XCEDE 2.0 - A Manual

XCEDE 2.0 - A Manual

Table of Contents

List of Figures

Introduction

This is a manual for version 2.0 of XCEDE (XML-based Clinical and Experimental Data Exchange). The target audience for this manual is anyone who is interested in using or learning more about XCEDE. This manual will serve as both a tutorial and as a reference.

XCEDE is an extensible schema designed to store scientific data and metadata. XCEDE has its origins in various XML schemas developed for collaborative neuroinformatics projects, and was developed to enable the transfer and storage of several types of data including (but not limited to) clinical, demographic, behavioral, physiological and image data.

Chapter 1. The XCEDE Experiment Hierarchy

This is where the hierarchy text goes.

1.1. Overview

1.2. Examples

1.3. Reference

Chapter 2. Binary Data Resources

2.1. Overview

The XCEDE *Binary Data Resource* component is used to provide a generic interface to a binary data stream stored in one or more external files. Any of the binary data resource types described in this chapter can be used anywhere an abstract_resource_t is called for (with the appropriate `xsi:type` attribute); in the current XCEDE schema, these locations are the top-level `<resource>` element and the `<dataResource>` child element of `<acquisition>`.

XCEDE provides multiple layers of derived types to store more specialized information about the binary data. The base type and each of the derived types are described in turn.

`abstract_resource_t`. The abstract base type `abstract_resource_t` provides a few elements and attributes that are especially important for binary data resources. In particular, the `<uri>` element and its `offset` and `size` attributes point to a "chunk" of data stored in an external file. A series of `<uri>` elements define a stream of data that may be described in greater detail by the data types described below.

`binaryDataResource_t`. This type derives from `abstract_resource_t` and allows an application to interpret the data stream as a sequence of data items with a given data type (`<elementType>`) and byte order (`<byteOrder>`).

`dimensionedBinaryDataResource_t`. The data stream, until now, could only be interpreted as a one-dimensional sequence. This type provides `<dimension>` elements that allow the data stream to be interpreted as a multi-dimensional array of data items. Each dimension has a `<size>` and a `<label>`, as well as the ability to discard subsets of the data in the data stream (using the `outputSelect` attribute).

`mappedBinaryDataResource_t`. This type places the multi-dimensional array of data items represented by `dimensionedBinaryDataResource_t` into an arbitrary coordinate system.

2.2. Examples

Several examples of binary data are presented here, each showing the use of one of the different binary data types described in this chapter.

2.2.1. Basic data stream

The basic binary data resource type describes a sequence of data items. For example, consider a data file (`random_data_file.bin`) containing 2048 random 32-bit floating point numbers, stored in little-endian (least-significant-byte first) order. The `<dataResource>` describing this data is shown in Figure 2.1.

Figure 2.1. Simple `binaryDataResource_t` example

```
<dataResource xsi:type="binaryDataResource_t">
  <uri offset="0" size="8192">random_data_file.bin</uri>
  <elementType>float32</elementType>
  <byteOrder>lsbfirst</byteOrder>
</dataResource>
```

Note the `xsi:type` specifying that this `<dataResource>` element is of type `binaryDataResource_t`. (The `xsi:` prefix should have already been declared previously in the XML file using something similar to `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`)

The `<elementType>` element is restricted one of several pre-defined strings (see the schema for details). The `<byteOrder>` element must be `lsbfirst` for little-endian data or `msbfirst` for big-endian data.

If the `<compression>` element is specified, it specifies that the file(s) pointed to by the `<uri>` elements are compressed. The content of the element should specify which type of compression (the only compression method specifically recognized by this specification is `gzip`). An example of this is shown in Figure 2.2.

Figure 2.2. `binaryDataResource_t` with compression

```
<dataResource xsi:type="binaryDataResource_t">
  <uri offset="0" size="8192">random_data_file.bin.gz</uri>
  <elementType>float32</elementType>
  <byteOrder>lsbfirst</byteOrder>
  <compression>gzip</compression>
</dataResource>
```

As a special case, applications may transparently support data files compressed with gzip, even if only the uncompressed file is referenced in the `<uri>` element. If `random_data_file.bin` did not exist, but `random_data_file.bin.gz` did, the application may uncompress the data itself and use the uncompressed data as if it had came from the file `random_data_file.bin`. Note that it would be an error to reference the uncompressed file `random_data_file.bin` and yet say that it was compressed using `<compression>gzip</compression>`. Compressed files that do not have the `.gz` suffix must use the explicit `compression` element.

2.2.2. Dimensioned data

Consider a camera that acquires an image using a 256x256 matrix of big-endian 32-bit signed integer voxels. This data has two spatial dimensions, which, by convention, we label x, and y (and z if a third spatial dimension is needed, and t if there is a time dimension). Figure 2.3 shows how this data might be represented.

Figure 2.3. `dimensionedBinaryDataResource_t` example

```
<dataResource xsi:type="dimensionedBinaryDataResource_t">
  <uri offset="0" size="262144">rawdata.img</uri>
  <elementType>int32</elementType>
  <byteOrder>msbfirst</byteOrder>
  <dimension label="x">
    <size>256</size>
  </dimension>
  <dimension label="y">
    <size>256</size>
  </dimension>
</dataResource>
```

Dimensions are ordered from fastest-moving to slowest-moving. So in the above example, the x dimension index changes on each consecutive data item, but the y dimension changes every 256 elements.

2.2.3. Mapped data

A “mapped” binary data resource is a (perhaps multidimensional) array of values, the location of each can be determined by converting the matrix indices into a given coordinate system. The location of the bounding box of the data in this space is given by specifying a location for the the first data item, and two things for each dimension: a unit direction vector (in the target coordinate system) and the spacing between successive data items in that dimension. The transformation matrix for a three-dimensional coordinate system has the form shown in Figure 2.4. This transformation matrix converts from matrix indices `(x,y,z)` to a coordinate location `(a,b,c)`. Figure 2.5 shows how the components of a transformation of MR image data into scanner RAS (Right/Anterior/Superior) coordinates are represented

in a `mappedBinaryDataResource_t`. The unit vectors for each dimension are (X_A X_B X_C) = (1 0 0), (Y_A Y_B Y_C) = (0 1 0), and (Z_A Z_B Z_C) = (0 0 1), and are placed in the `<direction>` elements in each `<dimension>` element. The spacing values (S_X S_Y S_Z) = (3.75mm 3.75mm 4mm) are put in the `<spacing>` element in each `<dimension>`. The coordinates of the first voxel in the data are given by (O_A O_B O_C) = (-120 -120 -52).

