clarify how to interpret the values found in the 'permission' element, in that we should make it clear that 'changePermission' permission is needed to change an access block, not just 'write' permission. Currently the values we have (read, write, changePermission, all) are only tersely defined.

Comments or suggestions are welcome!
Jing, Chad, Matt, Dan, and Chris

Related issues:

Blocked by Metacat - Bug #968: Access control for eml2 documents	**Resolved**	**01/23/2003**
Blocked by Metacat - Bug #1674: Access control for eml-2.1.0 documents	**Resolved**	**09/10/2004**
Blocked by EML - Bug #3508: create a stylesheet for EML2.0.x to EML 2.1.0	**New**	**10/06/2008**

History

#1 - 05/10/2004 03:22 PM - Saurabh Garg

Notes taken based on the conversation on 6th May 2004:

1. Discussion on how Access should be handled in Metacat while reading eml 2.0.0
In eml 2.0.0 Access can be specified in following places.
//eml/dataset/access
//eml/citation/access
//eml/software/access
//eml/protocol/access
//eml/additionalMetadata

Permissions in //eml/dataset/access will be applied to all the metadata and by default to the data. Any permissions specified in //eml/citation/access, //eml/software/access and //eml/protocol/access are ignored. So all the metadata has same permissions overall. However, access rules can still be specified for any data attached to the metadata. This can be done in following manner:
Define a reference id in the <distribution> for which you want to define the access control. e.g.
<distribution id="xxyyzz">
.
.
</distribution>
To define the access rule for the above distribution, use //eml/additionalMetadata is following way:
<additionalMetadata>
<describes>
xxyyzz
</describes>
<access>
.
.
<access>
</additionalMetadata>

If additioncalMetadata describes as id which is not defined in the current eml document, then that additioncalMetadata is ignored.
If the id is defined in the current eml document, but is not in distribution tag, then also that additioncalMetadata is ignored.
If a distribution tag contains reference id but there is no additional Metadata entry for that reference id, then the rules specified in //eml/dataset/access are applied to this distribution.

2. Discussion on how Access should be handled in Metacat while reading eml 2.1.0
In eml 2.1.0 Access can be specified in following places.
//eml/access
//eml/..../distribution/access

Permissions in //eml/access will be applied to all the metadata and by default to the data. So all the metadata has same permissions overall. However, access rules can still be specified for any data attached to the metadata. This can be done using the /access that will now be added to the distribution tag. e.g.
<distribution>
<access>
.
<access>
</distribution>

If no access is specified in distribution then //eml/access rules are applied.

#2 - 06/25/2004 10:49 AM - Saurabh Garg

Correction in the last comment. Any permissions specified in //eml/citation/access, //eml/software/access and //eml/protocol/access are not ignored. Hence we consider access in 2.0.1 in following places:

//eml/dataset/access
//eml/citation/access
//eml/software/access
//eml/protocol/access
//eml/additionalMetadata (only for distribution element)

#3 - 06/25/2004 11:29 AM - Matt Jones

Actually, we should only consider the eml root element, so in EML 2.0.1 the only places for access would be:

/eml/dataset/access
/eml/citation/access
/eml/software/access
/eml/protocol/access
/eml/additionalMetadata (only for distribution element)

#4 - 06/25/2004 11:56 AM - Saurabh Garg

As both eml 2.0.1 and 2.1.0, only distribution ids will be referenced from access, I was wondering if it is possible to change/add a new type of id in distribution and only those kind of ids could be refered from data access. Any views on this?

#5 - 06/28/2004 05:22 PM - Jing Tao

I am thinking if we can get rid of reference in "access" module in eml2.1.0.
In eml2.1.0, the document for access will look like(The element name maybe wrong)
<eml><access>....</access>
<dataAccess>
<dataid>100</dataid>
<dataid>200</dataid>
<access>.....</access>
</dataAccess>
<dataAccess>
<dataid>300</dataid>
<dataid>400</dataid>
<access>.....</access>
</dataAccess>
</eml>

If the top level access module is a refernce to the access module in first dataAccess, that means they have same access control. And this is a equivalent access rules without reference:

<eml><access>real rules from access module from first dataAccess</access>.
<dataAccess>
<dataid>300</dataid>
<dataid>400</dataid>
<access>.....</access>
</dataAccess>
<eml>

If the access module in first dataAccess is a reference to top level access. It makes no sense because the access rules in first dataAccess doesn't give more rules for distribution than top access rule. The equivalent rules can be:
eml><access>...</access>.
<dataAccess>
<dataid>300</dataid>
<dataid>400</dataid>
<access>.....</access>
</dataAccess>
<eml>

If the access module in frist dataAccess is a refernce to second dataAccess, we can revise the rules to:
eml><access>...</access>.
<dataAccess>
<dataid>100</dataid>
<dataid>200</dataid>
<dataid>300</dataid>
<dataid>400</dataid>
<access>.....</access>
</dataAccess>
<eml>

So without reference, we still can make access module reusable(like the above description). And if we get rid of reference, it will make implementation simpler.

