Technical Note TN2067
About the Keynote XML File Format (APXL Schema)

CONTENTS

This Technical Note contains the schema describing the XML file format used by Keynote documents (refered to here and in later documents simply as "APXL" -- Apple Presentation XmL). It is intended for developers who wish to create or modify Keynote presentations programmatically.

For more information about the Keynote application, please visit the Keynote web page at Apple.com <http://www.apple.com/keynote/>

[Feb 12 2003]

About the Keynote XML file format (APXL Schema)

Keynote documents are bundles (specially-constructed folders) containing a file named "presentation.apxl" and any supporting files (images, sound, QuickTime movies, etc.).

This document is a companion to the four-part Technical Note "Deconstructing a Keynote Document" which describes the document format for Keynote in-depth. Please refer to these Technical Notes as they are made available. (They will be linked to from this page.)

With this schema, you should be able to understand, create and parse a Keynote "presentation.apxl" file. This schema should validate all APXL files that Keynote is able to accept and display.

> **Note:**
> In some cases, in an effort to make this schema more readable, we compromised the strictness of the validation. These cases are commented in the schema itself. In addition, some restrictions could not be expressed by the W3C schema. We've also commented these instances in the APXL schema.

If you're interested in using or creating open-source software which works with the APXL schema, please join the keynote-tools mailing list and project at Open Darwin <http://www.opendarwin.org/mailman/listinfo/keynote-tools>.

In the schema, there are references to "plugins" and a "plugin schema". These are internal to Keynote. Keynote 1.0 does not provide support for third-party plug-ins.

Listing 1: The APXL Schema. Note: This listing has been reformatted for the screen. Please use the downloadable APXL.xsd file in the "Downloads" section rather than cutting-and-pasting this listing.

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<xs:schema  targetNamespace="http://developer.apple.com/schemas/APXL"
     elementFormDefault="qualified"
     xmlns="http://developer.apple.com/schemas/APXL"
     xmlns:plugin="http://developer.apple.com/schemas/APXLPlugins"
     xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!--
_____________________________________________
 Section 0: Schema metadata

_____________________________________________
-->
 <xs:annotation>
  <xs:documentation>
APXL Schema version ready to validate APXL files

Feb 11, 2003 - Released to the public.

  </xs:documentation>
 </xs:annotation>

<!--
Content:

Section 1 Simple Types
Section 2 Commonly used attribute groups
Section 3 Styling attribute groups
Section 4 Style elements and auxiliary elements used with styles
Section 5 Basic drawable types and graphic elements
Section 6 Text content types and elements
Section 7 Bullet styles and bullet elements
Section 8 Placeholders
Section 9 Plugins
Section 10 Drawable grouping
Section 11 Prototypes
Section 12 Transition/build style types and elements
Section 13 Slide and master slide structure
Section 14 Themes
Section 15 Presentation and presentation-level structures

Dictionary "sub-schema"
Section D1 "Keyed" extensions of simple types
Section D2 Types for complex keyed types used only in dictionaries
Section D3 Dictionary and array structures and related groups


_____________________________________________________________________________________
Section 1. Simple Types

_____________________________________________________________________________________
-->

<!-- Numeric types -->
 <xs:simpleType name="p-number-type">
  <xs:restriction base="xs:float">
   <xs:maxInclusive value="1.0"/>
   <xs:minInclusive value="0.0"/>
  </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="list-of-p-numbers">
  <xs:list itemType="p-number-type"/>
 </xs:simpleType>
 <xs:simpleType name="list-of-numbers">
  <xs:list itemType="xs:float"/>
 </xs:simpleType>
```

```
<!-- Geomterical types -->
 <xs:simpleType name="point-type">
  <xs:restriction base="list-of-numbers">
   <xs:length value="2"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="size-type">
  <xs:restriction base="list-of-numbers">
   <xs:length value="2"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="rect-type">
  <xs:restriction base="list-of-numbers">
   <xs:length value="4"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="range-type">
  <xs:restriction base="list-of-numbers">
   <xs:length value="2"/>
  </xs:restriction>
 </xs:simpleType>

 <!-- Transformation is defined by 6 numbers composing tranformation matrix -->
 <xs:simpleType name="transformation-type">
  <xs:restriction base="list-of-numbers">
   <xs:length value="6"/>
  </xs:restriction>
 </xs:simpleType>
<!-- Direction of a text or table element -->
 <xs:simpleType name="direction-type">
  <xs:restriction base="xs:string">
   <xs:enumeration value="vertical"/>
   <xs:enumeration value="horizontal"/>
  </xs:restriction>
 </xs:simpleType>

<!-- vertical-alignment-type -->
 <xs:simpleType name="vertical-alignment-type">
  <xs:restriction base="xs:string">
   <xs:enumeration value="top"/>
   <xs:enumeration value="middle"/>
   <xs:enumeration value="bottom"/>
   <xs:enumeration value="tracks-master"/>
  </xs:restriction>
 </xs:simpleType>
 <!-- Visibility enumeration -->
 <xs:simpleType name="visibility-type">
  <xs:restriction base="xs:string">
   <xs:enumeration value="visible"/>
   <xs:enumeration value="hidden"/>
   <xs:enumeration value="invisisble"/>
   <xs:enumeration value="tracks-master"/>
  </xs:restriction>
 </xs:simpleType>
 <!-- Bullet marker type enumeration -->
 <xs:simpleType name="bullet-marker-type">
  <xs:restriction base="xs:string">
   <xs:enumeration value="none"/>
   <xs:enumeration value="image"/>
```