Figure 2.4. Transformation matrix

$$\begin{pmatrix} X_A & Y_A & Z_A & O_A \\ X_B & Y_B & Z_B & O_A \\ X_C & Y_C & Z_C & O_C \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} S_X & 0 & 0 & 0 \\ 0 & S_Y & 0 & 0 \\ 0 & 0 & S_Z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} i \\ j \\ k \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \\ 1 \end{pmatrix}$$

Figure 2.5. `mappedBinaryDataResource_t` example

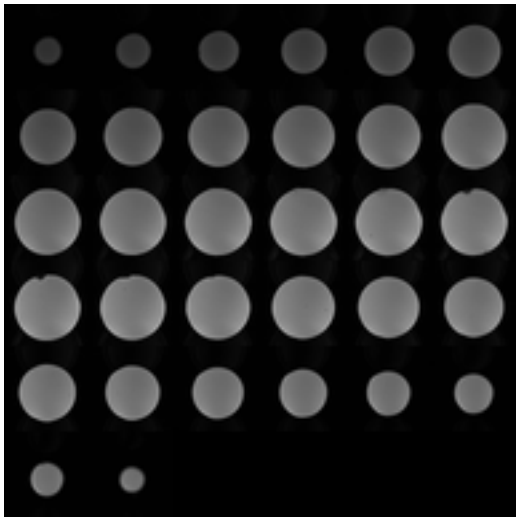
```
<dataResource xsi:type="mappedBinaryDataResource_t">
  <uri offset="0" size="442368">V0001.img</uri>
  <uri offset="0" size="442368">V0002.img</uri>
  <uri offset="0" size="442368">V0003.img</uri>
  <uri offset="0" size="442368">V0004.img</uri>
  <uri offset="0" size="442368">V0005.img</uri>
  <elementType>int32</elementType>
  <byteOrder>msbfirst</byteOrder>
  <dimension label="x">
    <size>64</size>
    <spacing>3.75</spacing>
    <gap>0</gap>
    <direction>1 0 0</direction>
    <units>mm</units>
  </dimension>
  <dimension label="y">
    <size>64</size>
    <spacing>3.75</spacing>
    <gap>0</gap>
    <direction>0 1 0</direction>
    <units>mm</units>
  </dimension>
  <dimension label="z">
    <size>27</size>
    <spacing>4</spacing>
    <gap>1</gap>
    <direction>0 0 1</direction>
    <units>mm</units>
  </dimension>
  <dimension label="t">
    <size>140</size>
    <spacing>2</spacing>
    <gap>0</gap>
    <datapoints>0 2 4 6 8</datapoints>
    <units>sec</units>
  </dimension>
  <originCoords>-120 -120 -52</originCoords>
</dataResource>
```

2.2.4. Advanced topic: split dimensions and outputSelect

A more complicated example is given by data generated by a Siemens MR scanner. In this case, the data represents a three-dimensional 64x64x32 image, stored in DICOM format. However, because the earlier

versions of the DICOM format did not support three-dimensional data in one file, Siemens came upon the clever idea to “tile” the 32 two-dimensional slices across an NxN two-dimensional grid (Figure 2.6).

Figure 2.6. A “tiled”image



Applications may naturally want to express this data as a three-dimensional block, with columns, rows, and slices. In a conventionally-stored three-dimensional XxYxZ image, the first X voxels compose the first row in the first slice, and then the next X voxels are the second row in the first slice; likewise the first X*Y voxels are the first slice, and the next X*Y voxels are the second slice, and so on. However, in the “tiled” image, though the first X voxels are again the first row in the first slice, the next X voxels are the first row in the second slice! At first it would seem that the dimension order has merely been switched, and specifying the labels of the dimensions as x, z, and y would fix things. However, we only hit six slices' first rows before hitting going to the second row of the same six slices. Only after going through all the rows in this fashion in the first six slices do we go on to the next six slices.

What has happened is what we are calling the z dimension has been split in two. The two components of the z dimension are interleaved with the x and y dimensions like so: x, z_1, y, z_2. The two components of the z dimension are distinguished with the `splitRank` attribute, as shown in Figure 2.7.

Figure 2.7. Split dimension example

```
<dataResource xsi:type="binaryDataResource_t">
  <uri offset="9240" size="589824">img0001.dcm</uri>
  <elementType>uint32</elementType>
  <byteOrder>lsbfirst</byteOrder>
  <dimension label="x">
    <size>64</size>
  </dimension>
  <dimension label="z" splitRank="1">
    <size>6</size>
  </dimension>
  <dimension label="y">
    <size>64</size>
  </dimension>
  <dimension label="z" splitRank="2">
    <size>6</size>
  </dimension>
</dataResource>
```

Applications should read this data as if it were four dimensions, and then permute the data to bring the two z dimensions together (in the order specified by splitRank) in the position of the highest-ranked split

dimension, and the two dimensions can then be merged into one. The size of the new z dimension is the product of the sizes of the component split dimensions, so 6 * 6 = 36.

You may recall that the original data was acquired as a 64x64x32 volume, but the NxN tiling representation requires that the number of tiles be the square of two integers. One more mechanism has been added to the `<dimension>` element to accomodate the presence of data that should be disregarded: the `outputSelect` attribute (see Figure 2.8).

Figure 2.8. `outputSelect` example

```
<dataResource xsi:type="binaryDataResource_t">
  <uri offset="0" size="589824">img0001.dcm</uri>
  <elementType>uint32</elementType>
  <byteOrder>lsbfirst</byteOrder>
  <dimension label="x">
    <size>64</size>
  </dimension>
  <dimension label="z" splitRank="1">
    <size>6</size>
  </dimension>
  <dimension label="y">
    <size>64</size>
  </dimension>
  <dimension label="z" splitRank="2" outputSelect="0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31">
    <size>6</size>
  </dimension>
</dataResource>
```

The `outputSelect` attribute specifies a list of indices along the given dimension (or combined dimension if it occurs on the highest-ranked component of a split dimension) that should be regarded as valid data. Data in the other indices should be ignored.

Chapter 3. Catalogs

This is where the catalog text goes.

3.1. Overview

3.2. Examples

3.3. Reference

Chapter 4. Provenance

This is where the provenance text goes.

4.1. Overview

4.2. Examples

4.3. Reference

Chapter 5. Events

5.1. Overview

Events in XCEDE are merely time intervals annotated with arbitrary metadata. This component can be used to represent several types of behavioral data, statistics calculated on time series data, or any other metadata whose proper interpretation requires that it be associated with a particular time interval.

An XCEDE event consists of the following:

onset	The onset (in seconds) of the time interval.
duration	The duration (in seconds) of the time interval.
type	Usage of this field is user-specified
name	Usage of this field is user-specified
units	The units of the onset and duration fields. This field is optional, and it is recommended that users of the schema prescribe an implicit unit of measurement and use it consistently. In that case, this field may be considered informational only.
values	A *value* adds named metadata to this event.

The following instance shows how each of these fields may be populated.

```
<event type="visual" name="event#1" units="sec">
  <onset>0</onset>
  <duration>2</duration>
  <value name="shape">square</value>
  <value name="shapecolor">red</value>
</event>
```

Event elements are stored within the `<data>` element of an `<acquisition>`. The `<data>` element should be of type `events_t` (using `xsi:type` — see examples below).

All onsets are relative to an arbitrary time reference. Typically, time 0 (zero) could mean the start of data acquisition. An event list may be interpreted as concurrent with data in other `<acquisition>` elements (which could be other event lists). If so, the same time reference should be used in all concurrent acquisition data.

