



RivCom Limited  
Lotmead Business Village  
Wanborough, Swindon  
Wiltshire, SN4 0UY. UK  
Tel +44 (0) 1793 792000  
Fax +44 (0) 1793 792001  
[www.rivcom.com](http://www.rivcom.com)

## XML and EXPRESS as schema definition languages

*Daniel Rivers-Moore, Director of New Technologies, RivCom, and Joint Project Leader of the ISO Preliminary Work Item on SGML and Industrial Data*

### **Abstract**

EXPRESS is a rich and mature language for the definition of data schemas. It is part of the STEP standard (ISO 10303), and is in widespread use to define data models for large-scale industrial applications, including manufacturing, engineering, defence, oil rigs, processing plants, etc. As part of the ongoing STEP/SGML harmonization activity under ISO TC184/SC4, a New Work Item proposal has just been put forward under the title "An XML representation of EXPRESS-driven data". The scope of the work item includes the use of XML to encode both the EXPRESS schema definition, and the instance data that conforms to that schema. This presentation will outline the main semantic features of the EXPRESS language, and show the current state of work on defining an XML representation of those semantics. The relationship of this work to other ongoing efforts to develop an XML syntax for schema definitions will be discussed.

### **EXPRESS and SGML compared**

EXPRESS and SGML have many features in common, as well as many significant differences.

Both are ISO standards. Both provide system-neutral syntaxes for the definition of abstract data structures. Both were designed for use in a specific arena, and both are sufficiently generic to be of considerable value outside the arena for which they were originally developed.

SGML's home ground is the world of documents. It was developed at a time when documents were almost exclusively produced in hard copy, and many of its features derive from the fact that it is optimized for document-centric applications. However, it has proved extremely versatile and extensible to other arenas. HyTime was developed as an application of SGML to the world of music and multimedia, but has proved invaluable in highly demanding industrial strength documentation applications for major manufacturing industries. HTML was developed to provide a document-like user interface to information accessible via the Internet, and has resulted in the transformation of the Internet into the World Wide Web, which has radically changed the way individuals and companies are doing business and interacting with each other on a global scale. XML has been developed to free the World Wide Web from the presentation-centric shackles of HTML, and to bring the more powerful abstractions of SGML to the Web world. This gives rise to major new opportunities, and also some significant challenges, as the data structures that will be exchanged using XML will require a machine-interpretable schema definition syntax that goes beyond what can currently be defined using a DTD.



EXPRESS is the data description language of STEP, the 'Standard for the Exchange of Product model data'. Its home ground is the world of computer aided design (CAD) systems. It is used to describe 3-dimensional geometry of solid parts, and configurations and assemblies of those parts into machines of all kinds, from automobiles to aircraft, to oil rigs, ships and power plants. Its intent was to free CAD data from dependence on proprietary computer systems and formats, and thus to enable data describing manufactured products to be exchanged between systems during the design and manufacturing process. But as with SGML, its usefulness has led to it needing to meet challenges beyond those originally envisaged. As well as the shape and configuration of a product, its design requirements, operating instructions, maintenance history and use within a business also need to be described. For large or expensive manufactured product, lifecycle information can be at least as important as the description of its physical configuration. And the information may need to be shared in real time between multiple users in a networked environment, not just exchanged periodically as a snapshot, which was the original intent of STEP.

### ***The STEP/SGML harmonization initiative***

The need for effective management, sharing and exchange of product information throughout the lifecycle of a large product such as an aircraft or a power plant has given rise to a recognition of the need to bring together the SGML and STEP worlds. Product documentation needs to reference product data, and be dynamically modified as that data changes over time. And the documentation is itself part of the product and needs to be managed as such. Out of this realization, the STEP/SGML harmonization effort was born.

STEP/SGML harmonization is officially endorsed as a Preliminary Work Item under ISO TC184/SC4/WG10, the STEP data architecture working group. We have heard from Eliot Kimber about the central part of this activity, which involves the development of a property set for EXPRESS, and EXPRESS models of SGML and HyTime, to bring the two worlds together into a seamless whole. We have also heard from David Price, who is a member of ISO TC184/SC4/WG10, about another of WG10's activities, which is the development of a new architecture for data integration, that should allow harmonisation of disparate models representing equivalent information.

