

# XML Templates for Tables and Descriptors in MPEG-2 Private-Section Format

## FEATURES

- XML-based template for the decoding of public and private tables and descriptors in MPEG-2 private-section format.

## APPLICATIONS

- Decoding of privately defined descriptors
- Decoding of privately defined tables

### *Example template for bouquet\_list\_descriptor*

```
<?xml version="1.0" encoding="UTF-8"?>
<Mp2TableTemplate xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xsi:noNamespaceSchemaLocation="file:Schema/Mp2TableTemplate.xsd">
  <DescriptorTempl tag="145" name="bouquet_list_descriptor" standard="dvb">
    <DescriptorPresentation>
      <LongName str="Bouquet List Descriptor"/>
    </DescriptorPresentation>
    <DescriptorBody>
      <Field length="8" name="descriptor_tag" encoding="uimsbf" semantics="descriptor_tag">
        <FieldPresentation>
          <Prefix str="Descriptor tag"/>
          <Format str="0x%02X"/>
        </FieldPresentation>
      </Field>
      <Field length="8" name="descriptor_length" encoding="uimsbf"
        semantics="descriptor_length"/>
      <Loop length_field="implicit">
        <LoopPresentation>
          <NoLoopHeader/>
          <LoopEmpty str="No bouquets listed"/>
          <LoopEntry>
            <EntryField entry_field="bouquet_id">
              <FieldPresentation>
                <Prefix str="Bouquet"/>
              </FieldPresentation>
            </EntryField>
          </LoopEntry>
        </LoopPresentation>
      <Body>
        <Field length="16" name="bouquet_id" encoding="uimsbf"/>
      </Body>
    </Loop>
  </DescriptorBody>
</DescriptorTempl>
</Mp2TableTemplate>
```

## RELEVANT PRODUCTS

Type	Description
DTC-320	StreamXpert MPEG-2 Transport-Stream Analyser Software

Copyright © 2003-2004 by DEKTEC Digital Video B.V.

DEKTEC Digital Video B.V. reserves the right to change products or specifications without notice. Information furnished in this document is believed to be accurate and reliable, but DekTec Digital Video assumes no responsibility for any errors that may appear in this material.

## Table of Contents

<b>1. Introduction .....</b>	<b>3</b>	<b>5. Descriptor Template .....</b>	<b>8</b>
1.1. Purpose of this Document.....	3	<b>6. Syntax Elements .....</b>	<b>9</b>
1.2. Intended Audience .....	3	6.1. Field .....	9
1.3. Interface Specification .....	3	6.2. If .....	9
1.4. Document Overview .....	3	6.3. Loop .....	10
1.5. Definitions, Acronyms and Abbreviations .....	3	6.4. Descriptor .....	10
1.6. References.....	3	6.5. Macro .....	10
<b>2. Location of Template Files .....</b>	<b>4</b>	6.6. EndOfHeader .....	10
2.1. Schema Definition File .....	4	6.7. MultiStringStruct .....	10
2.2. Standard Templates .....	4	<b>7. Presentation Elements .....</b>	<b>11</b>
2.3. Custom Templates .....	4	7.1. Field Presentation.....	11
<b>3. Overall Template Structure.....</b>	<b>5</b>	7.2. Loop Presentation .....	12
3.1. Default Presentation Attributes .....	5	7.3. Descriptor Presentation.....	13
3.2. Table Tree.....	5	7.4. Value Mapping .....	13
<b>4. Table Template .....</b>	<b>7</b>		

## 1. Introduction

### 1.1. Purpose of this Document

This document defines and explains the structure table- and descriptor- templates.

The DTC-320 StreamXpert software uses templates internally for defining standard DVB and ATSC tables and descriptors. The user may add private tables and descriptors using custom template files.

### 1.2. Intended Audience

This document is intended to be used by StreamXpert users that wish to write templates for custom descriptors and/or tables.

### 1.3. Interface Specification

Table and descriptor templates are encoded in XML documents. The permissible format of such XML files is defined by an XML Schema, written in XML Schema Definition Language [XSDL].

