

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 © 2003-2021 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

1. Introduction	3	5.1. MayOccurIn	8
1.1. Purpose of this Document	3	5.2. DescriptorPresentation	8
1.2. Intended Audience	3	5.3. Variable	8
1.3. Interface Specification	3	5.4. DescriptorBody	8
1.4. Document Overview	3	6. Syntax Elements	9
1.5. Definitions, Acronyms and Abbreviations	3	6.1. Field	9
1.6. References	3	6.1.1. Semantics	10
2. Location of Template Files	4	6.2. If	10
2.1. Schema Definition File	4	6.3. Loop	11
2.2. Standard Templates	4	6.4. Descriptor	11
2.3. Custom Templates	4	6.5. Macro	11
3. Overall Template Structure	5	6.6. EndOfHeader	12
3.1. Type: DispString	5	6.7. MultiStringStruct	12
3.2. Default Presentation Attributes	5	6.8. VariableAssignment	12
3.3. Table Tree	6	7. Presentation Elements	14
4. Table Template	7	7.1. Field Presentation	14
4.1. TableId	7	7.2. Loop Presentation	15
4.2. Variable	7	7.3. Descriptor Presentation	16
4.3. TableBody	7	7.4. VariablePresentation	16
5. Descriptor Template	8	8. Value mapping	17
		9. Icons	18

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. Knowledge of the mpeg standard is assumed (See ISO/IEC 13818-1 table 2-35 – Private section for more information about how a table is build in a transport stream).

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.

Type

Elements and attributes have a specified type, which is set in red. Attribute types can be e.g. string or integer. Element type consist of a description of expected attributes and/or child elements.

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:

`C:\Users\Public\PublicDocuments\DekTec\
StreamXpert\Profiles\Schema\
Tr101290Config.xsd`

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.

2.3. Custom Templates

Custom template files can be copied to:

`C:\Users\Public\PublicDocuments\DekTec\
StreamXpert\Profiles`

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 next optional child's are **TableTree**, and **SectionPidList**. **TableTree** defines the structure of the Table Tree for tables defined in the current template. **SectionPidList** defines the initial PID to decode for tables.

Immediately following these child elements, a list of table templates (**TableTempl**), descriptor templates (**DescriptorTempl**), macro templates (**MacroTempl**), value mappings (**ValueMapping**) and group comments (**GroupComment**) may occur in arbitrary sequence.

3.1. Type: DispString

The type DispString describes how a text is presented. The definition consists of the following attributes. When an attribute is not specified a default value is used.

str

Type: **string**
Text string

color

Type: **string**
Color of the text in the form of a 6 digit hexadecimal representing RGB: "#RRGGBB"

fontattr

Type: **string**
Fontattribute of the text, which can be empty or "**bold**"

gen_icon

Type: **integer**

Icon for the display string, see §0.

3.2. Default Presentation Attributes

The 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

DefaultPresentation defines the presentation attributes (bold, colour, icon) for the various display elements when no attributes are given in the presentation elements.

PrefixFmt

Type: **DispString** (See §3.1)
First part of the presentation element, which contains the field description.

ValueFmt

Type: **DispString** (See §3.1)
Default format for the middle part of the display element, which contains the field value.

StatFmt

Type: **DispString** (See §3.1)
Default format for the right part of the display element, which contains statistics from the stream analyser.

LoopHeaderFmt

Type: **DispString** (See §3.1)
Default prefix formatting for loop headers

LoopEmptyFmt

Type: **DispString** (See §3.1)
Default prefix formatting for the string displayed when the loop contains no elements.

TableHeaderFmt

Type: **DispString** (See §3.1)
Default prefix formatting for table names.

DescHeaderFmt

Type: **DispString** (See §3.1)

Default prefix formatting for descriptor names.

SvcNameFmt

Type: **DispString** (See §3.1)
Default value formatting when displaying the name of the service. Used in **mapping** `name="service_name"` in **FieldPresentation**.