Another important part of the STEP/SGML harmonisation activity is an ISO New Work Item proposal on the development of an XML representation of EXPRESS-driven data. The intent of this work is to enable the representation of an EXPRESS data model, or schema, and a set of data conforming to it, in XML syntax.

### ***What EXPRESS and XML can bring to each other***

The strengths and weaknesses of XML and EXPRESS mirror each other in interesting ways. EXPRESS is a rich and powerful language for the definition of the structure to which a set of data should conform. The mechanisms it provides for defining the types of object and their properties, that will be used in a given data set, and the constraints to which those objects should conform, are far richer than those provided by the DTD syntax of SGML and XML. On the other hand, XML instance syntax is far richer than the corresponding syntax used to transmit instance data conforming to an EXPRESS schema.

The so-called 'Part 21 physical file' used to exchange sets of data conforming to an EXPRESS schema has severe limitations, and is not easily interpretable by humans, or by machines that do not have prior knowledge of the schema itself. A Part 21 file without the schema is useless. In contrast, as we well know, an XML file without a DTD can be extremely useful in its own right.



Yet, despite the inherent readability of an XML instance, it remains true that to understand its meaning, some kind of explicit or implicit data schema is required. The very human-readability of XML files can mask this fact. A human reader who understands English (or whatever language has been used for the names of the element types and attributes used in the document) can guess at an implicit data schema that was in the mind of the information designer who chose those element type or attribute names. To make such an XML document usable by an automated system, and interoperable with XML files using other element type or attribute names with equivalent or related meanings, an explicit formal specification of the data schema is required. And as XML is used for the encoding of non-document-centric information, there is absolutely no guarantee that DTD syntax will provide the semantic richness necessary for the encoding of that schema definition. The constraints on the data that a DTD can express are quite limited, in ways that reflects the DTD's origins in a document-centric world. There are no mechanisms in the DTD to define even such vital things as supertype-subtype relationships between different classes of data object, let alone to define completely new data types, or constraints on element content or attribute values appropriate to particular data types.

These are some of the reasons that lie behind initiatives as XML-Data and RDF-schema. But they have the difficult task of starting from a blank sheet (or almost). The EXPRESS language provides us with an invaluable starting point for an approach to defining an XML schema definition language that draws on years of experience, and heavy testing in real-world industrial applications. If we can define XML constructs to represent the full semantic richness of an EXPRESS schema, we shall acquire the capability of enriching our XML data in ways that go at least as far, if not further, than those that the XML-Data and RDF-schema initiatives are setting out to achieve. In addition, we shall enable STEP data to be exchanged and shared over the Web and made accessible through XML-aware tools. XLL links will be able to be defined between STEP data objects, and XSL stylesheets applied to STEP content to make it navigable within Web browsers.

By bringing the strengths of both worlds together, each will gain significantly.

### ***The EXPRESS language***

In the space of this presentation, it is not possible to give a full picture of the EXPRESS language. However, we can give a few examples to show how, from simple underlying concepts, extremely complex data structures can be specified.

The following is a partial list of the declarations that the EXPRESS allows. After each example, a possible XML representation of the equivalent information is put forward. It should be stressed that this is not the only possible way of representing the EXPRESS semantics in XML. What is shown here represents work in progress. Comments, feedback and suggestions of alternative approaches or syntactic constructs are most welcome.

- **type declarations** define new data types in terms of a previously declared type, or one of the underlying data types defined by the EXPRESS language specification itself

#### *Example*

```
TYPE person_name = STRING
END_TYPE
```

#### *Possible XML representation*

```
<datatype name="person_name">
  <STRING/>
</datatype>
```

- **entity declarations** define the types of data object that can exist in a population (Note that an EXPRESS 'entity' corresponds to an SGML *element type*, and has nothing to do with the notion of an entity as SGML defines it.)

#### *Example 1*

```
ENTITY point;
  x_coord, y_coord : REAL
END_ENTITY
```



This example shows that a point has two attributes, `x_coord` and `y_coord`, whose values are both real numbers.