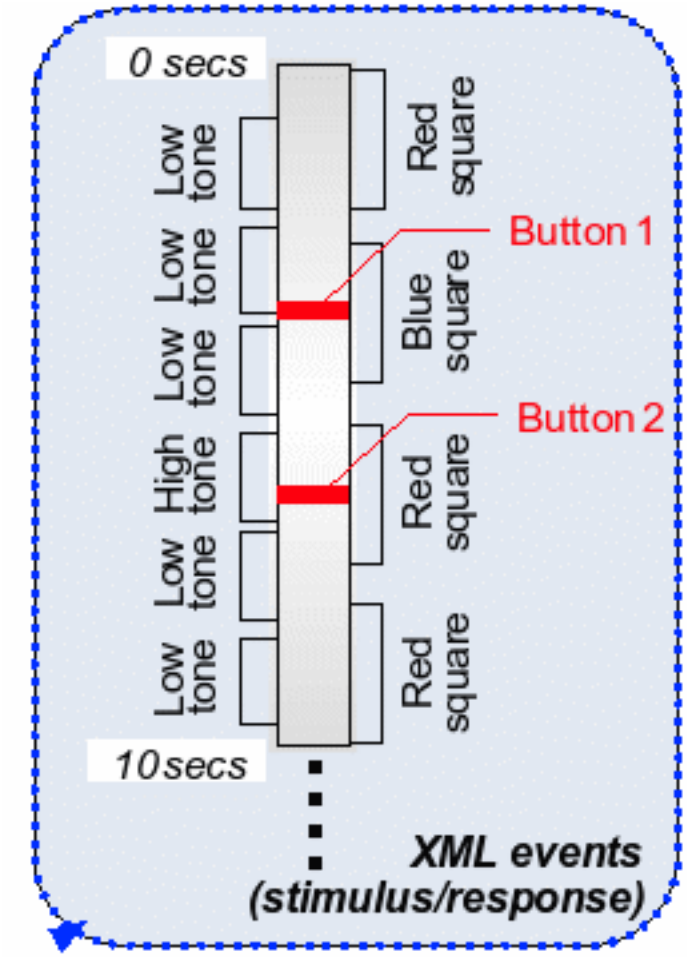
There is no ordering constraint on events in a list. Applications should depend on using the `<onset>` elements to order the events chronologically if they so desire.

An optional `<params>` element may precede the first event in a list, and this element stores arbitrary metadata (using the same `<value>` element used above) that apply to all events in the list.

5.2. Examples

> This example represents stimuli and responses in a neuroimaging study as XCEDE events. However, this is not the only type of data that can be represented using XCEDE events. A different example will be shown later in the chapter.

Figure 5.1. An event timeline



Consider the timeline shown in Figure 5.1. We show in Figure 5.2 how the first 5 seconds worth of the events might be represented in XCEDE.

Figure 5.2. XCEDE Events example

```
<XCEDE xmlns="http://www.xcede.org/xcede-2"
       xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'>
  <acquisition ID="my_events">
    <data xsi:type="events_t">
      <event type="visual">
        <onset>0</onset>
        <duration>2</duration>
        <value name="shape">square</value>
        <value name="shapecolor">red</value>
      </event>
      <event type="visual">
        <onset>2.5</onset>
        <duration>2</duration>
        <value name="shape">square</value>
        <value name="shapecolor">blue</value>
      </event>
      <event type="audio">
        <onset>0.3</onset>
        <duration>1.4</onset>
        <value name="frequency">low</value>
      </event>
      <event type="audio">
        <onset>2.0</onset>
        <duration>1.4</onset>
        <value name="frequency">low</value>
      </event>
      <event type="audio">
        <onset>3.5</onset>
        <duration>1.4</onset>
        <value name="frequency">low</value>
      </event>
      <event type="response">
        <onset>3.4</onset>
        <value name="button">1</value>
      </event>
    </data>
  </acquisition>
</XCEDE>
```

Each stimulus and each response are stored as separate event elements. Note that all the visual events appear first in the XCEDE file, then the audio events, and then the response event. This ordering is arbitrary, and the events could easily have been presented in chronological (or random!) order. The semantic interpretation of the events within an event list must not depend on their document order.

Chapter 6. Assessments

This is where the assessment text goes.

6.1. Overview

6.2. Examples

6.3. Reference

Chapter 7. Protocols

This is where the protocol text goes.