The schema is contained in the following .xsd file: [Mp2TableTempl.xsd](#).

The .xsd file is available as a separate file, enabling:

- Conveniently viewing the schema, using a XML authoring tool;
- Conveniently editing (by hand) of the configuration file, using a schema-driven XML editor;
- Automatic validation of the structure of the configuration file using a XSDL-enabled validating parser.

The elements and attributes in the schema are described in plain text in this document. The following conventions are used:

#### Element

XML elements are set in a green, bold font.

#### Attribute

Attributes of XML elements are set in brown, with character positions "expanded by 1 pt".

#### Enumeration

Enumeration values are set in blue italics.

### 1.4. Document Overview

Chapter 2 describes (for humans) the semantics of elements and attributes defined in the XML Schema for the table and descriptor templates.

Chapter 3 provides an example of a template for a custom descriptor.

### 1.5. Definitions, Acronyms and Abbreviations

#### XML

Extensible Markup Language. A markup language defined by the W3C that provides a strict set of standards for document syntax.

#### XSDL

XML Schema Definition Language. A schema definition language under development by the W3C Schema working group. Expressed in XML document syntax, XSDL is designed to support an extensible data typing system, inheritance, and namespaces.

#### W3C

The World Wide Web Consortium, which sets Web-oriented standards like XML.

### 1.6. References

#### [XSDL]

XML Schema Definition Language, W3C, Candidate Recommendation, October 24, 2000

## 2. Location of Template Files

### 2.1. Schema Definition File

The schema definition file `Mp2TableTempl.xsd` is included in the StreamXpert installation package. It will be installed in the following directory:

`<streamxpert_install_dir>\Custom\Schema`

### 2.2. Standard Templates

The StreamXpert uses three standard templates to cover tables and descriptors defined in:

- MPEG-2 Systems
- DVB-SI
- ATSC PSIP

The standard templates are included as resources in the StreamXpert executable.

Tables and descriptors defined in the standard templates can be overruled by definitions in custom templates (ed.: not implemented yet).

### 2.3. Custom Templates

Custom template files can be copied to:

`<streamxpert_install_dir>\Custom`

Upon start-up, the StreamXpert will read all `.xml` files found in this directory.

## 3. Overall Template Structure

The top-level element of each template file is **Mp2TableTemplate**. This is true for the main MPEG-2, DVB and ATSC template files, as well as for custom template files.

The syntax of this element is defined entirely by the **Mp2TableTempl.xsd** schema.

### Mp2TableTemplate

The first optional child is **DefaultPresentation** element, which defines default presentation attributes. This element may be used in the main MPEG-2 template only.

The second optional child is **TableTree**, which defines the structure of the Table Tree for tables defined in the current template. Immediately following these first two child elements, a list of table templates (**TableTempl**), descriptor templates (**DescriptorTempl**) value mappings (**ValuMapping**) and group comments (**GroupComment**) may occur, in arbitrary sequence.

### 3.1. Default Presentation Attributes

This children of this element define the default presentation format when the template does not contain explicit formatting instructions.

The default presentation attributes may be defined in the main MPEG-2 template only.

#### DefaultPresentation

#### PrefixFmt

#### StatFmt

#### LoopHeaderFmt

#### LoopEmptyFmt

#### TableHeaderFmt

#### DescHeaderFmt

#### SvcNameFmt

#### TableTreeFmt

#### MultStringStructEmptyFmt

#### TableTreeFmt

Default formatting of the top-level Table-Tree nodes, as defined by Table-Tree elements.

#### MultStringStructFmt

#### MultStringStructHdrFmt

### 3.2. Table Tree

The Table Tree defines the top-level nodes that may occur in the display tree with decoded tables.

**Note** Tables that do not occur in the input stream will not be shown in the Table Tree.

#### TableTree

This element recursively defines the Table Tree. Intermediate nodes have **TableTree** child elements. Leaf nodes don't have children, but do have the **table\_id** attribute defined.