```
    <xs:enumeration value="sequence"/>
    <xs:enumeration value="character"/>
    <xs:enumeration value="inherited"/>
   </xs:restriction>
  </xs:simpleType>
  <!-- Fill type enumeration -->
  <xs:simpleType name="fillType-type">
   <xs:restriction base="xs:string">
    <xs:enumeration value="none"/>
    <xs:enumeration value="color"/>
    <xs:enumeration value="gradient"/>
    <xs:enumeration value="image"/>
   </xs:restriction>
  </xs:simpleType>
  <!-- Image scale type enumeration -->
  <xs:simpleType name="image-scale-type">
   <xs:restriction base="xs:string">
    <xs:enumeration value="scale-to-fit"/>
    <xs:enumeration value="scale-to-fill"/>
    <xs:enumeration value="full-size"/>
    <xs:enumeration value="stretch"/>
    <xs:enumeration value="tile"/>
   </xs:restriction>
  </xs:simpleType>
  <!-- Font ligature style enumeration -->
  <xs:simpleType name="font-ligature-type">
   <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="none"/>
    <xs:enumeration value="default"/>
    <xs:enumeration value="all"/>
   </xs:restriction>
  </xs:simpleType>
  <!-- Horizontal alignment enumeration -->
  <xs:simpleType name="horizontal-alignment-type">
   <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="left"/>
    <xs:enumeration value="right"/>
    <xs:enumeration value="center"/>
    <xs:enumeration value="justified"/>
    <xs:enumeration value="natural"/>
   </xs:restriction>
  </xs:simpleType>
  <!-- Text wrap style enumeration -->
  <xs:simpleType name="text-wrap-type">
   <xs:restriction base="xs:string">
    <xs:enumeration value="word-wrap"/>
    <xs:enumeration value="char-wrap"/>
    <xs:enumeration value="clip"/>
    <xs:enumeration value="truncate-head"/>
    <xs:enumeration value="truncate-tail"/>
    <xs:enumeration value="truncate-middle"/>
   </xs:restriction>
  </xs:simpleType>
  <!-- number rounding style enumeration -->
  <xs:simpleType name="rounding-mode-type">
   <xs:restriction base="xs:string">
    <xs:enumeration value="plain"/>
    <xs:enumeration value="up"/>
    <xs:enumeration value="down"/>
    <xs:enumeration value="bankers"/>
   </xs:restriction>
  </xs:simpleType>
```

```
 <!-- How size is measured. Used for bullet size -->
 <xs:simpleType name="size-measure-type">
  <xs:restriction base="xs:string">
   <xs:enumeration value="relative"/>
   <xs:enumeration value="absolute"/>
  </xs:restriction>
 </xs:simpleType>
 <!-- Measures the sixe of an image in bytes
 It could be real size in bytes or '-1' for 'undefined -->
 <xs:simpleType name="byte-size-type">
  <xs:restriction base="xs:integer">
   <xs:minInclusive value="-1"/>
  </xs:restriction>
 </xs:simpleType>
<!-- File and Image name type -->
 <xs:simpleType name="file-name-type">
  <xs:restriction base="xs:anyURI">
  </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="image-data-type">
  <xs:restriction base="xs:anyURI">
  <!-- basically, it has to be a file name in the same bundle -->
  </xs:restriction>
 </xs:simpleType>

<!-- Path description -->
 <xs:simpleType name="path-type">

  <xs:annotation>
   <xs:documentation>
 The pattern specifies the way to describe the path in the APXL. It is
 significantly simplified version of the SVG path description.
 path := (step)*
 step := m-step | l-step | c-step | z-step
 m-step:= M num num
 l-step:= L num num
 c-step:= C num num num num num num
 z-step := Z
 num := (float number)
   </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
   <xs:pattern value="((([ML]( -?\d+(\.\d+)?(e-?\d+)?){2})|
      (C( -?\d+(\.\d+)?(e-?\d+)?){6})|Z)\s*)+"/>
  </xs:restriction>
 </xs:simpleType>
 <!-- Tab stop type -->
 <xs:simpleType name="tab-stop-type">
  <xs:restriction base="xs:string">
   <xs:pattern value="([LDRC]\s*\d*(\.\d+)?\s*)+"/>
  </xs:restriction>
 </xs:simpleType>
<!-- Describes the pattern of line
Dash pattern can have one of the following values:
  - word "none" for no stroke at all;
  - word "solid" for solid line;
  - up to six floating point values seperated by spaces, describing dash pattern.
  Example: pattern="3 5 3 4" -->
 <xs:simpleType name="dash-pattern-type">
  <xs:restriction base="xs:string">
   <xs:pattern value="none|solid|(\d*(\.\d+)?\s*){1,6}"/>
  </xs:restriction>
```

```
 </xs:simpleType>

<!-- Number style type - used in numbering pages(slides and bullets)
Repeats CPSequenceType enumeration
-->
 <xs:simpleType name="sequence-style-type">
  <xs:restriction base="xs:string">
   <xs:enumeration value="arabic"/>
   <xs:enumeration value="lowercase-roman"/>
   <xs:enumeration value="uppercase-roman"/>
   <xs:enumeration value="lowercase-alpha"/>
   <xs:enumeration value="uppercase-alpha"/>
  </xs:restriction>
 </xs:simpleType>
<!-- Color types -->

 <!-- RGB-like: A string that contains 3 (RGB) or 4(RGBA white-space separated
  floats (RGBA) all in the range between 0 and 1.0 -->
 <xs:simpleType name="color-type">
  <xs:union>
   <!-- RGB-color-type -->
   <xs:simpleType>
    <xs:restriction base="list-of-p-numbers">
     <xs:length value="3"/>
    </xs:restriction>
   </xs:simpleType>

   <!-- RGBA-color-type -->
   <xs:simpleType>
    <xs:restriction base="list-of-p-numbers">
     <xs:length value="4"/>
    </xs:restriction>
   </xs:simpleType>

   <!-- Grayscale-color-type -->
   <xs:simpleType>
    <xs:restriction base="xs:string">
     <xs:pattern value="g[01](\.0)?"/>
    </xs:restriction>
   </xs:simpleType>

   <xs:simpleType>
    <xs:restriction base="xs:string">
     <xs:pattern value="g0?\.\d+"/>
    </xs:restriction>
   </xs:simpleType>

   <!-- Grayscale-alpha-color-type -->
   <xs:simpleType>
    <xs:restriction base="xs:string">
     <xs:pattern value="g(([01](\.0)?)|(0?\.\d+))\s*([01](\.0)?)|(0?\.\d+)"/>
    </xs:restriction>
   </xs:simpleType>

   <!-- CYMK-color-type
   c<number> y<number> m<number> k<number>
   where <number> is between 0 and 1
   -->
   <xs:simpleType>
    <xs:restriction base="xs:string">
     <xs:pattern value="c([01](\.0)?)|(0?\.\d+)\s*y([01](\.0)?)|(0?\.\d+)
        \s*m([01](\.0)?)|(0?\.\d+)\s*k([01](\.0)?)|(0?\.\d+)">
```

```
      </xs:pattern>
     </xs:restriction>
    </xs:simpleType>

    <!-- CYMKA-color-type
    Same CMYK, followed by <numner> for "alpha" -->
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:pattern value="c([01](\.0)?)|(0?\.\d+)\s*y([01](\.0)?)|(0?\.\d+)
         \s*m([01](\.0)?)|(0?\.\d+)\s*k([01](\.0)?)|
         (0?\.\d+)\s*([01](\.0)?)|(0?\.\d+)">
      </xs:pattern>
     </xs:restriction>
    </xs:simpleType>