7.1. Overview

7.2. Examples

7.3. Reference

Appendix A. Schema

```
    <?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by dbk (UNIV CA IRVINE)
 -->
<xs:schema xmlns="http://www.xcede.org/xcede-2"
 xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xlink="http://www.w3.org/1999/xlink"
 targetNamespace="http://www.xcede.org/xcede-2" elementFormDefault="qualified"
 attributeFormDefault="unqualified">
 <xs:element name="XCEDE">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="annotationList" minOccurs="0">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="annotation" type="textAnnotation_t" minOccurs="0"
 maxOccurs="unbounded"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="revisionList" minOccurs="0">
     <xs:annotation>
      <xs:documentation>container for document revision history</xs:documentation>
     </xs:annotation>
     <xs:complexType>
      <xs:sequence>
       <xs:element name="revision" type="revision_t" minOccurs="0"
 maxOccurs="unbounded"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="project" type="project_t" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="subject" type="subject_t" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="visit" type="visit_t" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="study" type="study_t" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="episode" type="episode_t" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="acquisition" type="acquisition_t" minOccurs="0"
 maxOccurs="unbounded"/>
    <xs:element name="catalog" type="catalog_t" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="analysis" type="analysis_t" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="resource" type="abstract_resource_t" minOccurs="0"
 maxOccurs="unbounded"/>
    <xs:element name="protocol" type="protocol_t" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="acquisitionProtocol" type="acquisitionProtocol_t" minOccurs="0"
 maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="version" type="xs:string"/>
  </xs:complexType>
 </xs:element>
 <!--*********            Top-level containers            *********-->
 <xs:complexType name="project_t">
  <xs:complexContent>
   <xs:extension base="abstract_container_t">
    <xs:sequence>
     <xs:element name="projectInfo" type="projectInfo_t" minOccurs="0"/>
     <xs:element name="contributorList" minOccurs="0">
      <xs:complexType>
       <xs:sequence>
        <xs:element name="contributor" type="person_t" minOccurs="0"
 maxOccurs="unbounded"/>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
     <xs:element name="subjectList" minOccurs="0">
```

```
     <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
       <xs:element name="subject" type="subject_t"/>
       <xs:element name="subjectRef" type="ref_t">
        <xs:annotation>
         <xs:documentation>This should be an xlink reference</xs:documentation>
        </xs:annotation>
       </xs:element>
      </xs:choice>
     </xs:complexType>
    </xs:element>
    <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="projectInfo_t">
 <xs:complexContent>
  <xs:extension base="abstract_info_t">
   <xs:sequence>
    <xs:element name="exptDesignList" minOccurs="0">
     <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
       <xs:element name="exptDesign"/>
       <xs:element name="exptDesignRef" type="ref_t"/>
      </xs:choice>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="subject_t">
 <xs:complexContent>
  <xs:extension base="abstract_container_t">
   <xs:sequence>
    <xs:element name="projectList" minOccurs="0">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="projectRef" type="ref_t" minOccurs="0" maxOccurs="unbounded">
        <xs:annotation>
         <xs:documentation>This should include participation information, including
per project ID and group.</xs:documentation>
        </xs:annotation>
       </xs:element>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="subjectInfo" type="subjectInfo_t" minOccurs="0"/>
    <xs:element name="visitList" minOccurs="0">
     <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
       <xs:element name="visit" type="visit_t"/>
       <xs:element name="visitRef" type="ref_t"/>
      </xs:choice>
     </xs:complexType>
    </xs:element>
    <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="assessment" type="assessment_t" minOccurs="0"
maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="subjectInfo_t">
 <xs:complexContent>
  <xs:extension base="abstract_info_t">
```

```
    <xs:sequence>
     <xs:element name="sex" type="terminologyString_t" minOccurs="0"/>
     <xs:element name="species" type="terminologyString_t" minOccurs="0"/>
     <xs:element name="birthdate" type="terminologyString_t" minOccurs="0"/>
    </xs:sequence>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
 <xs:complexType name="visit_t">
  <xs:complexContent>
   <xs:extension base="abstract_container_t">
    <xs:sequence>
     <xs:element name="timestamp" type="xs:string" minOccurs="0"/>
     <xs:element name="projectRef" type="ref_t" minOccurs="0"/>
     <xs:element name="subjectRef" type="ref_t" minOccurs="0"/>
     <xs:element name="visitInfo" type="visitInfo_t" minOccurs="0"/>
     <xs:element name="studyList" minOccurs="0">
      <xs:complexType>
       <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="study" type="study_t"/>
        <xs:element name="studyRef" type="ref_t"/>
       </xs:choice>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
    <xs:attribute name="researchGroup" use="optional"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
 <xs:complexType name="visitInfo_t">
  <xs:complexContent>
   <xs:extension base="abstract_info_t"/>
  </xs:complexContent>
 </xs:complexType>
 <xs:complexType name="study_t">
  <xs:complexContent>
   <xs:extension base="abstract_container_t">
    <xs:sequence>
     <xs:element name="projectRef" type="ref_t" minOccurs="0"/>
     <xs:element name="subjectRef" type="ref_t" minOccurs="0"/>
     <xs:element name="visitRef" type="ref_t" minOccurs="0"/>
     <xs:element name="studyInfo" type="studyInfo_t" minOccurs="0"/>
     <xs:element name="episodeList" minOccurs="0">
      <xs:complexType>
       <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="episode" type="episode_t"/>
        <xs:element name="episodeRef" type="ref_t"/>
       </xs:choice>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
 <xs:complexType name="studyInfo_t">
  <xs:complexContent>
   <xs:extension base="abstract_info_t"/>
  </xs:complexContent>
 </xs:complexType>
 <xs:complexType name="episode_t">
  <xs:complexContent>
   <xs:extension base="abstract_container_t">
    <xs:sequence>
     <xs:element name="projectRef" type="ref_t" minOccurs="0"/>
     <xs:element name="subjectRef" type="ref_t" minOccurs="0"/>
     <xs:element name="visitRef" type="ref_t" minOccurs="0"/>
     <xs:element name="studyRef" type="ref_t" minOccurs="0"/>
```

```
    <xs:element name="episodeInfo" type="episodeInfo_t" minOccurs="0"/>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
     <xs:element name="acquisition" type="acquisition_t">
      <xs:annotation>
       <xs:documentation>These represent the actual protocols and data obtained during
an episode.  Multiple acquistions can occur simulataneously during an episode.  For
example, MR, heart rate, and button presses.</xs:documentation>
      </xs:annotation>
     </xs:element>
     <xs:element name="acquisitionRef" type="ref_t"/>
    </xs:choice>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="episodeInfo_t">
 <xs:complexContent>
  <xs:extension base="abstract_info_t"/>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="acquisition_t">
 <xs:complexContent>
  <xs:extension base="abstract_container_t">
   <xs:sequence>
    <xs:element name="acquisitionInfo" type="acquisitionInfo_t" minOccurs="0"/>
    <xs:choice minOccurs="0">
     <xs:element name="dataResource" type="abstract_resource_t">
      <xs:annotation>
       <xs:documentation>A resource that contains acquistion data (i.e. a URL to a
document that contains the data) . The resource could be any format: text file, a
binary file, or an XCEDE xml document, etc.</xs:documentation>
      </xs:annotation>
     </xs:element>
     <xs:element name="dataResourceRef" type="ref_t">
      <xs:annotation>
       <xs:documentation>A reference to a resource as described above.  The resource
could be part of a catalog, a root level resource, etc.</xs:documentation>
      </xs:annotation>
     </xs:element>
     <xs:element name="data" type="abstract_data_t">
      <xs:annotation>
       <xs:documentation>A container that the actual acquisition data can go into (as
opposed to being in an external reousrce)</xs:documentation>
      </xs:annotation>
     </xs:element>
    </xs:choice>
   </xs:sequence>
   <xs:attribute name="acquisitionProtocol" use="optional"/>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="acquisitionInfo_t">
 <xs:complexContent>
  <xs:extension base="abstract_info_t"/>
 </xs:complexContent>
</xs:complexType>
<!--*********           Abstract types           *********-->