## table\_id

Including this attribute will instruct the StreamXpert to insert the decoded table with this table ID at the current location in the Table Tree.

If the table has no subtables, then the current Table-Tree node will be *replaced* by the root of the decoded table. In this case, this Table-Tree node must be a leaf node and no **TableTree** children may be defined. If the table does have subtables, then the current Table-Tree node will be inserted in the display tree and the subtables will be inserted below this node.

## binding

Either *service* or *ts*.

---

## TableTree/NodeName

Type: **DispString**

Defines the table name to be displayed for this node in the Table Tree.

---

## TableTree/TableEntry

Type: **TFieldPresentation**

For tables with subtables: defines the formatting of the table entry used as header of the subtable.

## **4. Table Template**

## 5. Descriptor Template

A descriptor template defines the syntax and presentation for a single descriptor.

---

### DescriptorTempl

Top-level element defining a descriptor.

The first optional child **MayOccurIn** defines the tables in which the descriptor may occur. The next mandatory child is **DescriptorPresentation** (described in §7.3) which defines the overall display properties of the descriptor. The descriptor syntax is defined in **DescriptorBody**.

#### tag

Type: **nonNegativeInteger**

Defines the descriptor tag.

#### name

Type: **string**

Descriptor name for internal purposes only.

The display name of the descriptor is defined under **DescriptorPresentation**.

#### standard

Type: **string**

Indicates whether the descriptor tag is recognised in DVB tables only (attribute value **dvb**), in ATSC tables only (attribute value **atsc**) or in all tables (attribute value **mpeg**) .

---

### DescriptorBody

Type: **Syntax**

The main body of a descriptor template, defining the descriptor's syntax and the presentation of the syntax elements.



## 6. Syntax Elements

Tables and descriptors are defined using elements of type **Syntax**. This is the fundamental type for defining the actual structure of a bit stream.

A **Syntax**-element consists of an arbitrary sequence of the following elements: **Field**, **If**, **Loop**, **Descriptor**, **Macro**, **EndOfHeader** and **MultiStringStruct**.

### 6.1. Field

The core element of syntax descriptions that describes the properties of a single field.

#### Field

The properties of a single data field in a table or a descriptor.

#### length

Type: **positiveInteger**  
Length of field in bits.

#### name

Type: **string**  
Field name, for cross-reference purposes. The display name of the field is defined under **FieldPresentation**.

#### encoding

Type: **string**  
Field format as defined in MPEG-2. One of *bslbf*, *rpchbf*, *string*, *tcimsbf*, *uimsbf*.

#### semantics

Type: **string**  
Optional field that indicates whether the field has special semantics. If this attribute is not defined, no special semantics are associated with the field.

The following values are defined:

*descriptor\_tag*, *descriptor\_length*,  
*table\_id*, *table\_id\_ext*,  
*section\_syntax\_indicator*, *section\_length*,  
*section\_number*, *last\_section\_number*,  
*version\_number*, *current\_next\_indicator*,  
*last\_table\_id*, *segment\_last\_section\_number*,  
*crc\_32*, *stream\_type*,  
*reserved*, *reserved\_future\_use*

The semantics associated with these values correspond one-to-one with the semantics

of the same-named fields defined in MPEG-2 and DVB-SI.

#### length\_field

Type: **string** (field-name reference)  
For uncompressed, counted strings: name of field that specifies the length of the string.

#### fixed\_length

Type: **positiveInteger**  
For fixed-length strings: number of characters in fixed-length string.

#### string\_type

Type: **string**  
Type of string encoding: *dvb\_text* | *utf16* | *atsc\_compressed\_text*.

#### compression\_type\_field

Type: **string** (field-name reference)  
Special-purpose attribute for ATSC compressed strings: field that specifies the compression type.

#### mode\_field

Type: **string** (field-name reference)  
Special-purpose attribute for ATSC compressed strings: field that specifies the compression mode.

### 6.2. If

Describes a conditional statement in a bit-stream description.