*Possible XML representation*

```
<entity name="point">
  <attribute name="x_coord">
    <REAL/>
  </attribute>
  <attribute name="y_coord">
    <REAL/>
  </attribute>
</entity>
```

*Example 2*

```
ENTITY circle:
  centre : point
  radius : REAL
WHERE radius_non_negative :
  radius >= 0;
END_ENTITY
```

This example shows that a circle has a centre attribute, whose value is a point, and a radius attribute, whose value is a non-negative real number. Here we see that attributes can have values that are themselves objects whose type is defined elsewhere in the schema (or, as we shall see in a moment, in another schema). We also see how rules can be specified to define constraints on the values that attributes may have.

*Possible XML representation*

Here we use some of the constructs of MathML to represent aspects of the expression language used in EXPRESS rules and functions.

```
<entity name="circle">
  <attribute name="centre">
    <type href="#point"/>
  </attribute>
  <attribute name="radius">
    <REAL/>
  </attribute>
  <where-rule name="radius_non_negative">
    <MathML:reln>
      <MathML:geq/>
      <attval href="radius"/>
      <NUMBER>0</NUMBER>
    </MathML:reln>
  </where-rule>
</entity>
```

- **schema declarations** provide a name for a coherent set of entity and type declarations, thereby enabling entities and data types from one schema to be invoked by another (rather as XML namespace declarations will allow element types from different DTDs to be brought together in a single document instance or data set)

*Example*

```
SCHEMA geometry;
...
END_SCHEMA
SCHEMA map;
  USE FROM geometry (point AS map_reference);
...
END_SCHEMA;
```



*Possible XML representation*

```

<schema name="geometry">
  ...
</schema>
<schema name="map">
  <usefrom href="#geometry" entity="point"
    alias="map_reference"/>
  ...
</schema>

```

- **constant declarations** allow specific data objects to be reused by name

*Example*

```

CONSTANT
  thousand : NUMBER := 1000;
  million : NUMBER := thousand**2;
  origin : point := point(0.0, 0.0);
END_CONSTANT;

```

*Possible XML representation*

```

<constant name="thousand">
  <NUMBER>1000</NUMBER>
</constant>
<constant name="million">
  <NUMBER>
    <MathML:apply>
      <MathML:power/>
      <NUMBER href="#thousand"/>
      <NUMBER>2</NUMBER>
    </MathML:apply>
  </NUMBER>
</constant>
<constant name="origin">
  <type href="#point">
    <attval attribute="x_coord">0.0</attval>
    <attval attribute="y_coord">0.0</attval>
  </constant>

```

- **function declarations** allow algorithms to be defined to return a single result, of a specified data type, based on manipulation of the values of one or more data objects

*Example*

```

FUNCTION distance (p1, p2 : point) : REAL;
  RETURN (SQRT((p1.x_coord - p2.x_coord)**2 + (p1.y_coord -
    p2.y_coord)**2));
END_FUNCTION;

```

*Possible XML representation*

```

<function name="distance">
  <REAL/>
  <parameter name="p1">
    <type href="#point">
  </parameter>
  <parameter name="p2">
    <type href="#point">
  </parameter>
  <return>
    <MathML:apply>
      <MathML:power/>
      <MathML:apply>
        <MathML:plus/>
        <MathML:apply>

```



```

    <MathML:power/>
    <MathML:apply>
      <MathML:minus/>
      <attval href="#p1" att="x_coord"/>
      <attval href="#p2" att="x_coord"/>
    </MathML:apply>
    <NUMBER>2</NUMBER>
  </MathML:apply>
  <MathML:apply>
    <MathML:power/>
    <MathML:apply>
      <MathML:minus/>
      <attval href="#p1" att="y_coord"/>
      <attval href="#p2" att="y_coord"/>
    </MathML:apply>
    <NUMBER>2</NUMBER>
  </MathML:apply>
</MathML:apply>
</MathML:apply>
<NUMBER>0.5</NUMBER>
</MathML:apply>
</return>
</function>

```

- **subtype declarations** allow entities to be declared to be subtypes of other entities, together with a specification of the additional constraint or constraints that determine whether a given instance of the supertype is an instance of the subtype.