<xs:complexType name="abstract_acqProtocol_t" abstract="true">
 <xs:complexContent>
  <xs:extension base="abstract_container_t">
   <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="acqParam" type="protocolDescriptor_t" minOccurs="0"
maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
```

```
</xs:complexType>
<xs:complexType name="abstract_container_t" abstract="true">
 <xs:sequence>
  <xs:element name="commentList" minOccurs="0">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="comment" type="authoredText_t" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="annotationList" minOccurs="0">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="annotation" type="abstract_annotation_t" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="resourceList" minOccurs="0">
   <xs:annotation>
    <xs:documentation>Informational resources related to the
container</xs:documentation>
   </xs:annotation>
   <xs:complexType>
    <xs:sequence>
     <xs:element name="resource" type="informationResource_t" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="provenance" type="provenance_t" minOccurs="0"/>
  <xs:element name="analysisList" minOccurs="0">
   <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
     <xs:element name="analysis" type="analysis_t">
      <xs:annotation>
       <xs:documentation>This should be an abstract analysis_t that is extended to
capture derived data</xs:documentation>
      </xs:annotation>
     </xs:element>
     <xs:element name="analysisRef" type="ref_t"/>
    </xs:choice>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
 <xs:attribute name="ID" type="xs:string"/>
 <xs:attribute name="rev" type="xs:string">
  <xs:annotation>
   <xs:documentation>Revision number, should correspond with an appropriate revision
ID in the XCEDE/history element</xs:documentation>
  </xs:annotation>
 </xs:attribute>
 <xs:attribute name="type" type="xs:string">
  <xs:annotation>
   <xs:documentation>Attribute for creating categories within a container set. For
example, within study, types might include 'PET' or 'MR'.  One could also create
sub-classes using colon notation: "MR:STRUCT"</xs:documentation>
  </xs:annotation>
 </xs:attribute>
 <xs:attributeGroup ref="terminology_ag"/>
</xs:complexType>
<xs:complexType name="abstract_info_t">
 <xs:annotation>
  <xs:documentation>info elements are present in each of the hierarchy levels.
these can be extended to capture instance specific content (following recommendation
5)</xs:documentation>
```

```
 </xs:annotation>
 <xs:sequence>
  <xs:element name="description" type="xs:string" minOccurs="0"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="abstract_data_t"/>
<xs:complexType name="abstract_protocol_t" abstract="true">
 <xs:sequence>
  <xs:element name="protocolOffset" type="protocolOffset_t" minOccurs="0"
maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attributeGroup ref="ID_name_description"/>
 <xs:attributeGroup ref="terminology_ag"/>
 <xs:attribute name="level" type="levelDescriptor">
  <xs:annotation>
   <xs:documentation>Describes the level of the XCEDE hierarchy that this protocol
instance should be validated against</xs:documentation>
  </xs:annotation>
 </xs:attribute>
 <xs:attribute name="required" type="xs:boolean"/>
 <xs:attribute name="minOccurences" type="xs:integer"/>
 <xs:attribute name="maxOccurences" type="xs:integer">
  <xs:annotation>
   <xs:documentation>Are these occurences within a step (i.e. during a single time
point)?  How to refer to repeats across steps?</xs:documentation>
  </xs:annotation>
 </xs:attribute>
 <xs:attribute name="minTimeFromStart" type="xs:string">
  <xs:annotation>
   <xs:documentation>Absolute time from start of overall protocol</xs:documentation>
  </xs:annotation>
 </xs:attribute>
 <xs:attribute name="maxTimeFromStart" type="xs:string"/>
</xs:complexType>
<!--*********            Misc. types            *********-->
<xs:complexType name="acquisitionProtocol_t">
 <xs:complexContent>
  <xs:extension base="abstract_acqProtocol_t"/>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="protocolDescriptor_t">
 <xs:simpleContent>
  <xs:extension base="xs:string">
   <xs:attribute name="description" type="xs:string"/>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
<xs:complexType name="ref_t">
 <xs:simpleContent>
  <xs:extension base="xs:string"/>
 </xs:simpleContent>
</xs:complexType>
<xs:complexType name="assessment_t">
 <xs:complexContent>
  <xs:extension base="abstract_data_t">
   <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="dataInstance" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="assessmentInfo" type="assessmentInfo_t" minOccurs="0">
        <xs:annotation>
         <xs:documentation>Block for describing things like informant, clinical rater,
assessment date, etc</xs:documentation>
        </xs:annotation>
       </xs:element>
```

```
       <xs:element name="assessmentValue" type="assessmentValue_t" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="validated" type="xs:boolean" default="false">
       <xs:annotation>
        <xs:documentation>Indicates whether the instance has been validated (e.g. by
reconciling double-entry instances).  There should be only one validated instance per
assessment.</xs:documentation>
       </xs:annotation>
      </xs:attribute>
     </xs:complexType>
    </xs:element>
    <xs:element name="annotation" type="textAnnotation_t" minOccurs="0"
maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="assessmentInfo_t">
 <xs:complexContent>
  <xs:extension base="acquisitionInfo_t"/>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="assessmentValue_t">
 <xs:sequence>
  <xs:element name="itemText" minOccurs="0">
   <xs:annotation>
    <xs:documentation>'itemText' describes the text preceeding and following an
assessment form item. It is different then 'itemName' which stores the shortened item
identifier</xs:documentation>
   </xs:annotation>
   <xs:complexType>
    <xs:sequence>
     <xs:element name="leadText" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
     <xs:element name="trailText" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="itemName" type="xs:string">
   <xs:annotation>
    <xs:documentation>'itemName' is a short identifier for the assessment item.
'itemText' should be used for the actual question text</xs:documentation>
   </xs:annotation>
  </xs:element>
  <xs:element name="valueStatus" type="xs:string">
   <xs:annotation>
    <xs:documentation>Information on the status of a value (e.g. subject refused to
answer)</xs:documentation>
   </xs:annotation>
  </xs:element>
  <xs:element name="itemValue" type="value_t">
   <xs:annotation>
    <xs:documentation>Actual value of the assessment item as recorded on the
form</xs:documentation>
   </xs:annotation>
  </xs:element>
  <xs:element name="itemNormValue" type="value_t" minOccurs="0">
   <xs:annotation>
    <xs:documentation>Normalized or scaled value of the assessment
item</xs:documentation>
   </xs:annotation>
  </xs:element>
  <xs:element name="reconciliationNote" type="textAnnotation_t" minOccurs="0">
   <xs:annotation>
```

```
    <xs:documentation>Normalized or scaled value of the assessment
item</xs:documentation>
   </xs:annotation>
  </xs:element>
  <xs:element name="annotation" type="textAnnotation_t" minOccurs="0"/>
 </xs:sequence>
 <xs:attributeGroup ref="terminology_ag"/>
</xs:complexType>
<xs:complexType name="analysis_t"/>
<xs:complexType name="investigator_t">
 <xs:complexContent>
  <xs:extension base="person_t"/>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="authoredText_t">
 <xs:simpleContent>
  <xs:extension base="xs:string">
   <xs:attributeGroup ref="authoredText_ag"/>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
<xs:complexType name="abstract_annotation_t" abstract="true">
 <xs:attributeGroup ref="authoredText_ag"/>
</xs:complexType>
<xs:complexType name="textAnnotation_t">
 <xs:complexContent>
  <xs:extension base="abstract_annotation_t">
   <xs:sequence>
    <xs:element name="text" type="xs:string" minOccurs="0"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="nsTermAnnotation_t">
 <xs:complexContent>
  <xs:extension base="abstract_annotation_t">
   <xs:sequence>
    <xs:element name="ontologyClass" type="xs:string" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="nsOntologyAnnotation_t">
 <xs:complexContent>
  <xs:extension base="abstract_annotation_t">
   <xs:sequence>
    <xs:element name="term" type="xs:string" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="generator_t">
 <xs:sequence>
  <xs:element name="application" type="versionedEntity_t">
   <xs:annotation>
    <xs:documentation>Program used to generate document</xs:documentation>
   </xs:annotation>
  </xs:element>
  <xs:element name="invocation" type="xs:string">
   <xs:annotation>
    <xs:documentation>Application input required to generate this document. Should be
explicit such that this document can be re-generated from this info</xs:documentation>
   </xs:annotation>
  </xs:element>
  <xs:element name="dataSource" type="xs:string" minOccurs="0">
   <xs:annotation>
```

```