#### If

Main element of an "if".  
Children are **Condition**, **Then** (both mandatory) and optionally **Else**.  
An If statement has no associated presentation elements.

#### If/Condition

Specifies the condition of the If statement.  
The child element defines the condition type. At the moment two condition types are defined: either **CompareWithConst** or **CompareStreamType**.

#### If/Condition/CompareWithConst

Compare a field in the bit stream with a constant.

## field

Type: **string**

Field name to be compared with a constant.

## comp\_op

Type: **string**

Operator, either *equals* or *not\_equal*.

## const

Type: **int**

Constant with which field value is to be compared.

## If/Condition/CompareStreamType

Special purpose comparison mode for descriptors in the PMT: compare stream\_type of the associated elementary stream with a constant.

## comp\_op

Type: **string**

Operator, either *equals* or *not\_equal*.

## const

Type: **int**

Constant with which stream\_type is to be compared.

## If/Then

Type: **Syntax**

Mandatory "then" clause of an if statement.

## If/Else

Type: **Syntax**

Optional "else" clause of an if statement.

## 6.3. Loop

The **Loop** element enables the specification of loop construction in a bit-stream description.

### Loop

Main element of a loop. The presentation of the loop is controlled by child element **LoopPresentation**. The body of the loop is described in child element **Body**.

## length\_field

Type: **string**

Field name that defines the length of the loop (when attribute **length\_type** is *length\_in\_bytes* or not defined), or the num-

ber of loop iterations (when **length\_type** is *count*).

## length\_type

Type: **string**

Specifies whether the length of the loop is expressed in number of bytes (**length\_type** = *number\_of\_bytes*), or in number of iterations (**length\_type** = *count*).

If **length\_type** is not defined, the value defaults to *number\_of\_bytes*.

## Loop/Body

Type: **Syntax**

Bit-stream description of the body of the loop.

## 6.4. Descriptor

The **Descriptor** element indicates a location in a table where a descriptor may be located. This element does not have attributes and may not have child nodes.

## 6.5. Macro

Macro facility.

## 6.6. EndOfHeader

The **EndOfHeader** element is used as a marker to indicate the end of the section header.

## 6.7. MultiStringStruct

Special purpose structure used in the decoding of ATSC multi-string structures.

## 7. Presentation Elements

Presentation elements enable context-specific presentation instructions for bit-stream constructions, like fields or loops.

Generally speaking, field-presentation elements are optional. The absence of field-presentation elements may signify one of two things:

- The syntax element is not presented at all;
- Default presentation attributes are used.

### 7.1. Field Presentation

#### TFieldPresentation

A type that specifies how a data field is presented in the display tree. If no element of this type is present in the presentation part of a given field, then the field will not be shown in the display tree.

The optional children of **TFieldPresentation** specify additional presentation information: **Prefix**, **Format** and **Mapping**.

#### Prefix

Type: **DispString**

The display string to be put in front of the field value.

#### Format

Type: **DispString**

Specifies the formatting to be applied to the field value. See also the description of type **DispString**.

#### str

Specifies the formatting to be applied to a value.

When attribute **str** is left empty, the field value is presented in decimal format.

Otherwise, **str** is interpreted as *printf*-style format specifier, e.g:

**str**="0x%02x"

to display the value as two hexadecimal digits preceded by "0x".

Next to the standard *printf* format specifiers, the following special formatting codes have been defined:

**%m.nB**

BCD encoded field with m digits before the comma and n digits after the comma, e.g.:

**str**="%4.4B MHz"

This format is used amongst others for formatting the frequency in the delivery system descriptors.

#### %DU

Duration encoded as 6 digits, 4-bit BCD. Field duration in the EIT uses this format.

#### %MU

40-bit field with Modified Julian Date (MJD) and Universal Time Co-ordinated (UTC) are encoded, e.g.:

**str**="%MU"

Field start\_time in the EIT uses this format.

#### %nnT

Text to be wrapped over multiple lines, with a maximum line length of nn. For example,

**str**="%30T"

will format the text in lines of maximum 30 characters.

