BLIS-XML Methodology for Transforming EXPRESS data Models to XDR

BLIS Project Companies

November 2000

Overview

BLIS-XML is a methodology for transforming EXPRESS data models into 'XML Schema' (initially XML Data Reduced (XDR)). BLIS-XML can be used with any EXPRESS schema, or a subset of any EXPRESS schema.

BLIS-XML is a simple, straightforward methodology, based on consideration of what information software tools need from the source schema. The transformation is done such that data can be converted from Part21 files to BLIS-XML files and back to Part21 files without loss of information. This is possible, because the source EXPRESS schema is known and EXPRESS tools can map the information in BLIS-XML files to the full EXPRESS model using information in the source EXPRESS schema.

Currently, BLIS-XML schemas are defined using XDR, which defines XML schema information that can be used to validate data instance files. In the future, we will use the newer schema definition syntax just promoted to "Candidate Recommendation" by W3C, which uses the file extension XSD (see http://www.w3.org/XML/Schema).

BLIS-XML schemas are open content schemas. This means BLIS-XML files may contain more information (e.g. additional entities and attributes) than specified in the schema. Tools that do not understand the additional information should ignore that information.

Data transformation between EXPRESS and BLIS-XML for a specific Entity/Element set (schema) is straightforward, given XML and EXPRESS components with sufficient Application Programmer Interfaces (APIs). Such a converter is generic to any EXPRESS schema and can be used for any data structured according to the given schema. For example, during import of BLIS-XML data, this converter reads the BLIS-XML data file into the XML component (e.g. Microsoft's msxml.dll), iterates through the XML data and uses the API of the EXPRESS component to create the model inside that EXPRESS component. At this point, the data model looks exactly the same as when the EXPRESS component loads the equivalent Part21 file. The client application using the EXPRESS component sees no difference; there is no need to write custom code in the client application for supporting BLIS-XML.

Using BLIS-XML with XML components (e.g. msxml.dll) alone is not quite as simple. It can be done, but) the logic of the EXPRESS model must be implemented, as it has been in components that load/use the EXPRESS schema.
Another significant advantage to leveraging existing EXPRESS based components is that client applications can support import/export of both Part21 and BLIS-XML data

files, using the same code. Use of XML based components alone, will only support BLIS-XML data files.

The first BLIS-XML schema published by the BLIS project was for the BLIS project “views” of the IFC R2.0 model. This schema is called “BLIS-XML for IFC R2.0”. Examples in this document are taken from that schema.

Specifications

The BLIS-XML tag

A BLIS-XML data set is bounded by the following BLIS-XML tag.

```
<BLIS-XML schema="name_of_the_schema">
...
</BLIS-XML>
```

This provides a standard container for BLIS-XML data, even if that data is embedded in a larger XML data set (or file).

The BLIS-XML schema used to structure the data is identified by the “Schema” - attribute. This can be used to validate the data. It is not the same as the source EXPRESS schema name (which can be found in the header)

Header

The header replicates the header section of Part21 files. Unlike Part21 the header section can be expanded to contain other relevant information. The header used in “BLIS-XML for IFC R2” is as follows:

```
<ElementType name ="FILE_HEADER_SECTION" content="eltOnly" order="seq">
 <element type="FILE_DESCRIPTION" />
 <element type="FILE_NAME" />
 <element type="FILE_SCHEMA" />
</ElementType>

<ElementType name ="FILE_DESCRIPTION" content="eltOnly">
  <element type="description" minOccurs="1" maxOccurs="*"/>
  <attribute type="implemenetaion_level" required="no"/>
</ElementType>

<ElementType name="FILE_NAME" content="eltOnly" >
 <element type="author" minOccurs="1" maxOccurs="*"/>
 <element type="organization" minOccurs="1" maxOccurs="*"/>
 <attribute type="name" required="yes"/>
 <attribute type="time_stamp" required="no"/>
 <attribute type="preprocessor_version" required="no"/>
 <attribute type="originating_system" required="no"/>
 <attribute type="authorization" required="no"/>
</ElementType>

<ElementType name ="FILE_SCHEMA" content="eltOnly" >
 <element type="schema_identifiers" minOccurs="1" maxOccurs="*"/>
</ElementType>

<AttributeType name="implemenetaion_level" dt:type="string"/>
<AttributeType name="name" dt:type="string"/>
<AttributeType name="time_stamp" dt:type="string"/>
<AttributeType name="preprocessor_version" dt:type="string"/>
<AttributeType name="originating_system" dt:type="string"/>
<AttributeType name="authorization" dt:type="string"/>

<ElementType name ="description" content="empty">
 <attribute type="StringValue" required="yes"/>
</ElementType>

<ElementType name ="author" content="empty">
 <attribute type="StringValue" required="yes"/>
</ElementType>

<ElementType name ="organization" content="empty">
 <attribute type="StringValue" required="yes"/>
</ElementType>

<ElementType name ="schema_identifiers" content="empty">
 <attribute type="StringValue" required="yes"/>
</ElementType>
```