Any comments and suggestions are appreciated.

#6 - 07/09/2004 11:42 AM - Saurabh Garg

An issue that came up during implementation of access rules in metacat based on eml 2.0.1.

If a user is given read/write permission for inline data but no read/write permission for metadata, what should be sent back when he requests the eml document. So assuming we have the following eml document.

<eml>
<dataset> <- User NOT given permission to access
.
<inline>first</inline> <- User given permission to access
.
<inline>second</inline> <- User given permission to access
.
<inline>third</inline> <- User given permission to access
</dataset>
</eml>

So when the user tries to read the document, should metacat:
1. Reject the request
2. Send back inline data as following:
first
second
third
or maybe:
<inline>first</inline>
<inline>second</inline>
<inline>third</inline>

Second issue, when he tries to write back to the document, how should the data be parsed. For metacat to be able to write, there should be a defined schema for parsing and identifying different inline datas.
Another issue, once data is updated, the metadata would also need to be updated with new identifiers for data. But this cannot happen as user doesnt have write access for metadata. Hence the problem.

#7 - 07/15/2004 11:23 AM - David Blankman

It seems to me that read access to <inline> without read access to general metadata should not be allowed, Of what use is the ability to download data without being able to access (read) the metadata. The same thing is true for write aceess to <inline>. While it is possble that a change in data might not require a change in the metadata, how could someone make a change to an embedded portion or the metadata in isolation, that is, without having write access to the overall eml document.

I would suggest that <inline> could be more retrictive than other parts of the metadata, but not the reverse. That is, it is certainly possible that someone would want to grant broader access to overall metadata than to the data itself.

#8 - 09/02/2004 09:38 AM - Matt Jones

Changing QA contact to the list for all current EML bugs so that people can track what is happening.

#9 - 08/28/2008 04:51 PM - Margaret O'Brien

Text of a recent message on eml-dev, also here. This is a model that Chris and I came up with recently, for simplifying the access tree in EML.

In this model, the <access> tree appears only at the top level, and no longer under dataset, citation, software and protocol. The AccessRuleType now has a 0..many child, <describes>, for holding the id of the node that the rule applies to. If the <describes> is absent, then the rule applies to the whole document. An instance would look like this:

```
&lt;eml&gt;
   &lt;access authSystem="knb" order="allowFirst"&gt;
     &lt;allow&gt;
       &lt;describes&gt;table.1.1&lt;/describes&gt;
       &lt;principal ... &gt;
       &lt;permission ...&gt;
     &lt;/allow&gt;
     &lt;deny&gt;
       &lt;describes&gt;table.2.1&lt;/describes&gt;
       &lt;principal ... &gt;
       &lt;permission ...&gt;
     &lt;/deny&gt;
  &lt;/access&gt;
  &lt;dataset&gt;
           ...dataset markup...
  &lt;/dataset&gt;
&lt;/eml&gt;
```

We should encourage use of the order attribute (should it be required?) so that authors will be fully aware of the the rules they create. Rules should be applied in the order they appear (after what is dictated by the order attribute). Presumably, if no order attribute is included, then the rules are applied as they appear. Keeping the access tree in one area at the top of the document makes maintenance simpler, and the <describes> element acts as it does under <additionalMetadata>.

It would have to be decided if access rules should still be allowed in <additionalMetadata>. These could be 1) not recognized as EML access trees since node-level control can be described in eml/access, or 2) discouraged for the same reason, but applied, or 3) allowed and applied after the eml/access rules.

The model itself doesn't catch conflicting access rules, but it does simplify descriptions, making it easier for authors to see potential hang-ups. One way to control some basic conflicts might be to embed a rule-based schema, like Schematron. Conflict detection could also be added to the eml-parser.

Since the access tree is only available at the <eml> level, then this would end the use of dataset, citation, etc as root level elements for some purposes (e.g metacat) -- since metacat's default behavior is to allow access only to the logged-in owner if no access tree is present. In order to specify that a doc was publicly-readable, an author would have to wrap the dataset in <eml>...</eml> to include those access instructions.