    <xs:documentation>Description of data source with version numbers and/or
timestamp of data</xs:documentation>
   </xs:annotation>
  </xs:element>
 </xs:sequence>
</xs:complexType>
<xs:attributeGroup name="authoredText_ag">
 <xs:attribute name="author" type="xs:string"/>
 <xs:attribute name="timestamp" type="xs:dateTime"/>
</xs:attributeGroup>
<xs:attributeGroup name="terminology_ag">
 <xs:attribute name="termID" type="xs:string">
  <xs:annotation>
   <xs:documentation>Applications will likely want to constrain what are
valid IDs within the context of their application (for example, allowing only
LSID's)</xs:documentation>
  </xs:annotation>
 </xs:attribute>
 <xs:attribute name="termPath" type="xs:string"/>
</xs:attributeGroup>
<xs:simpleType name="uniqueID_t">
 <xs:annotation>
  <xs:documentation> Having a distinct unique ID type is a convenience for building
referential
   links. The reason we are not using the native XML Schema ID attribute is that
enforces
   document-wide uniqueness, whereas there may be instances of this bioterm schema
that contain
   multiple namespace-qualified term or ontology class sets where IDs are unique
within their
   namespace but not across the entire document. </xs:documentation>
 </xs:annotation>
 <xs:restriction base="xs:string">
  <xs:pattern value="[A-Za-z0-9\-:_.]+"/>
 </xs:restriction>
</xs:simpleType>
<xs:complexType name="person_t">
 <xs:annotation>
  <xs:documentation>Add additional fields (address, email, etc)</xs:documentation>
 </xs:annotation>
 <xs:sequence>
  <xs:element name="salutation" type="xs:string" minOccurs="0">
   <xs:annotation>
    <xs:documentation>e.g. Dr., Mr., Mrs.</xs:documentation>
   </xs:annotation>
  </xs:element>
  <xs:element name="givenName" type="xs:string" minOccurs="0"/>
  <xs:element name="middleName" type="xs:string" minOccurs="0"/>
  <xs:element name="surname" type="xs:string" minOccurs="0">
   <xs:annotation>
    <xs:documentation>Used for last name or only name (e.g. Prince)</xs:documentation>
   </xs:annotation>
  </xs:element>
  <xs:element name="academicTitles" type="xs:string" minOccurs="0"/>
  <xs:element name="institution" type="xs:string" minOccurs="0"/>
  <xs:element name="department" type="xs:string" minOccurs="0"/>
 </xs:sequence>
 <xs:attribute name="ID" type="xs:string"/>
 <xs:attribute name="role" type="xs:string"/>
</xs:complexType>
<xs:complexType name="mixedType_t" mixed="true"/>
<xs:complexType name="versionedEntity_t">
 <xs:simpleContent>
  <xs:extension base="xs:string">
   <xs:attribute name="version" type="xs:string"/>
  </xs:extension>
 </xs:simpleContent>
```

```
</xs:complexType>
<xs:simpleType name="levelDescriptor">
 <xs:restriction base="xs:string">
  <xs:enumeration value="project"/>
  <xs:enumeration value="subject"/>
  <xs:enumeration value="visit"/>
  <xs:enumeration value="study"/>
  <xs:enumeration value="episode"/>
  <xs:enumeration value="acquisition"/>
 </xs:restriction>
</xs:simpleType>
<xs:complexType name="protocolOffset_t">
 <xs:sequence>
  <xs:element name="protocolTimeRef" type="unitString_t" minOccurs="0"/>
  <xs:element name="preferedTimeOffset" type="unitString_t" minOccurs="0"/>
  <xs:element name="minTimeOffset" type="unitString_t" minOccurs="0"/>
  <xs:element name="maxTimeOffset" type="unitString_t" minOccurs="0"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="protocolMeasure_t">
 <xs:attribute name="name"/>
 <xs:attribute name="required"/>
 <xs:attribute name="minValue"/>
 <xs:attribute name="maxValue"/>
</xs:complexType>
<xs:complexType name="protocol_t">
 <xs:complexContent>
  <xs:extension base="abstract_protocol_t">
   <xs:sequence>
    <xs:element name="steps" minOccurs="0">
     <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
       <xs:element name="step" type="protocol_t"/>
       <xs:element name="stepRef" type="ref_t"/>
      </xs:choice>
     </xs:complexType>
    </xs:element>
    <xs:element name="measures" minOccurs="0">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="measure" type="protocolMeasure_t" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:attributeGroup name="ID_name_description">
 <xs:attribute name="ID" type="xs:string"/>
 <xs:attribute name="name" type="xs:string"/>
 <xs:attribute name="description" type="xs:string"/>
</xs:attributeGroup>
<xs:complexType name="terminologyString_t">
 <xs:simpleContent>
  <xs:extension base="xs:string">
   <xs:attributeGroup ref="terminology_ag"/>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
<xs:complexType name="admin">
 <xs:simpleContent>
  <xs:extension base="xs:string"/>
 </xs:simpleContent>
</xs:complexType>
<xs:complexType name="unitString_t">
```

```
 <xs:simpleContent>
  <xs:extension base="xs:string">
   <xs:attribute name="units" type="xs:string"/>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
<xs:complexType name="revision_t">
 <xs:sequence>
  <xs:element name="timestamp" type="xs:dateTime" minOccurs="0"/>
  <xs:element name="generator" type="generator_t" minOccurs="0"/>
  <xs:element name="annotation" type="textAnnotation_t" minOccurs="0"/>
 </xs:sequence>
 <xs:attribute name="ID" type="xs:string"/>
</xs:complexType>
<xs:complexType name="orderedString_t">
 <xs:simpleContent>
  <xs:extension base="xs:string">
   <xs:attribute name="order" type="xs:string"/>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
<xs:complexType name="value_t">
 <xs:simpleContent>
  <xs:extension base="xs:string">
   <xs:attribute name="units" type="xs:string"/>
   <xs:attribute name="codeID" type="xs:string"/>
   <xs:attributeGroup ref="terminology_ag"/>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
<xs:simpleType name="valueTypes_t">
 <xs:restriction base="xs:string">
  <xs:enumeration value="float"/>
  <xs:enumeration value="boolean"/>
  <xs:enumeration value="varchar"/>
  <xs:enumeration value="integer"/>
  <xs:enumeration value="URI"/>
 </xs:restriction>
</xs:simpleType>
<!--*********            Provenance types            *********-->
<xs:complexType name="processStep_t">
 <xs:sequence>
  <xs:element name="package" type="versionedEntity_t" minOccurs="0">
   <xs:annotation>
    <xs:documentation>Software package that contains the program</xs:documentation>
   </xs:annotation>
  </xs:element>
  <xs:element name="program" type="versionedEntity_t">
   <xs:annotation>
    <xs:documentation>Software executable to was run in this step </xs:documentation>
   </xs:annotation>
  </xs:element>
  <xs:element name="programInvocation" type="mixedType_t">
   <xs:annotation>
    <xs:documentation>Exact command line text used to run the
executable</xs:documentation>
   </xs:annotation>
  </xs:element>
  <xs:element name="timeStamp" type="mixedType_t"/>
  <xs:element name="cvs" type="mixedType_t" minOccurs="0"/>
  <xs:element name="user" type="mixedType_t" minOccurs="0"/>
  <xs:element name="machine" type="mixedType_t" minOccurs="0"/>
  <xs:element name="platform" type="versionedEntity_t" minOccurs="0"/>
  <xs:element name="compiler" type="versionedEntity_t" minOccurs="0"/>
  <xs:element name="library" type="versionedEntity_t" minOccurs="0"
maxOccurs="unbounded"/>
 </xs:sequence>
```

```
</xs:complexType>
<xs:complexType name="provenance_t">
 <xs:annotation>
  <xs:documentation>Note: sourceData should be included along with application
parameters and configuration values</xs:documentation>
 </xs:annotation>
 <xs:sequence>
  <xs:element name="processStep" type="processStep_t" maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="ID" type="xs:string">
  <xs:annotation>
   <xs:documentation>Optional identifier indicating the </xs:documentation>
  </xs:annotation>
 </xs:attribute>
</xs:complexType>
<!--*********            Resource types            *********-->
<xs:complexType name="abstract_resource_t" abstract="true">
 <xs:annotation>
  <xs:documentation>A resource is something that we haven't agreed on
yet.</xs:documentation>
 </xs:annotation>
 <xs:sequence>
  <xs:element name="provenance" type="provenance_t" minOccurs="0"/>
  <xs:element name="metaFields" minOccurs="0">
   <xs:complexType>
    <xs:sequence minOccurs="0">
     <xs:element name="metaField" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
       <xs:simpleContent>
        <xs:extension base="xs:string">
         <xs:attribute name="name" type="xs:string"/>
        </xs:extension>
       </xs:simpleContent>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="uri" type="frag_uri_t" minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attributeGroup ref="ID_name_description"/>
 <xs:attribute name="format" type="xs:string" use="optional">
  <xs:annotation>
   <xs:documentation>Format of file. E.g. DICOM, Analyze, 4dfp</xs:documentation>
  </xs:annotation>
 </xs:attribute>
 <xs:attribute name="content" type="xs:string" use="optional">
  <xs:annotation>
   <xs:documentation>Code indicating the contents of the image. E.g. GFC,
T88</xs:documentation>
  </xs:annotation>
 </xs:attribute>
 <xs:attribute name="cachePath" use="optional">
  <xs:simpleType>
   <xs:restriction base="xs:string">
    <xs:maxLength value="255"/>
   </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