    <!-- Catalog color types
    Catalog colors are defined by the pair
    catalog-name:color-name-in-the-catalog
    This definition migh be followed by RGB or RGBA specification used
    as a fallback in case when catalog is not available -->

    <!-- catalog-color-type -->
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:pattern value="[\w#x2D\-]+:[\w#x2D\-]+">
      </xs:pattern>
     </xs:restriction>
    </xs:simpleType>

    <!-- catalog-RGB-color-type -->
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:pattern value="[\w#x2D\-]+:[\w#x2D\-]+(\s*\(([01](\.0)?)|(0?\.\d+)){3}">
      </xs:pattern>
     </xs:restriction>
    </xs:simpleType>

    <!-- catalog-RGBA-color-type -->
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:pattern value="[\w#x2D\-]+:[\w#x2D\-]+(\s*\(([01](\.0)?)|(0?\.\d+)){4}">
      </xs:pattern>
     </xs:restriction>
    </xs:simpleType>

   </xs:union>
  </xs:simpleType>
  <!-- Key type; Attribute "key" mostly used in dictionaries -->
  <xs:simpleType name="key-type">
   <xs:restriction base="xs:string">
   </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="opacity-type">
   <xs:restriction base="p-number-type">
   </xs:restriction>
  </xs:simpleType>

<!--
_____________________________________________________________________________
Section 2. Commonly used attribute groups
_____________________________________________________________________________
-->
```

```
<!-- Geometric attribute groups -->
 <xs:attributeGroup name="bounds">
  <xs:attribute name="location" type="point-type"/>
  <xs:attribute name="size" type="size-type"/>
 </xs:attributeGroup>

<!-- Special attributes for drawables -->
 <xs:attributeGroup name="stroke-attributes">
  <xs:attribute name="stroke-color" type="color-type"/>
  <xs:attribute name="stroke-width" type="xs:float"/>
 </xs:attributeGroup>
 <xs:attributeGroup name="stroke-and-opacity">
  <xs:attributeGroup ref="stroke-attributes"/>
  <xs:attribute name="opacity" type="opacity-type"/>
 </xs:attributeGroup>

<!-- Should be quite close to svg:path element, but way simpler -->
 <xs:attributeGroup name="path">
  <xs:attribute name="path" type="path-type"/>
 </xs:attributeGroup>
<!--
_____________________________________________________________________________
Section 3. Styling attribute groups

Note that we are using "-styling" suffix for attributes and groups
as opposed to "-style" suffix for Style elements

_____________________________________________________________________________
-->

<!-- Text styling -->

 <!-- Character styling -->
 <xs:attributeGroup name="font-styling">
  <xs:attribute name="font-name" type="xs:string"/>
  <xs:attribute name="font-size" type="xs:float"/>
  <xs:attribute name="font-color" type="color-type"/>
 </xs:attributeGroup>
 <!-- These attributes are used for text runs in addition to font attributes. -->
 <xs:attributeGroup name="string-styling">
  <xs:attribute name="font-underline" type="xs:boolean" default="false"/>
  <xs:attribute name="font-superscript" type="xs:float"/>
  <xs:attribute name="font-baseline-offset" type="xs:float"/>
  <xs:attribute name="font-kerning" type="xs:float"/>
  <xs:attribute name="font-ligatures" type="font-ligature-type" default="default"/>
  <xs:attribute name="font-background-color" type="color-type"/>
 </xs:attributeGroup>
 <!-- Paragraph styling -->
 <xs:attributeGroup name="paragraph-styling">
  <xs:attribute name="paragraph-alignment" type="horizontal-alignment-type"/>
  <xs:attribute name="paragraph-line-break" type="text-wrap-type"/>
  <xs:attribute name="paragraph-head-indent" type="xs:float"/>
  <xs:attribute name="paragraph-tail-indent" type="xs:float"/>
  <xs:attribute name="paragraph-first-line-indent" type="xs:float"/>
  <xs:attribute name="paragraph-line-spacing" type="xs:float"/>
  <xs:attribute name="paragraph-line-min-height" type="xs:float"/>
  <xs:attribute name="paragraph-line-max-height" type="xs:float"/>
  <xs:attribute name="paragraph-spacing" type="xs:float"/>
  <xs:attribute name="tab-stops" type="tab-stop-type"/>
 </xs:attributeGroup>
 <!-- All text attributes combined for "fully styled text" -->
 <xs:attributeGroup name="text-styling">
  <xs:attributeGroup ref="font-styling"/>
```

```
  <xs:attributeGroup ref="string-styling"/>
  <xs:attributeGroup ref="paragraph-styling"/>
 </xs:attributeGroup>
<!-- Stroke and fill attributes -->

 <!-- Dash styling attributes -->
 <xs:attributeGroup name="dash-styling">
  <xs:attribute name="pattern" type="dash-pattern-type"/>
  <xs:attribute name="offset" type="xs:float"/>
 </xs:attributeGroup>
 <!-- Shadow-styling attributes -->
 <xs:attributeGroup name="shadow-style">
  <xs:attribute name="angle" type="xs:float" default="0"/>
  <xs:attribute name="offset" type="xs:float" default="0"/>
  <xs:attribute name="radius" type="xs:nonNegativeInteger" default="0"/>
  <xs:attribute name="opacity" type="opacity-type" default="1"/>
  <xs:attribute name="color" type="color-type"/>
 </xs:attributeGroup>

 <!-- Fill styling -->
 <xs:attributeGroup name="image-fill-styling-group">
  <xs:attribute name="image-data" type="image-data-type"/>
  <xs:attribute name="image-scale" type="image-scale-type"/>
  <xs:attribute name="byte-size" type="xs:nonNegativeInteger"/>
 </xs:attributeGroup>

 <xs:attributeGroup name="gradient-attribute-group">
  <xs:attribute name="start-color" type="color-type"/>
  <xs:attribute name="end-color" type="color-type"/>
  <xs:attribute name="gradient-angle" type="xs:float"/>
 </xs:attributeGroup>
<!--
_____________________________________________________________________________
Section 4. Style elements and auxiliary elements used with styles

Note that we are using "-styling" suffix for attributes and groups
as opposed to "-style" suffix for Style elements

_____________________________________________________________________________
-->

 <!-- Dash style type and element -->
 <xs:complexType name="dash-style-type">
  <xs:attributeGroup ref="dash-styling"/>
  <xs:attribute name="key" type="key-type"/>
  <xs:attribute name="id" type="xs:ID"/>
 </xs:complexType>

