

## Equality in the MPEG REL

In making authorization decisions using MPEG REL licenses, an MPEG REL interpreter frequently must determine whether two MPEG REL elements are equal. To determine equality, the MPEG REL interpreter must determine whether the elements are logically equivalent, regardless of their physical representation.

Several XML and MPEG REL features (such as schema-defined default values, [inventory](#), [pattern matching](#), and so on) preclude testing for equality using a simple string comparison. Instead, the MPEG REL interpreter must perform more sophisticated processing to determine whether the one MPEG REL element conveys the same meaning as another MPEG REL element.

The following sections illustrate some of the challenges inherent in comparing MPEG REL elements for equality.

### Canonicalization

MPEG REL elements are defined in XML schemas, and those schemas may augment the information conveyed by the elements they define. For instance, an element's schema definition may specify default values for the element's attributes. For this reason, an MPEG REL interpreter comparing two MPEG REL elements for equality must evaluate each element according to its schema definition to determine the meaning that it conveys. Only then can the MPEG REL interpreter meaningfully compare the two elements and determine whether they are equal.

Canonicalization creates a representation of an XML element that can be used in meaningful equality comparisons. In canonicalization, all the syntactic variations permissible in XML documents are represented in a consistent manner. For instance, canonicalization can include the following processes:

- normalizing line breaks, white space, and attribute values
- consistently representing empty elements (for instance, always representing them as start-end tag pairs)
- adding default attributes to elements

These are, of course, only a few of the variations that canonicalization algorithms address. The [Schema Centric Canonicalization](#) algorithm canonicalizes XML instance with respect to all representational variations permitted by XML Schema. An MPEG REL interpreter can compare the output of Schema Centric Canonicalization for two MPEG REL elements and authoritatively determine whether the two elements are equal.

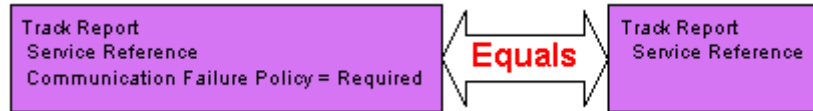
Unfortunately, implementing Schema Centric Canonicalization is very complex, and performing the full Schema Centric Canonicalization process on an MPEG REL element can be very time consuming. For this reason, the MPEG REL specification recommends an equality algorithm that more quickly addresses many common cases. For more information, refer to ISO/IEC FDIS 21000-5:2003(E), Annex C.

The following sections describe two of the syntactic issues addressed by the equality algorithm recommended in the MPEG REL specification.

## Substituting Schema Default Values

In the MPEG REL schema, some element attributes have default values. In this case, an element that omits the attribute would match an element that explicitly specifies the default value for that attribute. For this reason, an MPEG REL interpreter may need to assume the default value for element attributes that have been omitted in a license.

For example, an MPEG REL interpreter may need to compare two trackReport conditions, only one of which specifies the policy to follow in the event of a communication failure:



As you can see in this diagram, these two trackReport conditions match, because the schema specifies that the communicationFailurePolicy attribute's default value is "required". To correctly match two trackReport conditions, the MPEG REL interpreter must assume a value of "required" whenever a communicationFailurePolicy attribute is omitted.

The following MPEG REL fragments illustrate these two trackReport conditions. In this example and all following examples, the prefix `sx:` refers to the MPEG REL standard extension. All other elements are defined in the MPEG REL Core. For information on how these schema namespaces are declared in the license file, see the page that describes [licenses](#).