TableTreeFmt

Type: **DispString** (See §3.1)
Default prefix formatting of the top-level Table-Tree nodes, as defined by Table-Tree elements.

MultiStringStructFmt

Type: **DispString** (See §3.1)
Default prefix formatting for MultiString-Struct fields.

MultiStringStructHdrFmt

Type: **DispString** (See §3.1)
Default prefix formatting for MultiString-Struct headers.

MultiStringStructEmptyFmt

Type: **DispString** (See §3.1)
Default prefix formatting for MultiString-Struct when no strings are present.

PrefixValSep

Type: **DispString** (See §3.1)
Defines the separation of the prefix and the value.

3.3. 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 chil-

dren, 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** (See §3.1)
Defines the table name to be displayed for this node in the Table Tree.

TableTree/TableEntry

For tables with subtables: defines the formatting of the table entry used as header of the subtable. The TableEntry node has a single FieldPresentation node (See §7.1)

4. Table Template

A table template defines the syntax and presentation of a single table in the transport stream.

Example of a table template:

```
<TableTempl name="my_table" ref_name="M_TBL" standard="dvb">
  <TableId id="123" has_sub_tables="false">
    <TableIdPresentation>
      <ShortName str="MTBL"/>
      <LongName str="My table example"/>
    </TableIdPresentation>
  </TableId>
  <Variable name="my_var" initial_value="456"/>
  <TableBody>
    ...
  </TableBody>
</TableTempl>
```

TableTempl

Top-level element defining a table. Contains the child elements **TableId**, **Variable** and **TableBody**. Attributes:

name

Type: **string**

Table name for internal purposes only. The display name of the descriptor is defined under **TableId**.

ref_name

Type: **string**

Name as which is used in descriptors to indicate that they are allowed within the context of this table. See **MayOccurIn**.

standard

Type: **string**

Indicates to which standard this table belongs, *mpeg*, *dvb*, *atsc* or *isdb*. I.e. when streamXpert is in ATSC mode, only MPEG and ATSC tables are decoded.

4.1. TableId

TableId defines the name and id for this table. More than one **TableId** can be defined for the same table.

id

Type: **8 bit unsigned integer**

Defines the ID for this table. When the table decoder encounters this value in the transport stream, this table template is used

for decoding the table (provided the standard matches)

has_sub_tables

Type: **bool**

When *"true"* the table consists of multiple tables with one or more an table ID extension values.

TableIdPresentation

ShortName

str

Type: **string**

Table name as shown in the tables tree

LongName

str

Type: **string**

Table name for internal purposes only.

4.2. Variable

Sometimes a **Variable** is needed to perform some simple calculations before processing other fields or display a field value. One or more **Variable** can be defined in TableTempl or DescriptorTempl.

name

Type: **string**

Variable name which is used in **VariableAssignment** or as operand in a **VariableAssignment**.

initial_value

Type: **integer** or *"array"*

Initial value for this variable (default 0). Special:

When *initial_value="array"* this variable acts as an array. Each value assigned to this variable is stored in an array. When the variable is read as an operand, the values are read in the same order as they were stored.

4.3. TableBody

TableBody defined the main body of a table template, defining the table's syntax and the presentation of the syntax elements.

5. Descriptor Template

A descriptor template defines the syntax and presentation for a single descriptor. A descriptor occurs in tables like e.g. the network information table (NIT).

Example of a descriptor template:

```
<DescriptorTempl tag="82"
name="my_descriptor" standard="dvb">
  <DescriptorPresentation>
    <LongName str="My descriptor name"/>
  </DescriptorPresentation>
  <DescriptorBody>
    ...
  </DescriptorBody>
</DescriptorTempl>
```

DescriptorTempl

Top-level element defining a descriptor. Contains the child elements **MayOccurIn**, **DescriptorPresentation**, **Variable** and **DescriptorBody**. Attributes:

tag

Type: **8 bit unsigned integer**
Defines the descriptor tag.

tag_ext

Type: **8 bit unsigned integer**
Defines the optional extension descriptor tag. For tag=63 (mpeg) and tag=127 (dvb) the extension descriptor expands the address range for descriptors.