 <xs:element name="dash-style" type="dash-style-type"/>
 <!-- Line-end style type and elements -->
 <xs:complexType name="line-end-style-type">
  <!-- match-x and match-y specify a "match point" for a line-end.
   The match point is defined in the same co-ordinate space as the path.
   It is this point that is matched with the end of the line
   that is being decorated. (Rotation also occurs around this point.)
   -->
  <xs:attribute name="match-point" type="point-type" />
  <xs:attributeGroup ref="path"/>
  <xs:attribute name="is-filled" type="xs:boolean"/>
  <xs:attribute name="key" type="key-type"/>
  <xs:attribute name="id" type="xs:ID"/>
 </xs:complexType>
```

```
<xs:element name="line-head-style" type="line-end-style-type"/>
<xs:element name="line-tail-style" type="line-end-style-type"/>
<!-- Shadow -->
<xs:complexType name="shadow-style-type">
 <xs:attributeGroup ref="shadow-style"/>
 <xs:attribute name="key" type="key-type"/>
 <xs:attribute name="id" type="xs:ID"/>
</xs:complexType>

<xs:element name="shadow-style" type="shadow-style-type"/>
<!-- gradient and other fills -->
<xs:element name="gradient">
 <xs:complexType>
  <xs:attributeGroup ref="gradient-attribute-group"/>
 </xs:complexType>
</xs:element>
<xs:complexType name="fill-style-type">
<!-- In fact, we have 4 alternatives: gradient, solid fill, image fill, no fill
 The alternative is defined by the value of the "fill-type" attribute.
-->
 <!-- Only for gradient type -->
 <xs:choice minOccurs="0">
  <xs:element ref="gradient"/>
 </xs:choice>
 <!-- for all types -->
 <xs:attribute name="fill-type" type="fillType-type"/>
 <!-- only for solid fill type -->
 <xs:attribute name="fill-color" type="color-type"/>
 <!-- only for image fill type -->
 <xs:attributeGroup ref="image-fill-styling-group"/>
 <xs:attribute name="key" type="key-type"/>
 <xs:attribute name="id" type="xs:ID"/>
</xs:complexType>
<xs:element name="fill-style" type="fill-style-type"/>
<xs:element name="background-fill-style" type="fill-style-type"/>
<!-- all graphic styles as a group -->
<xs:group name="styles-group">
 <xs:choice>
  <xs:element ref="dash-style"/>
  <xs:element ref="shadow-style"/>
  <xs:element ref="fill-style"/>
  <xs:element ref="background-fill-style"/>
  <xs:element ref="line-head-style"/>
  <xs:element ref="line-tail-style"/>
 </xs:choice>
</xs:group>
<!-- Drawable styles hierarchy -->
<xs:complexType name="basic-drawable-style-type">
 <xs:all>
  <xs:element ref="shadow-style" minOccurs="0"/>
 </xs:all>
</xs:complexType>

<xs:complexType name="outlined-drawable-style-type">
 <xs:all>
  <xs:element ref="shadow-style" minOccurs="0"/>
  <xs:element ref="dash-style" minOccurs="0"/>
 </xs:all>
</xs:complexType>
<xs:complexType name="filled-drawable-style-type">
 <xs:all>
  <xs:element ref="shadow-style" minOccurs="0"/>
```

```
    <xs:element ref="dash-style" minOccurs="0"/>
    <xs:element ref="fill-style" minOccurs="0"/>
   </xs:all>
  </xs:complexType>
  <xs:complexType name="line-drawable-style-type">
   <xs:all>
    <xs:element ref="shadow-style" minOccurs="0"/>
    <xs:element ref="dash-style" minOccurs="0"/>
    <xs:element ref="line-head-style" minOccurs="0"/>
    <xs:element ref="line-tail-style" minOccurs="0"/>
   </xs:all>
  </xs:complexType>

 <!--
 ____________________________________________________________________________
 Section 5. Basic drawable types and graphic elements
 ____________________________________________________________________________
 -->
  <!-- Base type for all drawables on slides -->
  <xs:complexType name="drawable-type">
   <xs:attribute name="locked" type="xs:boolean" default="false"/>
   <xs:attribute name="id" type="xs:ID" />
  <xs:attribute name="key" type="key-type"/>
  </xs:complexType>

  <!-- All drawables that could be moved -->
  <xs:complexType name="content-type">
   <xs:complexContent>
    <xs:extension base="drawable-type">
     <xs:attribute name="transformation" type="transformation-type"/>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
  <!-- Graphic elements -->
  <xs:element name="image">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="content-type">
      <xs:sequence>
       <xs:element name="styles" type="outlined-drawable-style-type" minOccurs="0"/>
      </xs:sequence>
      <xs:attributeGroup ref="stroke-and-opacity"/>
      <!-- image-specific attributes -->
      <xs:attribute name="display-name" type="xs:string"/>
      <xs:attribute name="image-data" type="image-data-type"/>
      <xs:attribute name="lock-aspect-ratio" type="xs:boolean"/>
      <xs:attribute name="natural-size" type="size-type"/>
      <xs:attribute name="byte-size" type="byte-size-type"/>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="shape">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="content-type">
      <xs:sequence>
       <xs:element name="styles" type="filled-drawable-style-type" minOccurs="0"/>
      </xs:sequence>
      <xs:attributeGroup ref="path"/>
      <xs:attributeGroup ref="stroke-and-opacity"/>
     </xs:extension>
```

```
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="line">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="content-type">
      <xs:sequence>
       <xs:element name="styles" type="line-drawable-style-type" minOccurs="0"/>
      </xs:sequence>
    <!-- "required" property is removed to accomodate prototypes -->
      <xs:attribute name="head" type="point-type" />
      <xs:attribute name="tail" type="point-type" />
      <xs:attributeGroup ref="stroke-and-opacity"/>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
<!--
_____________________________________________________________________________
Section 6. Text content types and elements
_____________________________________________________________________________
-->
  <xs:complexType name="font-styled-text-type">
   <xs:simpleContent>
    <xs:extension base="xs:string">
     <xs:attributeGroup ref="font-styling"/>
    </xs:extension>
   </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="text-span-type">
   <xs:simpleContent>
    <xs:extension base="font-styled-text-type">
     <xs:attributeGroup ref="string-styling"/>
    </xs:extension>
   </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="full-styled-text-type">
   <xs:simpleContent>
    <xs:extension base="text-span-type">
     <xs:attributeGroup ref="paragraph-styling"/>
    </xs:extension>
   </xs:simpleContent>
  </xs:complexType>
  <xs:element name="span" type="text-span-type"/>
  <xs:element name="div">
   <xs:complexType mixed="true">
    <xs:sequence>
     <xs:element ref="span" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attributeGroup ref="text-styling"/>
   </xs:complexType>
  </xs:element>
  <xs:complexType name="text-content-type" mixed="true">
   <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="span"/>
    <xs:element ref="div"/>
   </xs:choice>
   <xs:attributeGroup ref="text-styling"/>
   <xs:attribute name="key" type="key-type"/>
   <xs:attribute name="id" type="xs:ID"/>
  </xs:complexType>
```