Elements

Entities in the EXPRESS model map to elements in the XML model.

```
<ElementType name="IfcBuilding" content="empty">
  ...
</ElementType>
```

BLIS-XML results in a ‘flat’ schema where the inheritance defined in the EXPRESS model has been removed because XDR does not support inheritance. As a result each element in the BLIS-XML schema that inherits attributes in the EXPRESS schema has a copy of the attributes it inherits. This BLIS-XML tool set creates redundant information in the schema files, but this does not affect the data files in any way. Even if XDR supported inheritance the data files would be exactly the same.

Although BLIS-XML methodology and tool set can generate a 'flat' copy of all inherited attributes defined in the EXPRESS schema, they do not require that all attributes be copied. In some cases, leaf node objects inherit attributes from superclasses that are not relevant to the leaf node. The BLIS-XML methodology allows omission of such irrelevant attributes in the BLIS-XML schema, so long as they are defined as `OPTIONAL` in the EXPRESS schema.

Attributes

There are two alternatives in transforming EXPRESS attributes to XML. The first is to use XML attributes. The second is to use XML 'sub elements.' BLIS-XML uses both techniques. Attributes are used for simple data types. Sub elements are used for `LIST` and `SET` attributes.

There is couple of special exceptions to these general rules. The first is that a `LIST` or `SET` of object references is transformed to `idrefs.` The second exception is for `SELECT` type values (see **SELECT types** below).

```
<ElementType name="IfcAddress" content="eltOnly">
  <element type="AddressLines" minOccurs="0" maxOccurs="*"></element>
  ...
  <attribute type="WWWHomePageURL" required="no"></attribute>
</ElementType>
```

It would be easier to manipulate BLIS-XML files with XSL scripts if all attributes were modeled as sub elements. However, since the object references in a BLIS-XML files use the `idref` mechanism (see **References** below), use of XSL scripts is already very limited. In general, manipulating BLIS-XML files will require a programming language. When using a programming language to manipulate an XML Document Object Model (DOM), it makes no difference if EXPRESS attributes are modeled as XML attributes or XML sub elements. Since XML attributes are more compact, the BLIS group has opted to use them for simple attributes.

Overriding attributes

Usually attributes in BLIS-XML schema definitions are defined globally. This means that there is one definition, with a unique name, in the BLIS-XML schema. This works well so long as EXPRESS attribute names are used consistently (i.e. all attributes with the same name share the same data type and semantic meaning). However, this consistency is not required, nor enforced in EXPRESS schemas. In situations where two EXPRESS attributes have different data types or semantic meanings, the BLIS-XML schema will override global attributes locally. A local override is defined by putting the local definition of the attribute inside the `ElementType` tag.

```
<!--  Global definition of the "PredefinedType" attribute -->
<AttributeType name="PredefinedType"
               dt:type="enumeration"
               dt:values="Office Restroom Storage ...">
</AttributeType>


<!--  Local override of the "PredefinedType" attribute -->
<ElementType name="IfcPlumbingFixture" content="empty">
  ...
  <AttributeType name="PredefinedType"
                 dt:type="enumeration"
                 dt:values="Faucet Sink Toilet Urinal Shower ...">
  </AttributeType>
  <attribute type="PredefinedType" required="yes"></attribute>
 </ElementType>
```

INVERSE attributes

INVERSE attributes are used in EXPRESS schemas as back-pointers in object references. The references can be queried on attribute level asking – which objects have references to this object from an attribute called e.g. ‘xyz’. EXPRESS-based tools have methods for querying these attributes, but this is not directly possible with XML tools since XDR does not support this ‘back-pointer’ concept. This is a challenges, since many EXPRESS models, such as IFC, rely heavily on the use of INVERSE attributes.

While INVERSE attributes are not included in Part21 files, the BLIS-XML methodology supports their addition to the XML schema and data files. Inverse attributes in the XML schema are exactly like other attributes. When BLIS-XML data is imported by EXPRESS based tools the INVERSE attributes are ignored. When BLIS-XML data is imported by XML based tools the INVERSE attributes can be used as they are in EXPRESS tool sets.

```
<ElementType name="IfcWall" content="empty">
  ...
  <attribute type="IsDefinedBy" required="no"></attribute>
  ...
</ElementType>
```

Data types

BLIS-XML does not yet support all data types that could be supported by XML, e.g. all Real values use r8 (r4 is not used). This is because in BLIS-XML the data types have been reduced to the basic data types used in programming languages such as C++ and VB.