#### %GPU

GPS UTC time: seconds expired since 00:00:00 Jan 6 1980.

Example: start\_time in the EIT (PSIP).

#### %EID

Present a channel\_etm\_id or event\_etm\_id field (PSIP).

#### color

Optional: color in which the value will be displayed.

#### fontattr

Optional: font attributes to be applied to the value.

#### Mapping

Empty element that specifies a value-to-string mapping to be applied to the value.

#### name

Name of the value mapping to be used. The following standard value mappings are defined:

## ISO639-2

Mapping from 3-character ISO 639-2 Language Code, e.g. "dut", to a language string, e.g. "Dutch".

**NOTE:** attribute **mode** cannot be used with this value mapping.

### service\_name

Pseudo-mapping for translating a service ID to a service name, as decoded from the SDT.

If **name** is not one of the standard mappings, the value mapping must be defined in the table-template file. If not, an error will be generated.

### mode

This attribute defines the "mode" in which the value and the mnemonic (mapped value) are displayed.

### mnem

Just the mnemonic is shown. Example:

Service type: **Teletext service**

### val

Just show the value. Not really useful, because the same effect can be achieved without mapping. Used internally in the table decoder as back-up in case e.g. a service name is not available.

Example:

Service type: **3**

### val\_mnem

Put decoded mnemonic after value, without parentheses. Example:

Service type: **3 Teletext service**

### val\_mnem\_pars

Put decoded mnemonic after value, between parentheses. Example:

Service type: **3 (Teletext service)**

## Concatenate

This presentation flag is only used for concatenated descriptors. If specified, this flag indicates that the field values from the concatenated descriptors have to be presented as a comma-separated list.

## 7.2. Loop Presentation

### LoopPresentation

This element contains the presentation instructions for this loop. The following child elements may occur: **LoopHeader**, **NoLoopHeader**, **LoopEmpty** and **LoopEntry**.

### LoopPresentation/LoopHeader

Type: **DispString**

String that is put at the top of the loop in the display tree.

### LoopPresentation/NoLoopHeader

If this empty element is included, no header string will be inserted above the loop.

Obviously, the use of **NoLoopHeader** and **LoopHeader** is mutual.

### LoopPresentation/LoopHeader

Type: **DispString**

String that is displayed when the loop does not contain any entry.

### LoopPresentation/LoopEntry

Defines the formatting of the header of the loop-entries. Either **Fixed** or **EntryField**.

### LoopEntry/Fixed

Type: **DispString**

Loop-entry header is a fixed display string.

### LoopEntry/EntryField

Display a field occurring in the loop as loop-entry header.

An optional child element **FieldPresentation** may be used to specify special formatting instructions.

### entry\_field

Field in the loop that should be displayed as header for the loop entry.

## 7.3. Descriptor Presentation

### DescriptorPresentation

Defines the display name of the descriptor (in **LongName**) and whether strings within the descriptor can be concatenated (child element **Concatenate**), for special-purpose application in the EIT.

### LongName

Type: **DispString**

Required. Specifies the name of the descriptor, as it will appear in the decoded display tree.

### Concatenate

Empty element that can be used to concatenate multiple consecutive descriptors (of the same type) into a single Display tree entry. This element has been designed specifically for the extended event descriptor in the EIT.

Minimum value (inclusive) to be mapped.

**val\_max**

Type: **nonNegativeInteger**

Maximum value (inclusive) to be mapped.

### ValueMapping/ValueRange/ ValString

Display string associated with the value range.

## 7.4. Value Mapping

A value mapping associates display strings to numerical values.

### ValueMapping

Sequence of **Value** and/or **ValueRange** elements.

### ValueMapping/Value

Single value-to-string mapping.

**value**

Type: **nonNegativeInteger**

Value to be mapped.

### ValueMapping/Value/ValString

Type: **DispString**

Display string associated with the value.

### ValueMapping/ValueRange

Range of values to be mapped to the same string.

**val\_min**

Type: **nonNegativeInteger**