```
 <!-- Note that this element is also used in the Table plugin -->
 <xs:element name="content" type="text-content-type"/>

 <!-- Textbox -->
 <xs:element name="textbox">
  <xs:complexType>
   <xs:complexContent>
    <xs:extension base="content-type">
     <xs:all>
      <xs:element name="styles" type="basic-drawable-style-type" minOccurs="0"/>
      <xs:element ref="content" minOccurs="0"/>
     </xs:all>
     <xs:attribute name="size" type="size-type"/>
     <xs:attribute name="grow-horizontally" type="xs:boolean" default="false"/>
     <xs:attribute name="opacity" type="opacity-type" default="1"/>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
 </xs:element>
<!--
_____________________________________________________________________________
Section 7. Bullet styles and bullet elements
_____________________________________________________________________________
-->
 <xs:complexType name="bullet-style-type">
  <xs:attribute name="size" type="xs:float"/>
  <xs:attribute name="size-technique" type="size-measure-type"/>
  <xs:attribute name="offset" type="xs:float"/>
  <xs:attribute name="key" type="key-type"/>
 </xs:complexType>
 <xs:element name="image-bullet-style">
  <xs:complexType>
  <xs:complexContent>
   <xs:extension base="bullet-style-type">
    <xs:attribute name="image-data" use="required" type="image-data-type"/>
    <xs:attribute name="byte-size" type="byte-size-type"/>
   </xs:extension>
  </xs:complexContent>
  </xs:complexType>
 </xs:element>

 <xs:element name="sequence-bullet-style">
  <xs:complexType>
  <xs:complexContent>
   <xs:extension base="bullet-style-type">
    <xs:attribute name="sequence-type" type="sequence-style-type"/>
    <xs:attribute name="format-string" type="xs:string"/>
   </xs:extension>
  </xs:complexContent>
  </xs:complexType>
 </xs:element>
 <xs:element name="bullet-characters" type="full-styled-text-type"/>

 <xs:element name="character-bullet-style">
  <xs:complexType >
   <xs:complexContent>
    <xs:extension base="bullet-style-type">
    <!-- Note that the real characters go inside this element
    as a fully styled text -->
     <xs:sequence>
      <xs:element ref="bullet-characters"/>
     </xs:sequence>
```

```
    </xs:extension>
   </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="bullet">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="content"/>
     <xs:choice minOccurs="0" maxOccurs="1">
      <xs:element ref="character-bullet-style"/>
      <xs:element ref="sequence-bullet-style"/>
      <xs:element ref="image-bullet-style"/>
     </xs:choice>
    </xs:sequence>
    <xs:attribute name="level" type="xs:nonNegativeInteger"/>
    <xs:attribute name="marker-type" type="bullet-marker-type"/>
    <xs:attribute name="spacing" type="xs:float"/>
    <xs:attribute name="offset" type="xs:float"/>
    <xs:attribute name="id" type="xs:ID" />
   </xs:complexType>
  </xs:element>
  <xs:element name="bullets">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="bullet" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>

 <!--
 ____________________________________________________________________________
 Section 8. Placeholders
 ____________________________________________________________________________
 -->
  <!-- Placeholder type  -->
  <xs:complexType name="placeholder-type">
   <xs:complexContent>
    <xs:extension base="drawable-type">
     <xs:sequence>
      <xs:element name="styles" type="filled-drawable-style-type" minOccurs="0"/>
     </xs:sequence>
     <xs:attributeGroup ref="bounds"/>
     <xs:attributeGroup ref="stroke-and-opacity"/>
     <xs:attribute name="vertical-alignment" type="vertical-alignment-type"/>
     <xs:attribute name="visibility" type="visibility-type"/>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>

  <!-- Specific placeholders -->
  <xs:element name="title">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="placeholder-type"/>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>

  <xs:element name="body">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="placeholder-type">
```

```
      <xs:attribute name="bullet-indentation" type="list-of-numbers"/>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="text-attributes">
   <xs:complexType>
    <xs:attributeGroup ref="text-styling"/>
   </xs:complexType>
  </xs:element>

  <!-- Only on master slides -->
  <xs:element name="page-number">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="placeholder-type">
      <xs:sequence>
       <xs:element ref="text-attributes"/>
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
 <!--
 ____________________________________________________________________________
 Section 9. Plugins
 Note that for rendering purposes, plugins are just drawables that could be
 moved, transformeds, assigned opcaity, etc.

 ____________________________________________________________________________
 -->
  <xs:element name="plugin-data">
   <xs:complexType>
    <xs:sequence>
     <xs:group ref="content-data-group" maxOccurs="unbounded" minOccurs="0"/>
     <xs:any namespace="http://apple.com/schemas/APXLPlugins" processContents="lax"/>
     <!--
     The proper way of specifying the content here would be:
     <choice>
      <xs:element ref="plugin:chart"/>
      <xs:group ref="plugin:table"/>
     </choice>

     Unfortunately, this would require "circular refernce between APXL.xsd
     and APXLPlugins.xsd It is OK with W3C XML schema spec, but is not
     with most validators
     -->
     <xs:group ref="content-data-group" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="plugin">
   <xs:complexType>
   <xs:complexContent>
    <xs:extension base="content-type">
     <xs:all>
      <xs:element name="styles" type="basic-drawable-style-type" minOccurs="0"/>
      <xs:element ref="plugin-data"/>
     </xs:all>
     <xs:attribute name="opacity" type="opacity-type"/>
    </xs:extension>
   </xs:complexContent>
   </xs:complexType>
```