The following table shows the data type conversions used in BLIS-XML

Express	**BLIS-XML**
STRING	string
NUMBER	number
INTEGER	int
REAL	r8
LOGICAL	boolean
BOOLEAN	boolean
BINARY	bin.hex

References

XML models are best suited to represent simple hierarchical structures where one thing is clearly contained by another thing (e.g. an address contains a phone number). EXPRESS models generally support a richer set of relationships. It is simply not possible to represent a typical EXPRESS model using a single XML hierarchy.

Therefore, BLIS-XML schemas are not structured according to any single relationship in the source EXPRESS schema. Instead, BLIS-XML uses the `id/idref/idrefs` construct in XML to encode the references (relationships) between objects. Similar to the line number in Part21 files, each entity has a `XMLID` attribute of type `ID`. This `XMLID` can be any string value that is legal for a token of type `ID`.

For example, the XMLIDs are assigned as a following rule:

A prefix letter “`i`” + sequence number e.g. “`i1`”, “`i2`”,…”`i100`”.

```
<AttributeType name="XMLID" dt:type="id"></AttributeType>

<ElementType name="IfcBuilding" content="empty">
  <attribute type="XMLID" required="yes"></attribute>
  ...
</ElementType>
```

References to elements are made using attributes of type `idref` or `idrefs` and the `XMLID` of the element that is being referenced. The XMLIDs in the attribute of type `idrefs` will be appeared like "`i1 i2 i3`". A white space is inserted between XMLIDs.

```
<AttributeType name="RelatingObject" dt:type="idref"></AttributeType>
<AttributeType name="RelatedObjects" dt:type="idrefs"></AttributeType>
```

EXPRESS models are strongly typed, which makes it possible to define exactly the type of objects that can be referenced by another object. XML has virtually no typing in this sense and an `idref` or `idrefs` can legally point to any entity type in the model. There is currently no way to constrain this in XDR. Therefore, it is also a limitation in BLIS-XML. We have been careful to specify the legal references in the documentation. If BLIS-XML data is imported into an EXPRESS based component. That component can validate the references. XML based components will be unable to do this automatically.

As a result of this transformation, BLIS-XML files are almost completely 'flat'. They don't contain the typical hierarchical tag structure of XML files, the structure is much closer to the one used in Part21 files.

Enumerations

Enumerations in EXPRESS map directly to the `enumeration` data type in XML.

```
<AttributeType name="ProfileType"
               dt:type="enumeration"
               dt:values="Curve Area">
</AttributeType>
```

SELECT types

Select types must be transformed in BLIS-XML so that enough information is preserved; to enable EXPRESS based tools to recreate the model. To enable validation of select types in XML, they are modeled using the `group` construct. This means that the select type attribute in EXPRESS will be appeared as a child element of the entity element which contains the select type attribute. The child element which is from select type attribute contains an element which includes the `group` construct of select types.

If SELECT types are nested in the EXPRESS schema also the BLIS-XML entities representing these SELECT types shall be nested.

These rules apply only to SELECT values that point to defined types. Object references created through a SELECT type use the standard `idref` mechanism. This is similar to the system used in Part21 exchange files.

```
<ElementType name="IfcMeasureValue" content="eltOnly">
 <group order="one">
  <element type="IfcAreaMeasure"/>
  <element type="IfcBoolean"/>
  <element type="IfcInteger"/>
  <element type="IfcString"/>

 </group>
</ElementType>

<ElementType name="IfcString" content="empty">
 <attribute type="StringValue"></attribute>
</ElementType>
```

The resulting BLIS-XML data file has the following format:

```
<IfcSimpleProperty XMLID="i9068" Name="FireRating">
 <ValueComponent>
  <IfcMeasureValue>
   <IfcString StringValue="60 minute"/>
  </IfcMeasureValue>
 </ValueComponent>
</IfcSimpleProperty>
```

Acknowledgements

- Adachi, Yoshinobu - Secom Co. Ltd. (Japan)
- Hietanen, Jiri - qPartners Oy (Finland)

- Bazjanac, Vladimir - Lawrence Berkeley National Laboratory (USA)
- Drogemuller, Robin - CSIRO (Australia)
- Gowri, Krishnan - Pacific Northwest National Laboratories (USA)
- Houbaux, Patrick – CSTB (France)
- See, Richard - Microsoft Corporation (USA)
- Yu, Kevin – Timberline Software (USA)