</xs:complexType>
<xs:complexType name="informationResource_t">
 <xs:complexContent>
  <xs:extension base="abstract_resource_t"/>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="dcResource_t">
 <xs:complexContent>
```

```
  <xs:extension base="abstract_resource_t">
   <xs:sequence>
    <xs:element name="title" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="creator" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="subject" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="description" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="publisher" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="contributor" type="orderedString_t" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="date" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="type" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="format" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="identifier" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="source" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="language" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="relation" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="coverage" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="rights" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="dataResource_t">
 <xs:complexContent>
  <xs:extension base="abstract_resource_t"/>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="binaryDataResource_t">
 <xs:annotation>
  <xs:documentation>The parent type (abstract_resource_t) can describe a stream of
data.  The extensions in this derived type (binaryDataResource_t) tell you that this
data stream is composed of a sequence of units of a given data type and byte order.
If the </xs:documentation>
 </xs:annotation>
 <xs:complexContent>
  <xs:extension base="abstract_resource_t">
   <xs:sequence>
    <xs:element name="elementType" minOccurs="0">
     <xs:annotation>
      <xs:documentation>This element describes the type of individual data elements in
the data record.  For numeric data types, this indicates whether the element type is
a signed integer ("int"), unsigned integer ("uint"), or floating-point ("float"), as
well as the number of bits allocated to each element.</xs:documentation>
     </xs:annotation>
     <xs:simpleType>
      <xs:restriction base="xs:string">
       <xs:enumeration value="int8"/>
       <xs:enumeration value="uint8"/>
       <xs:enumeration value="int16"/>
       <xs:enumeration value="uint16"/>
       <xs:enumeration value="int32"/>
       <xs:enumeration value="uint32"/>
       <xs:enumeration value="int64"/>
       <xs:enumeration value="uint64"/>
       <xs:enumeration value="float32"/>
       <xs:enumeration value="float64"/>
       <xs:enumeration value="ascii"/>
      </xs:restriction>
     </xs:simpleType>
    </xs:element>
    <xs:element name="byteOrder" minOccurs="0">
     <xs:annotation>
      <xs:documentation>This element describes whether the individual data elements
in the data record are stored with the most-significant-byte first (msbfirst)
```

```
or least-significant-byte first (lsbfirst).  This element is required if the
the data type given by the "elementtype" element has a size larger than one
byte.</xs:documentation>
     </xs:annotation>
     <xs:simpleType>
      <xs:restriction base="xs:string">
       <xs:enumeration value="lsbfirst"/>
       <xs:enumeration value="msbfirst"/>
      </xs:restriction>
     </xs:simpleType>
    </xs:element>
    <xs:element name="compression" type="xs:string" minOccurs="0">
     <xs:annotation>
      <xs:documentation>If this element is present, the files pointed to by the uri
elements are compressed data files.  The only compression method specifically named by
this specification is "gzip".  As a special case for binaryDataResource_t and derived
types, files compressed with gzip and containing a .gz suffix can be referenced in
the uri element without the suffix.  If a file pointed to by the URI does not exist,
the application should search for the same file with the .gz suffix appended -- if it
exists, use that file and act as if the compression element had been specified with
the value "gzip".  This allows the referenced files to be compressed or uncompressed
at will (as long as the .gz suffix is appropriately added/removed from the filename),
without needing to change the URI's in this element.</xs:documentation>
     </xs:annotation>
    </xs:element>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="dimensionedBinaryDataResource_t">
 <xs:annotation>
  <xs:documentation xml:lang="en">This type adds multi-dimensionality to the
(uni-dimensional) data stream represented by binaryDataResource_t.</xs:documentation>
 </xs:annotation>
 <xs:complexContent>
  <xs:extension base="binaryDataResource_t">
   <xs:sequence>
    <xs:element name="dimension" type="binaryDataDimension_t" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="mappedBinaryDataResource_t">
 <xs:annotation>
  <xs:documentation xml:lang="en">This type places the multi-dimensional
data array (say a 3-dimensional cube) into a coordinate space (say MR scanner
coordinates).</xs:documentation>
 </xs:annotation>
 <xs:complexContent>
  <xs:extension base="binaryDataResource_t">
   <xs:sequence>
    <xs:element name="dimension" type="mappedBinaryDataDimension_t"
maxOccurs="unbounded"/>
    <xs:element name="originCoords" type="xs:string" minOccurs="0">
     <xs:annotation>
      <xs:documentation xml:lang="en">This is a coordinate tuple giving the location
of the first item in the data.  For example, if this is an MR volume, this could
be a triple giving the location in RAS coordinates of the first voxel in the
data.</xs:documentation>
     </xs:annotation>
    </xs:element>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="binaryDataDimension_t">
 <xs:annotation>
```

```
  <xs:documentation xml:lang="en">This element stores information about one of the N
dimensions in the data record.  Multiple instances of this element are ordered from
fastest-moving to slowest-moving.  These elements provide information to describe the
size (in data elements) of the N-dimensional bounding box for the data, and in some
cases to describe the mapping of indexes within this bounding box to 'real-world'
coordinates.</xs:documentation>
 </xs:annotation>
 <xs:sequence>
  <xs:element name="size" type="xs:int">
   <xs:annotation>
    <xs:documentation xml:lang="en">The number of elements in the data along one
traversal of this dimension.</xs:documentation>
   </xs:annotation>
  </xs:element>
 </xs:sequence>
 <xs:attribute name="label" type="xs:string">
  <xs:annotation>
   <xs:documentation xml:lang="en">This is a label for the dimension.  The first three
spatial dimensions (or however many exist) must be labeled, in order, 'x', 'y', and
'z'.  The first temporal dimension must be labeled 't'.</xs:documentation>
  </xs:annotation>
 </xs:attribute>
               <xs:attribute name="splitRank" type="xs:string">
                 <xs:annotation>
                   <xs:documentation>If this attribute exists, this dimension is
a "split" dimension, and this dimension must be "merged" with one or more other
dimensions (with the same label) before presenting the data to the application.  This
is useful, for example, if the data is stored in Siemens' Mosaic DICOM format, where
slices of a 3-D volume are arranged to look like they are tiled onto a square 2-D
area.  In this case, what would normally be called the 'z' dimension has two forks,
one that occurs before the 'y' dimension (the first row in the data covers the first
row of several slices), and one that occurs after the 'y' dimension.  If, as in this
case, there are two or more dimensions that should be merged into one, both component
dimensions should have the label 'z', but have splitRank attributes "1" and "2",
which specifies the order in which all 'split' dimensions of the same label will be
merged.  After merging, the resultant 'z' dimension element should contain the same
children of the highest-ranked split 'z' dimension, except for the 'size' element,
which will be the product of the sizes of all 'z' split dimensions.  The position
of the resultant dimension should be the position of the highest-ranked 'z' split
dimension.  The data itself should also be reordered to reflect the new dimension
structure.</xs:documentation>
                 </xs:annotation>
               </xs:attribute>
 <xs:attribute name="outputSelect" type="xs:string">
  <xs:annotation>
   <xs:documentation xml:lang="en">In the same way that the 'splitRank' attribute
allows you to specify dimensions that should be merged before presenting the data to
an application, this attribute specifies a data filter along this dimension.  If this
attribute exists, it should contain a whitespace-separated list of indices (indexed
starting at 0).  Only data points along this dimension that occur in the index list
should be presented to the application.  Likewise, the 'size' of the dimension, after
selection, should be updated to reflect the new size of this dimension (which should