```
 </xs:element>

<!--
_____________________________________________________________________________
Section 11. Drawable grouping
_____________________________________________________________________________
-->
<!-- It contains evrything that could be drown, with exception of placeholders -->
 <xs:group name="drawable-group">
  <xs:choice>
   <xs:element ref="shape"/>
   <xs:element ref="line"/>
   <xs:element ref="g"/>
   <xs:element ref="textbox"/>
   <xs:element ref="image"/>
   <xs:element ref="plugin"/>
  </xs:choice>
 </xs:group>
 <xs:element name="g">
  <xs:complexType>
   <xs:complexContent>
    <xs:extension base="content-type">
     <xs:sequence>
      <xs:group ref="drawable-group" maxOccurs="unbounded"/>
     </xs:sequence>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
 </xs:element>
 <xs:complexType name="slide-drawables-type">
 <!-- Slide drawables can contain any number of "normal" drawables,
 plus exactly one <title> placeholder and exactly one <body> placeholder mixed
 within drawables arbitrary
  -->
  <xs:sequence>
   <xs:group ref="drawable-group" minOccurs="0" maxOccurs="unbounded"/>
   <!-- which one goes first? title or body? -->
   <xs:choice>
    <!-- title is first, followed by body -->
    <xs:sequence>
     <xs:element ref="title"/>
     <xs:group ref="drawable-group" minOccurs="0" maxOccurs="unbounded"/>
     <xs:element ref="body"/>
    </xs:sequence>

    <!-- body goes first -->
    <xs:sequence>
     <xs:element ref="body"/>
     <xs:group ref="drawable-group" minOccurs="0" maxOccurs="unbounded"/>
     <xs:element ref="title"/>
    </xs:sequence>

   </xs:choice>
  <!-- OK, finally placeholders are followed by zero or more "normal" drawables -->
  <xs:group ref="drawable-group" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="master-slide-drawables-type">
 <!-- Master slide drawables are similar, but in addition
 they can contain an optional <page-number> placeholder.
 Unfortunately, using the same approach as in <slide-drawable-type>
 leads to a non-deterministic schema
```

```
 So, we put instead a crude approximation that will validate almost everything
  -->
  <xs:sequence>
   <xs:choice minOccurs="1" maxOccurs="unbounded">
   <!-- master slide placeholders -->
    <xs:element ref="body"/>
    <xs:element ref="title"/>
    <xs:element ref="page-number"/>
   <!-- the rest of drawable elements -->
    <xs:group ref="drawable-group"/>
   </xs:choice>
  </xs:sequence>

 </xs:complexType>

<!--
_____________________________________________________________________________
Section 11. Prototypes
Prototypes are used on master slides to define default styles for drawable
elements
_____________________________________________________________________________
-->
 <!-- This group is used inside prototype drwabales.
 Note that it contains no <plugin>s and no <g>. -->
 <xs:group name="prototype-drawable-group">
  <xs:choice>
   <xs:element ref="shape"/>
   <xs:element ref="line"/>
   <xs:element ref="textbox"/>
   <xs:element ref="image"/>
  </xs:choice>
 </xs:group>
 <!-- Prototype for drawables could be used on theme or on master slide level -->
 <xs:element name="prototype-drawables">
  <xs:complexType>
   <xs:group ref="prototype-drawable-group" minOccurs="0" maxOccurs="unbounded"/>
  </xs:complexType>
 </xs:element>
 <!-- Prototype-bullets elements always contain excatly 6 bullets! -->
 <xs:element name="prototype-bullets">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="bullet" minOccurs="6" maxOccurs="6"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>

 <!-- In fact, it completely repeates "plugin" element defintion
 but has a "key" attribute in addition.
 Then, unlike <plugin-data>, <prototype-plugin-data> is an array with "table"
 or "chart" element being only one of possible elements of it. In fact, it
 is completely up to the plugin developer what do they put in there.
 "prototype-data" acts as an array
 -->
 <xs:element name="prototype-data">
  <xs:complexType>
   <xs:group ref="general-data-group" maxOccurs="unbounded" minOccurs="1"/>
  </xs:complexType>
 </xs:element>

 <xs:element name="prototype-plugin">
  <xs:complexType>
```

```
    <xs:complexContent>
     <xs:extension base="content-type">
      <xs:all>
       <xs:element name="styles" type="basic-drawable-style-type" minOccurs="0"/>
       <xs:element ref="prototype-data"/>
      </xs:all>
      <xs:attribute name="opacity" type="opacity-type"/>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="prototype-plugins">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="prototype-plugin" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>

  </xs:element>

 <!--
 ____________________________________________________________________________
 Section 12. Transition/build style types and elements
 ____________________________________________________________________________
 -->
  <xs:complexType name="name-integer-value-type">
   <xs:attribute name="name" type="xs:string" use="required"/>
   <xs:attribute name="value" type="xs:integer" use="required"/>
   <xs:attribute name="key" type="key-type"/>
  </xs:complexType>
  <xs:element name="property" type="name-integer-value-type"/>
  <xs:complexType name="transition-style-type">
   <xs:sequence>
    <xs:element ref="property" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
   <!-- There is no enumeration in the code - just a string for now,
   so we keep it open for more transition styles -->
   <xs:attribute name="type" type="xs:string" use="required"/>
   <xs:attribute name="duration" type="xs:float" />
   <xs:attribute name="key" type="key-type"/>
  </xs:complexType>
  <xs:element name="transition-style" type="transition-style-type"/>
  <xs:complexType name="build-type">
   <xs:complexContent>
    <xs:extension base="transition-style-type">
     <xs:attribute name="target-id" type="xs:IDREF" use="required"/>
     <xs:attribute name="delivery" type="xs:integer"/>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
  <xs:element name="build" type="build-type"/>
  <!-- Build events are grouped in the slide -->
  <xs:element name="events">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="build" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 <!--
 ____________________________________________________________________________
 Section 12. Other slide-level elements and groups
```

```
-->
 <!-- Guides -->
 <xs:element name="guide">
  <xs:complexType>
   <xs:attribute name="orientation" type="direction-type"/>
   <xs:attribute name="offset" type="xs:float"/>
   <xs:attribute name="key" type="key-type"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="guides">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="guide" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <!-- Text notes -->
 <xs:element name="notes" type="text-content-type">
 </xs:element>
 <!-- Thumbnails -->
 <xs:element name="thumbnail">
  <xs:complexType>
   <xs:attribute name="file" use="required" type="file-name-type"/>
   <xs:attribute name="size" use="required" type="size-type"/>
   <xs:attribute name="byte-size" type="byte-size-type"/>
   <xs:attribute name="key" type="key-type"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="thumbnails">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="thumbnail" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>