### Track Report With the Communication Failure Policy Specified

```
<sx:trackReport>
  <serviceReference>
    <sx:uddi>
      <sx:serviceKey>
        <sx:uuid>D04951E4-332C-4693-B7DB-D3D1D1C20844</sx:uuid>
      </sx:serviceKey>
    </sx:uddi>
  </serviceReference>
  <sx:communicationFailurePolicy>required</sx:communicationFailurePolicy>
</sx:trackReport>
```

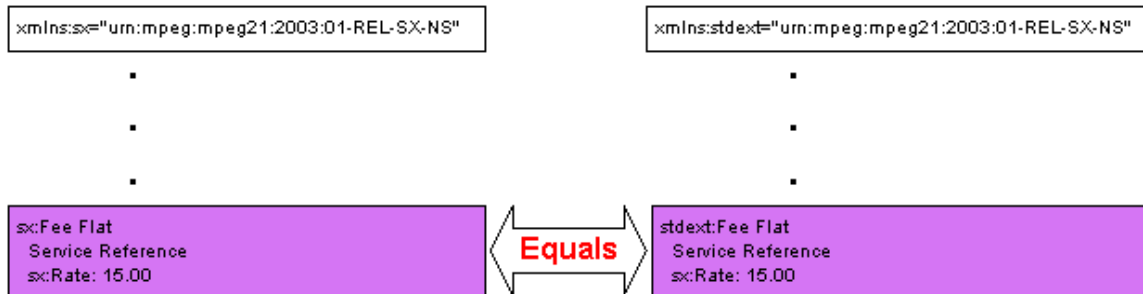
### Track Report Without the Communication Failure Policy Specified

```
<sx:trackReport>
  <serviceReference>
    <sx:uddi>
      <sx:serviceKey>
        <sx:uuid>D04951E4-332C-4693-B7DB-D3D1D1C20844</sx:uuid>
      </sx:serviceKey>
    </sx:uddi>
  </serviceReference>
</sx:trackReport>
```

## Resolving XML Namespace References

Each MPEG REL license must [declare the prefixes and XML namespaces](#) it uses. The license may use any prefix to refer to a particular namespace. If two licenses use different prefixes to refer to the same namespace, a simple string comparison would not match elements from those licenses, even if those elements convey the same meaning. For this reason, an MPEG REL interpreter must resolve the namespace references before comparing elements for equality.

For example, an MPEG REL interpreter may need to compare two fee conditions that use different prefixes to refer to the MPEG REL standard extension:



As you can see in this diagram, the fee using the prefix `sx:` matches the fee using the prefix `stdext:`, because both prefixes are defined to reference the same extension namespace. The following MPEG REL fragments illustrate this example:

```
License Using sx: to Refer to the MPEG REL Standard Extension

<license xmlns:sx="urn:mpeg:mpeg21:2003:01-REL-SX-NS"
.
.
.
  <sx:feeFlat>
    <serviceReference>
      <sx:uddi>
        <sx:serviceKey>
          <sx:uuid>D04951E4-332C-4693-B7DB-D3D1D1C20844</sx:uuid>
        </sx:serviceKey>
      </sx:uddi>
    </serviceReference>
    <sx:rate>15.00</sx:rate>
  </sx:feeFlat>
</license>
```

## License Using stdext: to Refer to the MPEG REL Standard Extension

```
<license xmlns:stdext="urn:mpeg:mpeg21:2003:01-REL-SX-NS"
.
.
.
  <stdext:feeFlat>
    <serviceReference>
      <stdext:uddi>
        <stdext:serviceKey>
          <stdext:uuid>D04951E4-332C-4693-B7DB-
D3D1D1C20844</stdext:uuid>
        </stdext:serviceKey>
      </stdext:uddi>
    </serviceReference>
    <stdext:rate>15.00</stdext:rate>
  </stdext:feeFlat>
</license>
```

### Addressing Inventory and Variable References

In addition to canonicalization issues, an MPEG REL interpreter must address the following issues relating to MPEG REL-specific features:

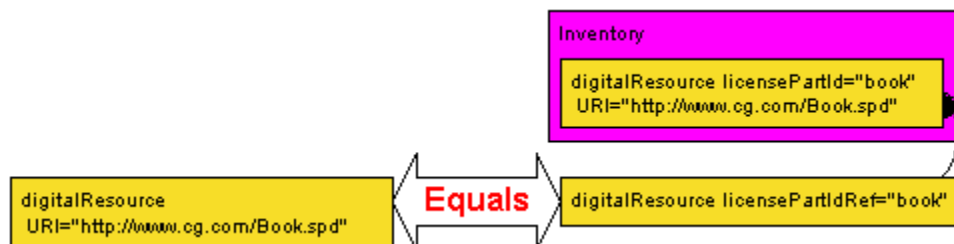
- references to the [inventory](#)
- [license part identifiers](#)
- references to [variables](#)

These issues are described in detail below.