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**) .

5.1. MayOccurIn

Optional field **MayOccurIn** limits the use of this descriptor to specific table(s).

table

Type: **string**

Optional, when present, limits the occurrence of this descriptor to the given table. MayOccurIn can be defined more than once.

5.2. DescriptorPresentation

Child **DescriptorPresentation** defines the display name of the descriptor. See §7.3

5.3. Variable

Optional child **Variable** defines a variable which can be used in syntax elements in the descriptor body. See **TableTemplate** for a detailed description of **Variable**.

5.4. DescriptorBody

DescriptorBody defined 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** in TableBody and DescriptorBody. 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**, **MultiStringStruct**, and **VariableAssignment**.

6.1. Field

Field is the core element of syntax descriptions that describes the properties of a single field or a loop describing a character string.

Field

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

```
<Field length = "8" name = "descriptor_tag"
encoding = "uimsbf" semantics = "descriptor_tag">
  <FieldPresentation>
    <Prefix str = "Descriptor tag"/>
    <Format str = "0x%02X"/>
  </FieldPresentation>
</Field>
```

length

Type: **positiveInteger**

Length of field in bits.

(for encoding=escapedValue, length consists of 3 values separated by a comma)

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 the following:

bslbf (bitfield)

uimsbf (unsigned integer)

tcimsbf (two's complement)

rpchof (CRC32)

spfmsbf (single precision float)

upcrmsf (time)

ip_addr (formats 'length' as 10.0..)

ip_addr_slash (formats 'length' as 10.0../8)

mac_addr (formats as 8A-10-..)

align (special, alligns bitstream to 8bit boundary, length should be 0)

string (extended processing with 'length_field')

escapedValue (variable length coding)

Bslbf+1, **uimsbf+1**, **escapedValue+1** (special, +1 is added to the result)

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. See §6.1.1

length_field

Type: **string** (field-name reference)

For uncompressed, counted strings: name of field that specifies the length of the string. When the string length is the remaining table/descriptor length, use length_field = "implicit". **length_field** cannot be used in combination with **fixed_length**.

fixed_length

Type: **positiveInteger**

For fixed-length strings, number of characters in string. **fixed_length** cannot be used in combination with **length_field**.

string_type

Type: **string**

Type of used string encoding in the bitstream: **dvb_text** | **utf8** | **utf16** | **atsc_compressed_text** | **hex** | **hex16**.

For **dvb_text** the first byte determines the string format, according to the dvb standard. String type **hex** formats the string as hex characters with a '0x' prefix. **hex16** formats the string as 16 hex characters per row.

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.

6.1.1. Semantics

Semantics describe the function of the field in a table/descriptor template. StreamXpert can use this information to properly decode table/descriptor data. The following values are defined:

crc_32 (32bit CRC field in a table),
current_next_indicator
descriptor_length (descriptor length field, should be 8bit)
descriptor_tag (descriptor tag field, 8 bit)
descriptor_tag_ext (descriptor tag extension field, 8 bit, used by descriptor tag=63 (MPEG) and tag=127 (DVB))
last_section_number
last_table_id
private_data_specifier (Sets the private data specifier field for following descriptors in a descriptor loop reserved reserved field)
reserved_future_use
section_length
section_number
section_syntax_indicator
segment_last_section_number
stream_type (PMT table stream_type field, this value is used in **CompareStreamType** when decoding descriptors)
table_id (table id, 8bit)
table_id_ext (table id extension field(s), up to 3 fields can be defined)
version_number
descriptor_number
last_descriptor_number

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.