<!--
_____________________________________________________________________________
Section 13. Slide and master slide structure
_____________________________________________________________________________
-->
 <xs:element name="master-slide">
  <xs:complexType>
   <xs:all>
    <xs:element name="drawables" type="master-slide-drawables-type" minOccurs="0"/>
    <xs:element ref="background-fill-style" minOccurs="0"/>
    <xs:element ref="prototype-bullets"/>
    <xs:element ref="transition-style" minOccurs="0"/>
    <!-- thumbnails - as many as necessary for different sizes -->
    <xs:element ref="thumbnails" minOccurs="0"/>
    <xs:element ref="prototype-plugins" minOccurs="0"/>
    <xs:element ref="prototype-drawables" minOccurs="0"/>
    <!-- Master slide can also have guides -->
    <xs:element ref="guides" minOccurs="0"/>

   </xs:all>
   <xs:attribute name="id" type="xs:ID" use="required"/>
   <xs:attribute name="name" type="xs:string"/>
   <!-- If true, master slide objects can be put at any z-order on a slide -->
   <xs:attribute name="floating-content" type="xs:boolean" default="false"/>
```

```
  </xs:complexType>
 </xs:element>
 <xs:element name="slide">
  <xs:complexType>
   <xs:all>
    <xs:element name="drawables" type="slide-drawables-type" minOccurs="0"/>
    <xs:element ref="background-fill-style" minOccurs="0"/>
    <xs:element ref="bullets" minOccurs="0"/>
    <xs:element ref="transition-style" />
    <!-- thumbnails - as many as necessary for different sizes -->
    <xs:element ref="thumbnails" minOccurs="0"/>
    <xs:element ref="guides" minOccurs="0"/>
    <xs:element ref="events" minOccurs="0"/>
    <xs:element ref="notes" minOccurs="0"/>
   </xs:all>
   <xs:attribute name="master-slide-id" type="xs:IDREF" use="required"/>
   <xs:attribute name="depth" type="xs:nonNegativeInteger" default="0"/>
   <!-- indicates z-order of master slide objects inside sliede' drawable list -->
   <xs:attribute name="master-content-index" type="xs:nonNegativeInteger" default="0"/>
   <xs:attribute name="hidden" type="xs:boolean" default="false"/>
   <xs:attribute name="collapsed" type="xs:boolean" default="false"/>
   <xs:attribute name="autobuild" type="xs:boolean" default="false"/>
   <xs:attribute name="id" type="xs:ID"/>
  </xs:complexType>
 </xs:element>
<!--
_____________________________________________________________________________
Section 14. Themes
_____________________________________________________________________________
-->
 <xs:element name="master-slides">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="master-slide" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="theme">
  <xs:complexType>
   <xs:all>
    <!-- Templates for plugins and drawables - replace styles -->
    <!-- Prototype drawables decsribe prototype for shapes and text
    and basically define master styles -->
    <xs:element ref="prototype-drawables" minOccurs="0"/>
    <!-- Similar function for prototype plugins -->
    <xs:element ref="prototype-plugins" minOccurs="0"/>

    <!-- we have to make them children to be able to
    escape quotes, etc. -->
    <xs:element name="description" minOccurs="0" type="xs:string"/>

    <!-- Local defintiion for Master slide list -->
    <xs:element ref="master-slides"/>
   </xs:all>

   <xs:attribute name="slide-size" type="size-type" use="required"/>
   <xs:attribute name="changed" type="xs:boolean" default="false"/>
   <xs:attribute name="self-contained" type="xs:boolean" default="false"/>

  </xs:complexType>
 </xs:element>
<!--
```

```
Section 15. Presentation and presentation-level structures
-->
 <!-- Slide list element describes the real sequence of slides
     to be shown.-->
 <xs:element name="slide-list">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="slide" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <!-- Metadata -->
 <xs:element name="application-name" type="xs:string"/>
 <xs:element name="application-version" type="xs:string" />
 <xs:element name="time-stamp" type="xs:string"/>
 <xs:element name="metadata">
  <xs:complexType>
   <xs:all>
    <xs:element ref="application-name" />
    <xs:element ref="application-version" minOccurs="0"/>
    <xs:element ref="time-stamp" minOccurs="0"/>
   </xs:all>
  </xs:complexType>
 </xs:element>
 <!-- UI state
 This is a dictionary of almost arbitrary parameters that could be stored
 to preserve the UI state. -->
 <xs:element name="ui-state" type="dictionary-type">
  <xs:key name="ui-key">
   <xs:selector xpath="*"/>
   <xs:field xpath="@key"/>
  </xs:key>
 </xs:element>
 <xs:element name="presentation">
  <xs:complexType>
   <xs:sequence>

    <!-- metadata -->
    <xs:element ref="metadata" minOccurs="0"/>

    <!-- Local definition - corresponds to BGTheme class -->
    <xs:element ref="theme"/>

    <!-- Slide list element describes the real sequence of slides
     to be shown.-->
    <xs:element ref="slide-list"/>

    <!-- Corresponds to uiSTate member -->
    <xs:element ref="ui-state" />

   </xs:sequence>

   <xs:attribute name="version" type="xs:string"/>
   <xs:attribute name="self-contained" type="xs:boolean"/>
  </xs:complexType>

 </xs:element>

<!--
```

```
 Dictionary schema -
 Auxiliary schema for APXL to describe "dictionaries"(<dict>) and arrays

 Integrated with the main schema to provide simpler validation.
_____________________________________________________________________________________
-->
<!--
_____________________________________________________________________________________
Section D1. "Keyed" extensions of simple types
_____________________________________________________________________________________
-->
 <xs:complexType name="keyed-empty-type">
  <xs:attribute name="key" type="key-type"/>
 </xs:complexType>

 <xs:complexType name="keyed-color-type">
  <xs:simpleContent>
   <xs:extension base="color-type">
    <xs:attribute name="key" type="key-type"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
 <xs:complexType name="keyed-boolean-type">
  <xs:simpleContent>
   <xs:extension base="xs:boolean">
    <xs:attribute name="key" type="key-type"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>

 <xs:complexType name="keyed-number-type">
  <xs:simpleContent>
   <xs:extension base="xs:float">
    <xs:attribute name="key" type="key-type"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
 <xs:complexType name="keyed-integer-type">
  <xs:simpleContent>
   <xs:extension base="xs:integer">
    <xs:attribute name="key" type="key-type"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>

 <xs:complexType name="keyed-string-type">
  <xs:simpleContent>
   <xs:extension base="xs:string">
    <xs:attribute name="key" type="key-type"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>

 <xs:complexType name="keyed-path-type">
  <xs:simpleContent>
   <xs:extension base="path-type">
    <xs:attribute name="key" type="key-type"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>