Access is a major issue, and it would be good to get some discussion going. Here, we've only addressed simplifying the location of the access tree, it's another whole issue to deal with conflicts. It's always been slated for 2.1, but we may need to discuss that, too.

thanks -
Margaret and Chris

#10 - 08/28/2008 06:06 PM - Matt Jones

This solution, although nicely consolidated, fails to address the major problem that originally caused us such trouble and triggered the access control changes in the first place. Basically, the use of references to define access restrictions on arbitrary subtrees creates massive complexity when trying to figure out how people that have partial read and partial write capabilities on a document could succeed in modifying a document. I don't think using references to arbitrary subtrees is a viable solution, for the same reasons outlined at the start of this bug report. So, I still think the best solution is the one we proposed earlier in this thread:

<eml>
...
<access>...</access> <-- defines overall access to
all metadata
...
<dataset>
<dataTable>
<physical>
<distribution>
<access>...</access> <-- defines access to the data object
in inline, online, or offline
elements (ie, not the metadata
itself, just the data)
<inline>...</inline>
</distribution>
</physical>
</dataTable>
</dataset>
</eml>

This uses a top-level eml/access element for defining access to the whole metadata document, and it allows entity-specific access blocks in the distribution element for specifying rules for handling specific data entities. This does not allow access constraints on arbitrary subtrees, and therefore is tractable from an implementation perspective. Plus, it is really just setting separate ACLs for the metadata and data files, which is manageable in our current access table because each of these has its own docid.

Note that any access blocks in additionalMetadata would now be ignored, plus there is no longer an access element as a child of the eml-dataset module (or any other eml-resource extensions) because these rules are specified at a higher level in the /eml/access element.

This approach represents an incompatible schema change from EML 2.0.1 that would require document conversion to create valid instance of 2.1.0, but I think it is worth it given the gains in simplicity and clarity.

#11 - 08/29/2008 11:00 AM - Margaret O'Brien

Putting access control in only 2 areas will cover >90% of EML's access needs. There are other nodes that people have wanted to obfuscated somehow, but usually there are other methods of accomplishing this.

I am looking at the distribution and physical types, and have some discussion points to bring up there which don't concern access, and so will start a new bug!

#12 - 08/29/2008 03:21 PM - Margaret O'Brien

Code that bug #3480, comment2 is refering to. Access trees could be optionally included in any distribution element of this type. The <access> element follows the base type (res:DistributionType) as is required by schema language when creating derived types.

<xs:complexType name="PhysicalDistributionType">
<xs:complexContent>
<xs:extension base="res:DistributionType">
<xs:sequence>
<xs:element name="access" type="acc:AccessType" minOccurs="0"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

#13 - 11/06/2008 10:12 AM - Margaret O'Brien

fixed, long ago, but documentation of the fixed types is still to be done, see bug #3599

#14 - 11/13/2008 05:09 PM - Margaret O'Brien

The physicalDistributionType should have looked like this (below), so that if the references element is used, it is the only child of <distribution>. I wrongly used the sequence model, which meant that <references> could have had an <access> sibling. This has been checked into cvs as rev 1.75

<xs:complexType name="PhysicalDistributionType">
<xs:annotation>
<xs:appinfo>
<doc:tooltip>this type specifically for physical connections.</doc:tooltip>
</xs:appinfo>
</xs:annotation>
<xs:choice>
<xs:sequence>
<xs:choice>
<xs:element name="online" type="PhysicalOnlineType"/>
<xs:element name="offline" type="res:OfflineType"/>
<xs:element name="inline" type="res:InlineType"/>

```
</xs:choice>
<xs:element name="access" type="acc:AccessType" minOccurs="0"/>
</xs:sequence>
<xs:group ref="res:ReferencesGroup"/>
</xs:choice>
<xs:attribute name="id" type="res:IDType" use="optional"/>
<xs:attribute name="system" type="res:SystemType" use="optional"/>
<xs:attribute name="scope" type="res:ScopeType" use="optional" default="document"/>
</xs:complexType>
```

#16 - 03/27/2013 02:16 PM - Redmine Admin

Original Bugzilla ID was 1132

Files

eml_with_access_proposed.png	8.71 KB	08/28/2008	Margaret O'Brien
new_physical_distribution_type.png	6.64 KB	08/29/2008	Margaret O'Brien
transformation_access_additionalMetadata.doc	35 KB	12/03/2008	Margaret O'Brien