6.2. If

Syntax element **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.

Condition

Specifies the condition of the If statement.
The child element defines the condition type:
CompareWithConst,
CompareWithConstMult or
CompareStreamType.

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*, *not_equal*, *larger_than* or *smaller_than*.

const

Type: **int**
Constant with which field value is to be compared.

CompareWithConstMult

Compare a field in the bit stream with multiple constants. One or more child elements of **ConstValue** define the values to compare with.

field

Type: **string**
Field name to be compared with a constant.

comp_op

Type: **string**
Operator, either *equals* or *not_equal*.

ConstValue

Value to compare with in CompareWithConstMult. Multiple ConstValue childs can be defined.

const

Type: **int**
The value to compare with.

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.

Then

Type: **Syntax**

Mandatory "then" clause of an if statement.

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 number 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**.

LoopPresentation

LoopPresentation defines the way a loop is presented. It controls the loop header and the loop iteration header. See §7.2 for a detailed description.

Body

Type: **Syntax**

Bit-stream description of the body of the loop, consisting of one or more syntax elements.

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

The **Macro** element inserts a copy of a **MacroTempl**. When variables are used in a macro template, these variables should have been defined in a TableTempl or DescriptorTempl.

name

Type: **string**

Name of the MacroTempl to insert.

length

Type: **string**

Optional, length in bits of macro. Unused bits are skipped.

length_field

Type: **string**

Optional. Field name that defines the length of the macro (in **length** units). When the length of the macro does not match an error message is shown (macro length may be up to **length** - 1 bits smaller).

standard

Type: **string**

Optional, when set the macro will read the bitstream with the given standard. Can be **mpeg**, **dvb**, **atsc** or **isdb**.

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.

name

Type: **string**

MultiStringStruct name, for cross-reference purposes.

display_name

Type: **string**

Display name of the MultiStringStruct.

concat_segments

Type: **boolean**

When true, string sections are concatenated

show_segments

Type: **boolean**

text_width

Type: **positive integer**

When set, maximum width per display row

6.8. VariableAssignment

The **VariableAssignment** element is used to perform simple arithmetic on field values and/or to display these values. Variable only works with integers.

name

Type: **string**

Name of the variable to assign. This variable must be earlier defined in a **TableTempl** or **DescriptorTempl** (as **Variable**). The formula of VariableAssignment is then name = operand1 **operator** operand2.

FieldPresentation

Optional FieldPresentation defines the way a VariableAssignment is presented. See §7.1 for a detailed description.

Operand1

First operand of a VariableAssignment. Contains child element **Field**, **Variable** or **Constant**.

Field

When operand is Field, the name attribute defines the name of the field.

name

Type: **string**
Name of the field.

Variable

When operand is Variable, the name attribute defines the name of the variable.

name

Type: **string**
Name of the variable.

Constant

When operand is Constant, the value attribute defines a value.

value

Type: **integer**
Value of the operand.

Operator

Operation of the VariableAssignment.

operator

Type: **string**

Operator is one of the following operations:

Assignment variable = Operand1

add variable = Operand1 + Operand2

subtract variable = Operand1 - Operand2

multiply variable = Operand1 * Operand2

divide variable = floor(Operand1 / Operand2)

modulus variable = remainder(Operand1 / Operand2)

log2

Operand2 = 0?

variable = floor(log2(Operand1))

Operand2 = 1?

variable = floor(log2(Operand1-1))

Operand2 = 2?

variable = floor(log2(Operand1))-1

Operand2 = 3?

variable = floor(log2(Operand1-1))-1

index

when variable is defined with `initial_value="array"`, each time this variable is assigned a value, this value is stored starting from index 0. When this variable is read in an assignment as operand, the value at 'readindex' is returned and this 'readindex' is incremented. Operator `index` sets this 'readindex' to a custom value.