 <xs:complexType name="keyed-file-name-type">
```

```
  <xs:simpleContent>
   <xs:extension base="file-name-type">
    <xs:attribute name="key" type="key-type"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>

 <xs:complexType name="keyed-point-type">
  <xs:simpleContent>
   <xs:extension base="point-type">
    <xs:attribute name="key" type="key-type"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>

 <xs:complexType name="keyed-size-type">
  <xs:simpleContent>
   <xs:extension base="size-type">
    <xs:attribute name="key" type="key-type"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>

 <xs:complexType name="keyed-rect-type">
  <xs:simpleContent>
   <xs:extension base="rect-type">
    <xs:attribute name="key" type="key-type" />
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
 <xs:complexType name="keyed-range-type">
  <xs:simpleContent>
   <xs:extension base="range-type">
    <xs:attribute name="key" type="key-type" />
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
 <xs:complexType name="keyed-transformation-type">
  <xs:simpleContent>
   <xs:extension base="transformation-type">
    <xs:attribute name="key" type="key-type" />
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>

 <xs:complexType name="keyed-url-type">
  <xs:simpleContent>
   <xs:extension base="xs:anyURI">
    <xs:attribute name="key" type="key-type"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
<!--
_____________________________________________________________________________
Section D2. Types for complex keyed types used only in dictionaries
_____________________________________________________________________________
-->
 <xs:complexType name="keyed-reference-type">
  <xs:attribute name="id-ref" type="xs:IDREF"/>
  <xs:attribute name="key" type="key-type"/>
 </xs:complexType>
 <xs:complexType name="keyed-font-type">
```

```
  <xs:attribute name="name" type="xs:string"/>
  <xs:attribute name="font-size" type="xs:float"/>
  <xs:attribute name="transformation" type="transformation-type"/>
  <xs:attribute name="key" type="key-type"/>
 </xs:complexType>
 <xs:complexType name="number-handler-type">
  <xs:attribute name="rounding-mode" type="rounding-mode-type"/>
  <xs:attribute name="raise-on-exactness" type="xs:boolean"/>
  <xs:attribute name="raise-on-divide-by-zero" type="xs:boolean"/>
  <xs:attribute name="raise-on-overflow" type="xs:boolean"/>
  <xs:attribute name="raise-on-underflow" type="xs:boolean"/>
  <xs:attribute name="scale" type="xs:integer"/>
 </xs:complexType>
 <xs:complexType name="date-formatter-type">
  <xs:attribute name="allows-natural-language" type="xs:boolean"/>
  <xs:attribute name="key" type="key-type"/>
 </xs:complexType>
 <xs:complexType name="number-formatter-type">
  <xs:sequence>
   <xs:element name="negative" type="xs:string" minOccurs="0"/>
   <xs:element name="positive" type="xs:string" minOccurs="0"/>
   <xs:element name="zero" type="xs:string" minOccurs="0"/>
   <xs:element name="nil" type="xs:string" minOccurs="0"/>
   <xs:element name="nan" type="xs:string" minOccurs="0"/>
   <xs:element name="thousand-separator" type="xs:string" minOccurs="0"/>
   <xs:element name="float-separator" type="xs:string" minOccurs="0"/>
   <xs:element name="rounding-behavior" type="number-handler-type" minOccurs="0"/>
   <xs:element name="min" type="xs:float" minOccurs="0"/>
   <xs:element name="max" type="xs:float" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="key" type="key-type"/>
 </xs:complexType>

<!--
_____________________________________________________________________________
Section D3. Dictionary and array structures and related groups
_____________________________________________________________________________
-->
 <xs:complexType name="dictionary-type">
  <xs:sequence>
   <xs:group ref="general-data-group" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="key" type="key-type"/>
 </xs:complexType>

 <xs:element name="array" type="dictionary-type"/>

 <xs:element name="dict" type="dictionary-type">
  <xs:key name="dict-key">
   <xs:selector xpath="*"/>
   <xs:field xpath="@key"/>
  </xs:key>
 </xs:element>

<!-- Possible dictionary content as a group. -->
 <xs:group name="dict-data-group">
  <xs:choice>
   <xs:element ref="dict"/>
   <xs:element ref="array"/>
   <!-- elements used exclusively inside dictionaries -->
   <xs:element name="null" type="keyed-empty-type"/>
   <xs:element name="number" type="keyed-number-type"/>
```

```
    <xs:element name="string" type="keyed-string-type"/>

    <xs:element name="transformation" type="keyed-transformation-type"/>
    <xs:element name="bezier-path" type="keyed-path-type"/>
    <xs:element name="color" type="keyed-color-type"/>

    <xs:element name="data" type="keyed-file-name-type"/>
    <xs:element name="image-binary" type="keyed-file-name-type"/>

    <xs:element name="file-wrapper" type="keyed-file-name-type"/>

    <xs:element name="attributed-string" type="text-content-type"/>
    <xs:element name="point" type="keyed-point-type"/>
    <xs:element name="size" type="keyed-size-type"/>

    <xs:element name="rect" type="keyed-rect-type"/>
    <xs:element name="range" type="keyed-range-type"/>
    <xs:element name="url" type="keyed-url-type"/>
    <xs:element name="reference" type="keyed-reference-type"/>
    <xs:element name="font" type="keyed-font-type"/>
    <xs:element name="date-formatter" type="date-formatter-type"/>
    <xs:element name="number-formatter" type="number-formatter-type"/>

   </xs:choice>
  </xs:group>

  <xs:group name="content-data-group">
   <xs:choice>
    <xs:group ref="dict-data-group"/>
    <xs:group ref="drawable-group"/>
    <xs:group ref="styles-group"/>
    <xs:element ref="content"/> <!-- exclusively for Tables plugin data -->
   </xs:choice>
  </xs:group>

  <xs:group name="general-data-group">
   <xs:choice>
    <xs:group ref="content-data-group"/>
    <!-- plugin:namespace elements may go in here as well -->
    <xs:any namespace="http://apple.com/schemas/APXLPlugins" processContents="lax"/>
    <!-- -->
   </xs:choice>
  </xs:group>
 </xs:schema>
```

Back to top

Change History

[Feb 12 2003] Initial version of Technical Note.

Back to top

Downloads

Acrobat version of this Note (200K) Download

APXL.xsd (in the file named tn2067.hqx, 12K) Download

Back to top

Technical Notes by Date | Number | Technology | Title
Developer Documentation | Technical Q&As | Development Kits | Sample Code