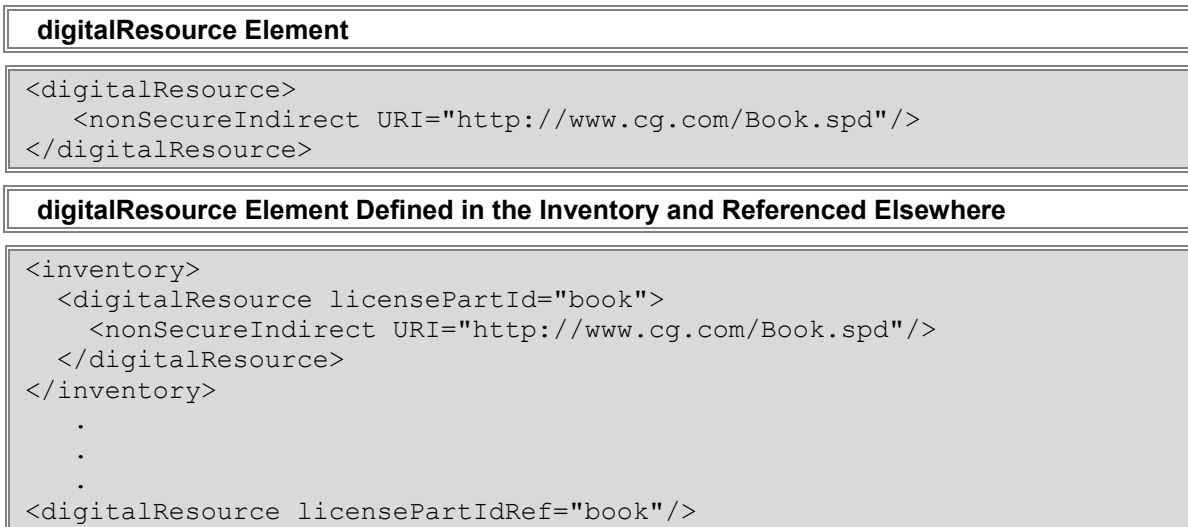
### Resolving References to the Inventory

The [inventory](#) is simply a container that enables you to define commonly-used license parts (such as principals, rights, resources, and conditions) in one place and reference them in another. Since elements defined in the inventory are simply referenced within the grants or grantGroups in a license, these references must be resolved before those grants and grantGroups can be tested for equality.

For example, an MPEG REL interpreter may need to compare two digital resource elements, one of which is defined in a license [inventory](#):



In this diagram, it is obvious that these two digital resources reference the same URI. However, looking at the MPEG REL representation of these elements, it is clear that a simple string comparison will not find these elements equal. The following examples illustrate how this information is represented in the MPEG REL:

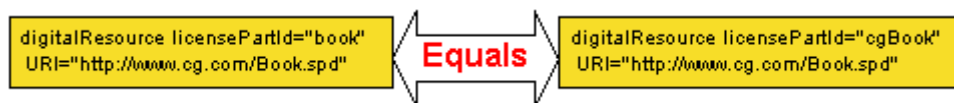


Before MPEG REL elements can be compared for equality, the MPEG REL interpreter must substitute any MPEG REL definitions in the [inventory](#) into the locations where those definitions are referenced.

### Removing License Part Identifiers

All license parts have an optional [license part ID](#) attribute. Typically, this identifier is used in conjunction with the [inventory](#) feature to provide a mechanism for referencing inventory elements elsewhere in a license, as described [above](#). However, the definition of any license part may include a license part ID.

