DSCOMPOSITE-MODEL and Schemas

DS-COMPOSITE datastream

Content Models may contain this reserved datastream. It lists the datastreams that must exist in subscribing data objects, and the few requirements on them.

DSCompositeSchema

The old schema for the datastream can be seen below.

```
<xsd:schema
        targetNamespace="info:fedora/fedora-system:def/dsCompositeModel#"
        xmlns="info:fedora/fedora-system:def/dsCompositeModel#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">
    <xsd:element name="dsCompositeModel">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element minOccurs="0" maxOccurs="unbounded" ref="dsTypeModel"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="dsTypeModel">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element minOccurs="0" maxOccurs="unbounded" ref="form"/>
            </xsd:sequence>
            <xsd:attribute name="ID" use="required" type="xsd:NCName"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="form">
        <xsd:complexType>
            <xsd:attribute name="FORMAT_URI" use="optional" type="xsd:anyURI"/>
            <xsd:attribute name="MIME" use="optional"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

The content of a DS-COMPOSITE datastream could look like this

```
<dsCompositeModel
        xmlns="info:fedora/fedora-system:def/dsCompositeModel#">
    <dsTypeModel ID="DC">
        <form MIME="text/xml"/>
    </dsTypeModel>
    <dsTypeModel ID="ORIGIN">
        <form MIME="text/xml"/>
    </dsTypeModel>
</dsCompositeModel>
```

Optional datastreams

First, we will ammend the schema to support the property "optional" to datastream declarations. The JIRA issue for this is http://fedora-commons.org/jira/browse/FCREPO-531

The semantic meaning of optional is:
1. The datastream can exist in the substribing objects but does not have to
2. If the datastream exist, it must adhere to the specification in the content model

This will allow you to express optional datastreams like this

```
<dsCompositeModel
        xmlns="info:fedora/fedora-system:def/dsCompositeModel#">
    <dsTypeModel ID="DC">
        <form MIME="text/xml"/>
    </dsTypeModel>
    <dsTypeModel ID="ORIGIN" optional="true">
        <form MIME="text/xml"/>
    </dsTypeModel>
</dsCompositeModel>
```

To allow this, the schema is now

```
<xsd:schema
        targetNamespace="info:fedora/fedora-system:def/dsCompositeModel#"
        xmlns="info:fedora/fedora-system:def/dsCompositeModel#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">
    <xsd:element name="dsCompositeModel">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element minOccurs="0" maxOccurs="unbounded" ref="dsTypeModel"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="dsTypeModel">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element minOccurs="0" maxOccurs="unbounded" ref="form"/>
            </xsd:sequence>
            <xsd:attribute name="ID" use="required" type="xsd:NCName"/>
            <xsd:attribute name="optional" use="optional" type="xsd:boolean" default="false"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="form">
        <xsd:complexType>
            <xsd:attribute name="FORMAT_URI" use="optional" type="xsd:anyURI"/>
            <xsd:attribute name="MIME" use="optional"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Allowing extensions in DS-COMPOSITE

Since Fedora already use DS-COMPOSITE to declare the existence of datastreams, it is the natural location to specify restrictions on the contents of datastreams. Unfortunately, the schema for the DS-COMPOSITE datastream does not allow for any extra content, To that effect, we have made a small change to the schema.

```
<xsd:schema
        targetNamespace="info:fedora/fedora-system:def/dsCompositeModel#"
        xmlns="info:fedora/fedora-system:def/dsCompositeModel#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">
    <xsd:element name="dsCompositeModel">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element minOccurs="0" maxOccurs="unbounded" ref="dsTypeModel"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="dsTypeModel">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element minOccurs="0" maxOccurs="unbounded" ref="form"/>
                <xsd:element minOccurs="0" maxOccurs="unbounded" ref="extension"/>
            </xsd:sequence>
            <xsd:attribute name="ID" use="required"/>
            <xsd:attribute name="optional" use="optional" type="xsd:boolean" default="false"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="extension">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:any namespace="##any" processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="name" use="required"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="form">
        <xsd:complexType>
            <xsd:attribute name="FORMAT_URI" use="optional" type="xsd:anyURI"/>
            <xsd:attribute name="MIME" use="optional"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="reference">
        <xsd:complexType>
            <xsd:attribute name="type"/>
            <xsd:attribute name="value"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

With this changed schema, the contents could look like this:

```
<dsCompositeModel
        xmlns="info:fedora/fedora-system:def/dsCompositeModel#">

    <dsTypeModel ID="DC">
        <form MIME="text/xml"/>
        <extension name="SCHEMA">

        </extension>
    </dsTypeModel>
    <dsTypeModel ID="ORIGIN">
        <form MIME="text/xml"/>
        <extension name="SCHEMA">

        </extension>
    </dsTypeModel>
</dsCompositeModel>
```

What is noteworthy here is that the <dsTypeModel> and the <form> elements are left unchanged. The Fedora code, working with DS-COMPOSITE only looks for these tags, so the new schema will not cause conflicts, and the extensions will be quietly ignored. This is exactly as we want, this change should not make our objects incompatible with an older Fedora.

Reserved extensions in DS-COMPOSITE - the SCHEMA extension

We have reserved one extention name.

- "SCHEMA" - used to specify that the datastream content must adhere to a schema

That leaves the question on where to place the content (schema). Currently, three ways are supported.
Embedding it directly in DS-COMPOSITE makes for a very unreadable datastream. Alternatively, you could just specify an URL to the schema, but this approach have problems too. Having the Content Model depend on schemas defined elsewhere, perhaps on remote servers, mean that the content models could break by actions totally unrelated to the repository. The best way, we have found, is to embed the schema in a datastream in the content model.