#### Example

```

ENTITY line_segment;
  line_start, line_end : point;
END_ENTITY;
ENTITY long_line_segment;
  SUBTYPE OF (line_segment);
  WHERE longer_than_1000 :
    distance(line_start, line_end) > thousand;
END_ENTITY;

```

Notice the use of a function (distance) and a constant (thousand) within the specification of this constraint rule.

#### Possible XML representation

```

<entity name="line_segment">
  <attribute name="line_start">
    <type href="#point"/>
  </attribute>
  <attribute name="line_end">
    <type href="#point"/>
  </attribute>
</entity>
<entity name="long_line_segment">
  <where-rule name="longer_than_1000">
    <MathML:reln>
      <MathML:gt/>
      <MathML:apply>
        <function href="#distance"/>
        <attval href="line_start"/>
        <attval href="line_end"/>
      </MathML:apply>
      <constval href="#thousand"/>
    </MathML:reln>
  </where-rule>

```



```

</entity>
<typehierarchy>
  <supertype href="#line_segment"/>
  <subtype href="#long_line_segment"/>
</typehierarchy>

```

- **supertype declarations** allow complex supertype-subtype graphs to be defined. In the example which follows, we shall see that male, female, citizen and alien are all subtypes of person. We also see that every person must be either male or female, *and* either a citizen or an alien. We also see that an entity can have multiple supertypes. Here, resident\_alien is a subtype of alien and of taxpayer (though alien itself is not a subtype of taxpayer).

#### Example

```

ENTITY male
  SUBTYPE OF (person);
END_ENTITY;
ENTITY female
  SUBTYPE OF (person);
END_ENTITY;
ENTITY citizen
  SUBTYPE OF (person);
END_ENTITY;
ENTITY alien
  SUBTYPE OF (person);
END_ENTITY;
ENTITY resident_alien
  SUBTYPE OF (alien,taxpayer);
END_ENTITY
ENTITY organization
  SUBTYPE OF (taxpayer);
END_ENTITY;
ENTITY person
  SUPERTYPE OF (ONE OF(male,female) AND
                ONE OF(citizen,alien));
END_ENTITY;
ENTITY taxpayer
  SUPERTYPE OF (ONE OF
  (citizen,resident_alien,organization));
END_ENTITY

```

#### Possible XML representation

```

<entity name="person"/>
<entity name="organization"/>
<entity name="male"/>
<entity name="female"/>
<entity name="alien"/>
<entity name="resident_alien"/>
<entity name="citizen"/>
<entity name="taxpayer"/>
<typehierarchy>
  <supertype href="#alien"/>
  <subtype href="#resident_alien"/>
</typehierarchy>
<typehierarchy>
  <supertype href="#taxpayer"/>
  <subtype href="#oneof1"/>
</typehierarchy>
<orgroup ID="oneof1">
  <subtype href="#citizen"/>
  <subtype href="#resident_alien"/>
  <subtype href="#organization"/>
</orgroup>

```



```

<typehierarchy>
  <supertype href="#person"/>
  <subtype href="#and1"/>
</typehierarchy>
<andgroup ID="and1">
  <subtype href="#oneof2"/>
  <subtype href="#oneof3"/>
</andgroup>
<orgroup ID="oneof2">
  <subtype href="#male"/>
  <subtype href="#female"/>
</orgroup>
<orgroup ID="oneof3">
  <subtype href="#alien"/>
  <subtype href="#citizen"/>
</orgroup>

```

- **rule declarations** allow constraints to be defined on an entire population of data objects

#### Example

```

RULE at_least_three_large_circles FOR (circle);
LOCAL
  large_circles : SET OF circle := []
END_LOCAL
large_circles := QUERY(c <* circle | (c.radius > 100);
WHERE
  SIZEOF(large_circles) >= 3;
END_RULE;

```

This rule declares a local variable whose data type is a set of circles. It then defines the initial value of this variable to be the empty set. Next, it populates the set with all those circles whose radius is greater than 100, and finally states that the size of the resultant set must be greater than or equal to three.