be the number of indices in the content of this attribute).</xs:documentation>
  </xs:annotation>
 </xs:attribute>
</xs:complexType>
<xs:complexType name="mappedBinaryDataDimension_t">
 <xs:complexContent>
  <xs:extension base="binaryDataDimension_t">
   <xs:sequence>
    <xs:element name="origin" type="xs:float" minOccurs="0">
     <xs:annotation>
      <xs:documentation xml:lang="en">A value assigned to the first data element
along this dimension.  For example, if this dimension corresponds to "time",
this element could store the time corresponding to the first data element.
If this is a two-dimensional projection of the surface of the Earth, and this
dimension takes you around the Earth parallel to the equator, this value could
```

```
be the degrees longitude.  For MRI data, this is the single coordinate on the
Left-to-Right, Posterior-to-Anterior, or Inferior-to-Superior axis to which this
dimension most closely matches (see 'direction' element and 'rasOrigin' element in
'mrImageDataResource_t').</xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="spacing" type="xs:float" minOccurs="0">
     <xs:annotation>
      <xs:documentation xml:lang="en">This is the average distance between consecutive
data elements in this dimension.  If the spacing is not regular, then it may be
possible to calculate the actual distance between any two data elements in this
dimension using the 'datapoints' element.</xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="gap" type="xs:float" minOccurs="0">
     <xs:annotation>
      <xs:documentation xml:lang="en">This is the length of the unsampled space
between consecutive data elements in this dimension, i.e. the distance between the end
of one data element and the beginning of the next.  For MRI data, this can be used to
specify the gap between two collected slices &#x02015; the actual width of each slice
can be calculated as 'spacing' minus 'gap'.</xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="datapoints" type="xs:string" minOccurs="0">
     <xs:annotation>
      <xs:documentation xml:lang="en">The content of this element is either (1) a
whitespace-separated list of values, or (2) a list of 'value' elements, that can
be used as a label for each data point along this dimension.  The values can be
numbers representing points on an axis (this is the typical case), text strings,
coordinate tuples, etc.  Any datapoint label that includes whitespace (coordinate
tuples included) must be encapsulated within a child 'value' element.  If this element
is missing, it is assumed that labels can be calculated using information in other
fields (such as 'origin', 'spacing', etc.).  This element is particularly useful for
dimensions with irregular spacing.</xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="direction" type="listoffloats_t" minOccurs="0">
     <xs:annotation>
      <xs:documentation xml:lang="en">This element contains a vector (represented as
a whitespace-separated list of floating-point values in the appropriate coordinate
system) that is parallel to this dimension's edge of the bounding box.  The vector
starts at the first element in the data and points towards subsequent elements along
this dimension.  For MRI data, this should be a unit vector in (R,A,S) coordinates
(positive values are Right, Anterior, or Superior respectively) &#x02015; for 'x' and
'y' dimensions, this corresponds to the two vectors in the ImagePatientOrientation
field in DICOM.</xs:documentation>
     </xs:annotation>
    </xs:element>
    <xs:element name="units" type="xs:string" minOccurs="0">
     <xs:annotation>
      <xs:documentation xml:lang="en">This stores the units used for all numeric
values in this dimension element.  In MRI data, this should be 'mm' for all spatial
dimensions ('x', 'y', 'z') and 'ms' for the temporal dimension 't'.</xs:documentation>
     </xs:annotation>
    </xs:element>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="frag_uri_t">
 <xs:annotation>
  <xs:documentation>The external data pointed to by this uri is a "fragment", where a
"fragment" is defined as a stream of data contiguously stored in the same file offset
by 'offset' bytes and of 'size' bytes.</xs:documentation>
 </xs:annotation>
 <xs:simpleContent>
  <xs:extension base="xs:anyURI">
```

```
   <xs:attribute name="offset" type="xs:unsignedLong">
    <xs:annotation>
     <xs:documentation>The data for this fragment will start at this byte position in
the resource specified by the 'uri' element.  If this attribute does not exist or is
empty, it is assumed to be zero.</xs:documentation>
    </xs:annotation>
   </xs:attribute>
   <xs:attribute name="size" type="xs:unsignedLong">
    <xs:annotation>
     <xs:documentation>This specifies the size of this block (in bytes) in
the resource specified by the 'uri' element.  If this attribute does not
exist or is empty, it is calculated using the dimension and elementtype
element.</xs:documentation>
    </xs:annotation>
   </xs:attribute>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
<xs:simpleType name="listoffloats_t">
 <xs:list itemType="xs:float"/>
</xs:simpleType>
<xs:complexType name="catalog_t">
 <xs:sequence minOccurs="0">
  <xs:element name="metaDataRef" type="ref_t" minOccurs="0">
   <xs:annotation>
    <xs:documentation>Reference to the meta-date element that this catalog is related
to.  Should point to one of the core xcede hierarchy components: project, subject,
visit, study, acquisition.</xs:documentation>
   </xs:annotation>
  </xs:element>
  <xs:element name="catalogList" minOccurs="0">
   <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
     <xs:element name="catalog" type="catalog_t"/>
     <xs:element name="catalogRef" type="ref_t"/>
    </xs:choice>
   </xs:complexType>
  </xs:element>
  <xs:element name="entryList" minOccurs="0">
   <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
     <xs:element name="entry" type="abstract_resource_t"/>
     <xs:element name="entryRef" type="ref_t"/>
    </xs:choice>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
 <xs:attribute name="ID" type="xs:string" use="optional"/>
</xs:complexType>
<xs:complexType name="format_t">
 <xs:annotation>
  <xs:documentation>Container for describing imaging formats and file name extensions
(currently underimplemented)</xs:documentation>
 </xs:annotation>
 <xs:sequence>
  <xs:element name="description" type="xs:string" minOccurs="0"/>
  <xs:element name="documentationList" minOccurs="0">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="documentation" type="informationResource_t" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="extensionList" minOccurs="0">
   <xs:complexType>
    <xs:sequence>
```

```
     <xs:element name="extension" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
 <xs:attribute name="name"/>
</xs:complexType>
<xs:complexType name="catalog_t_expt">
 <xs:complexContent>
  <xs:extension base="abstract_resource_t"/>
 </xs:complexContent>
</xs:complexType>
<!--*********            Event types            *********-->
<xs:complexType name="events_t">
 <xs:complexContent>
  <xs:extension base="abstract_data_t">
   <xs:sequence>
    <xs:element name="params" type="eventParams_t" minOccurs="0"/>
    <xs:element name="event" type="event_t" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="description" type="xs:string" minOccurs="0"/>
    <xs:element name="annotation" type="textAnnotation_t" minOccurs="0"
maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="event_t">
 <xs:annotation>
  <xs:documentation>This element represents an interval of time, with arbitrary
metadata (in the value element).</xs:documentation>
 </xs:annotation>
 <xs:sequence>
  <xs:element name="onset" type="xs:float" minOccurs="0"/>
  <xs:element name="duration" type="xs:float" minOccurs="0"/>
  <xs:element name="value" type="eventValue_t" minOccurs="0" maxOccurs="unbounded"/>
  <xs:element name="annotation" type="textAnnotation_t" minOccurs="0"
maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="type" type="xs:string" use="optional"/>
 <xs:attribute name="units" type="xs:string" use="optional">
  <xs:annotation>
   <xs:documentation>This attribute is optional, but an group using this schema should
agree on, use, and enforce measurement units consistently, to avoid the need for unit
conversion in an application.</xs:documentation>
  </xs:annotation>
 </xs:attribute>
 <xs:attribute name="name" type="xs:string" use="optional"/>
</xs:complexType>
<xs:complexType name="eventValue_t">
 <xs:annotation>
  <xs:documentation>User-specified metadata associated with an
event.</xs:documentation>
 </xs:annotation>
 <xs:simpleContent>
  <xs:extension base="xs:string">
   <xs:attribute name="name" type="xs:string"/>
   <xs:anyAttribute processContents="lax"/>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
<xs:complexType name="eventParams_t">
 <xs:annotation>
  <xs:documentation>These value elements apply to all events in the parent event
list.</xs:documentation>
 </xs:annotation>
 <xs:sequence>
```

```
    <xs:element name="value" type="eventValue_t" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:schema>
```