```
<dsCompositeModel
        xmlns="info:fedora/fedora-system:def/dsCompositeModel#"
        xmlns:schema="http://ecm.sourceforge.net/types/dscompositeschema/0/1/#">

    <!-- The DC datastream is declared. It's mime type must be text/xml. It must adhere to the xml schema
residing in on the specified URL.
    <dsTypeModel ID="DC">
        <form MIME="text/xml"/>
        <extension name="SCHEMA">
            <reference type="url" value="http://www.openarchives.org/OAI/2.0/oai_dc.xsd"/>
        </extension>
    </dsTypeModel>


    <!-- The PBCORE datastream is declared. It's mime type must be text/xml. It must adhere to the xml schema
residing in the PDCORE_SCHEMA datastream in this content model-->
    <dsTypeModel ID="PBCORE">
        <form MIME="text/xml"/>
        <extension name="SCHEMA">
            <reference type="datastream" value="PDCORE_SCHEMA"/>
        </extension>
    </dsTypeModel>

    <!-- The ORIGIN datastream is declared. It's mime type must be text/xml. It must adhere to the xml schema
inlined here
    <dsTypeModel ID="ORIGIN">
        <form MIME="text/xml"/>
        <extension name="SCHEMA">
            <schema targetNamespace="originNamespace" xmlns="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
                <element name="origin" type="string"/>
            </schema>
        </extension>
    </dsTypeModel>

</dsCompositeModel>
```

XML schema has a few problems. The most apparent problem here is that one schema can only declare a single namespace. In order to have elements from several namespaces, other schemas must be imported. These cannot be inlined in the schema, they must be residing on some other uri. In effect, this means that we can put one schema in a datastream, but it might still import schemas from remote locations. So, without being able to tell the schema to import from other datastreams, this would serve little purpose. For this purpose, we have defined the $THIS$ keyword, meaning "this object". It is used like this

```
<schema targetNamespace="http://www.openarchives.org/OAI/2.0/oai_dc/"
        xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">

    <annotation>
        <documentation>
            XML Schema 2002-03-18 by Pete Johnston.
            Adjusted for usage in the OAI-PMH.
            Schema imports the Dublin Core elements from the DCMI schema for unqualified Dublin Core.
            2002-12-19 updated to use simpledc20021212.xsd (instead of simpledc20020312.xsd)
        </documentation>
    </annotation>

    <import namespace="http://purl.org/dc/elements/1.1/"
            schemaLocation="$THIS$/SIMPLEDC-SCHEMA"/>

    <element name="dc" type="oai_dc:oai_dcType"/>

    <complexType name="oai_dcType">
        <choice minOccurs="0" maxOccurs="unbounded">
            <element ref="dc:title"/>
            <element ref="dc:creator"/>
            <element ref="dc:subject"/>
            <element ref="dc:description"/>
            <element ref="dc:publisher"/>
            <element ref="dc:contributor"/>
            <element ref="dc:date"/>
            <element ref="dc:type"/>
            <element ref="dc:format"/>
            <element ref="dc:identifier"/>
            <element ref="dc:source"/>
            <element ref="dc:language"/>
            <element ref="dc:relation"/>
            <element ref="dc:coverage"/>
            <element ref="dc:rights"/>
        </choice>
    </complexType>

</schema>
```

Notice that in the import tag, the schema location is given as "$THIS$/SIMPLEDC-SCHEMA". This means that the schema location is the residing in the SIMPLEDC-SCHEMA datastream in this content model. This functionality only works in schemas, and only when validating with the built-in validate method. It works both for schemas inlined in DS-COMPOSITE-MODEL, and stored in separate datastreams. You can have multiple schemas imported, in a hirachical structure, but the $THIS$ will always refer to the object containing the DS-COMPOSITE-MODEL in question.

This setup would not work, or rather, it would attemp the get the SCHEMA2 datastream from content model 1.

```
Content model 1
  * DS-COMPOSITE-MODEL
    * SOMESTREAM
      * URL to http://localhost/fedora/objects/demo:contentmodel2/SOMESTREAM-SCHEMA

Content model 2
  * SOMESTREAM-SCHEMA
     * xml schema
       * import $THIS$/SCHEMA2
  * SCHEMA2
     * xml schema
```