`get_group_size`

when variable is defined with `initial_value="array"`, each time this variable is assigned a value, this value is stored in an array. Operator `get_group_size` considers each entry as a group count. For a given value 'index' the groupcount entry is returned, assuming the first entry starts at index 0. E.g. for an array 4,2,6,3 an index 7 returns '6', as the first entry [4] represents index 0..3, then [2] 4..5, [6] 6..11 and [3] 12..14.

Operand2

Second operand of a VariableAssignment. Contains child element **Field**, **Variable** or **Constant**. Operand2 has the same syntax as Operand1.

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

If and how a single field is displayed in streamXpert is specified with the **FieldPresentation** element. 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 **FieldPresentation** specify additional presentation information: **Prefix**, **Format** and **Mapping**.

Prefix

Type: **DispString** (See §3.1)

The display string to be put in front of the field value. A **PrefixFmt** is appended when displayed (default ": ").

Format

Type: **DispString** (See §3.1)

Specifies the formatting to be applied to the field value.

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.

The dot in the format string may be replaced by ':'.
%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.

%nn.nnnT

As **%nnT**, however displays a maximum of nnn characters. When the string consists of more than nnn characters, three dots are appended after character nnn and rest of characters are omitted.

%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).

%PCRBASE

Present PCR field.

%PCREXT

Present PCR extension field.

color

Optional: colour 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 predefined value mappings are defined:

ISO639-2

Mapping from 3-character ISO 3166 Language Code, e.g. "dut", to a language string, e.g. "Dutch".

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

ISO3166

Mapping from 3-character ISO 639-2 Country Code, e.g. "nld", to a country string, e.g. "Netherlands".

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.

service_name_via_source_id

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

When **name** is not one of the standard mappings, the value mapping must be defined in the table-template file. See 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

Deprecated, Concatenated is not functional in streamXpert.

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**.

LoopHeader

Type: **DispString**

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

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.

LoopEmpty

Type: **DispString**

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

LoopEntry

Defines the formatting of the header of the loop-entries. The following child elements may occur: **Fixed**, **EntryField**, **IconField**.

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.

LoopEntry/IconField

Type: **String**

Field name which specifies the icon used in the loop header. The field needs to have a Mapping defined which maps the field value to the icon id.

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.

7.4. VariablePresentation

Optional, defines the way how a variable is displayed after assignment/calculation. See **FieldPresentation** for the syntax.

8. Value mapping

Value mapping associates display strings to numerical values. These strings are defined with **ValueMapping**. Then in a presentation **Mapping** can be used to translate field values to a meaningful string. In a loop entry header, **IconField** can be used to select an icon.

Both **Value** as **ValueRange** have a **ValString** child, but **Value** can also use a short form and define the string with the **str/color/fontattr** attributes.

ValueMapping

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

Value

Type: **DispString** (See §3.1)

Type: (Optional) **DispString** (See §3.1)

Single value-to-string mapping.

value

Type: **Integer**

Value to be mapped.

str

Type: **string**

The display string

color

Type: **string**

Optional: color in which the display string will be displayed.

fontattr

Optional: font attributes to be applied to the display string.

gen_icon

Optional: icon used in **IconField**.

ValueRange

Range of values to be mapped to the same string.

val_min

Type: **nonNegativeInteger**

Minimum value (inclusive) to be mapped.

val_max

Type: **nonNegativeInteger**

Maximum value (inclusive) to be mapped.






























ValString

Type: **DispString**

Display string associated with the value.

9. Icons

Icons are defined by number in presentations and value mapping definitions.

0		tree root
1		services root
2		tables root
3		default
4		error
5(6)		video
7(8)		audio
9(10-13, 16-18)		data
14		
15		
19		ghost
20		unreferenced
21		null packet
22		table
23		service tv
24(25)		service radio
26		service teletext
27		service nvod
28		service mosaic
29		service data broadcast
30		
31		
32		
33		
34		
35		
36		
37		
38(39)		conditional access