Since these identifiers are included as a syntactic convenience feature, they do not add any semantic meaning to MPEG REL elements. Two elements that are logically equivalent can have two different [license part IDs](#), as shown in the following illustration:

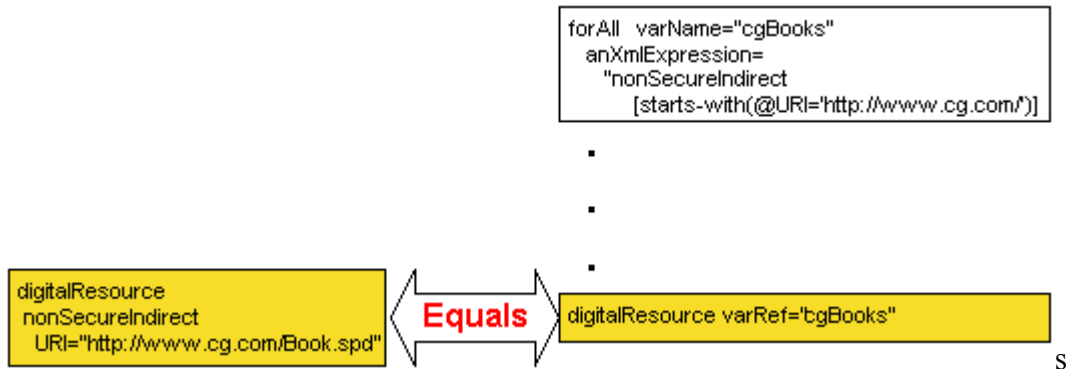


As you can see in this illustration, both digital resource elements refer to the same resource. They only differ in the license part ID assigned to them. For this reason, all license part IDs must be removed before comparing two MPEG REL elements for equality.

## Resolving Variables

Using [variables](#), you can use [pattern matching](#) to identify a set of principals, rights, resources, or conditions within a single grant or grantGroup. To use variables, you define the variable at the beginning of a grant or grantGroup and then simply reference the variable in place of a specific principal, right, resource, or condition. Before comparing for equality, any variables defined outside of the scope of the elements being compared should be resolved if possible.

For example, an MPEG REL interpreter may need to compare a specific digital resource to a digital resource defined using a [variable](#):



As you can see in this diagram, the [variable](#) *cgBooks* identifies all resources at the domain `http://www.cg.com`. To compare the two digital resource elements for equality, the MPEG REL interpreter must:

1. Resolve the reference to the *cgBooks* [variable](#).
2. Determine whether the digital resource identified by the URI `"http://www.cg.com/book.spd"` matches the [pattern](#) specified in the [variable](#) definition.

Once these two steps are complete, the two elements are equal. The following MPEG REL fragments illustrate this example:

digitalResource Element
<pre>&lt;digitalResource&gt;   &lt;nonSecureIndirect URI="http://www.cg.com/Book.spd"/&gt; &lt;/digitalResource&gt;</pre>
digitalResource Element Defined Using a Variable
<pre>&lt;forall&gt;   &lt;digitalResource varName="cgBook"&gt;     &lt;anXmlAttribute&gt;nonSecureIndirect [starts- with(@URI='http://www.cg.com/') ]   &lt;/anXmlAttribute&gt; &lt;/forall&gt; . . . &lt;digitalResource varRef="cgBook"/&gt;</pre>

## Implementation Considerations

As mentioned above, the MPEG REL specification recommends an equality algorithm that addresses many common cases. For more information, refer to ISO/IEC FDIS 21000-5:2003(E), Annex C. In addition to this information, you may wish to keep the following considerations in mind.

For greatest efficiency, you may wish to implement equality comparison in conjunction with [variable](#) reference resolution. In that case, you would probably want to handle the various aspects of equality comparison in the following order:

1. Resolve references to elements defined in the [inventory](#), and remove any [license part ID](#) references. Making this substitution is fairly easy, and it will address some of the equality comparison cases.
2. Write a custom equality function that traverses the MPEG REL element hierarchy and compares the elements in it. While traversing the hierarchy, this function can evaluate all [variable](#) references to determine whether they can be resolved within the scope of the current element or whether they must be evaluated in the context of containing elements.

Copyright © 2002, 2004 ContentGuard Holdings, Inc. ContentGuard is a registered trademark of ContentGuard Holdings, Inc.