#### Possible XML representation

```

<rule name="at_least_three_large_circles" href="#circle"
lang="EXPRESS">
  <![CDATA[
    LOCAL
      large : SET OF circle := [];
    END_LOCAL;
    large_circles := QUERY(c <* circle | (c.radius > 100);
    WHERE
      SIZEOF(large_circles) >= 3;
  ]]>
</rule>

```

The STEP/SGML harmonization group has not as yet defined an XML representation of the EXPRESS local variable and query constructs. Therefore, in this example we have used the alternative mechanism of declaring an "EXPRESS" language attribute on the rule, with a CDATA marked section to wrap the EXPRESS code without modification. It will be of interest to see whether any mechanisms emerge from the XSL or XLL Working Group that will be of relevance here. One could imagine local variables being defined within the context of the pattern-recognition constructs that will determine the criteria for triggering an XSL style rule. Likewise, one could imagine various query mechanisms being used to identify the anchors of an XLL link. If these Working Groups do propose mechanisms that can be used in the representation of EXPRESS's local variable and query constructs, there would be obvious benefit in using the same techniques. If not, we shall propose our own approach in order to complete the work of rendering EXPRESS into XML.



## ***Current status of the XML representation of EXPRESS-driven data***

An initial demonstration of many of the constructs of the EXPRESS language in XML syntax was given at the STEP meetings in Bad Aibling, Germany in June 1998. Both instance data and schema definition were shown to be expressible in XML, and different XML-aware tools were applied to the resultant data. This included presentation and navigation of the information using DSSSL, XSL (in the form initially proposed by Microsoft, Inso and ArborText), and the RivComet style language from RivCom. The EXPRESS-driven data in XML form was also shown to be storable in an SGML/XML repository (Chrystal Software's Astoria), where it was merged with an SGML document already present in the repository.

Since June, work has continued on refining the XML representation, taking into account comments and feedback received as a result of that demo session, and addressing aspects of EXPRESS that were not included in the initial work. This work will continue to move forward and will be made available for review and comment to a wider community between now and the end of the year, in order to prepare a first draft of a prospective standard in early 1999.

## ***XML-Data, RDF-Schema and MathML***

It is interesting to compare and contrast the XML representation of EXPRESS-driven data with XML-Data, RDF-Schema and MathML.

XML-Data begins from the notion that the constraints defined in a DTD can also be described using XML instance syntax. It proposes ways of doing this, and adds to them ways of defining additional data types and supertype-subtype hierarchies. The result is a richer version of the semantics of a DTD, that does not use DTD syntax. While going some way towards what an XML version of EXPRESS will provide, XML-Data does not offer the full richness of subtype-supertype structures of EXPRESS. Nor does it provide anything like the power of the constraint-specification capabilities of EXPRESS. However, it does provide a very direct rendering of the document-structure constraints that are central to SGML's traditional areas of application. It will be interesting to see whether the XML representation of EXPRESS will be able to provide a simple intuitive way of specifying document structures equivalent to those provided by a DTD.

The RDF approach does not begin from the DTD constructs, but from a highly generic approach to modeling, where all data structures are expressed as directed graphs of nodes and arcs. From this approach, it provides a way of defining classes of nodes, and supertype-subtype relationships between these classes. Like EXPRESS, RDF embraces the very powerful notion that the value of a property of an object of a given class, may itself be an object of another class. Because it begins from a highly generic, rather than a document-centric approach to data modeling, RDF is probably closer to the spirit of the EXPRESS approach than is XML-Data.

MathML provides an XML representation of most of the basic concepts of mathematics. It therefore provides a powerful and useful model that XML for EXPRESS will be able to draw upon in representing some of the more complex aspects of EXPRESS functions and rules.

It is my hope that by bringing together the best aspects of XML-Data, RDF, XML for EXPRESS, and appropriate parts of MathML, we shall be able to find a single, unified approach to the definition of data schemas in XML. If the attempt to achieve this unification proves vain, we will at least in the process identify the areas of inherent difference between the various approaches, and the reasons for them. This in itself will be extremely valuable, as it will make possible an informed choice as to which approach to use for what specific purposes.

