

Using and Defining TR 101 290 Profiles for the StreamXpert

FEATURES

- XML-based definition of TR 101 290 Profiles
- Fully customisable TR 101 290 rules

APPLICATIONS

- Verifying TR 101 290 compliancy

Example of a custom TR 101 290 profile

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Tr101290Config xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation = "./Schema/Tr101290Config.xsd">
  <Profile name = "Override_Default_DVB_Profile_Sample" base_profile = "Default_DVB_Profile">
    <Priority1 enabled = "true">
      <PID_error enabled = "false"/>
    </Priority1>
    <Priority2 enabled = "true">
      <PCR_repetition_error enabled = "true" max_interval_ms = "60"/>
    </Priority2>
    <Priority3 enabled = "true">
      <SI_repetition_error enabled = "true">
        <TableRepetitionRule name = "PAT_repetition_rule" max_interval_ms = "250">
          <Filter>
            <Pid value = "0"/>
          </Filter>
          <TableIdentifier>
            <TableId value = "0"/>
          </TableIdentifier>
        </TableRepetitionRule>
        <TableRepetitionRule name = "PMT_repetition_rule" max_interval_ms = "1000">
          <TableIdentifier>
            <TableId value = "2"/>
          </TableIdentifier>
        </SI_repetition_error>
      </Priority3>
    </Profile>
  </Tr101290Config>
```

RELEVANT PRODUCTS

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

Copyright © 2005 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	4.2.3. Priority3	8
1.1. Purpose of this Document.....	3	4.3. Global Log Settings.....	10
1.2. Intended Audience.....	3	5. Base XML Types	12
1.3. Interface Specification	3	5.1. Base Priority Type.....	12
1.4. Document Overview	3	5.2. Base Indicator Type.....	12
1.5. Definitions, Acronyms and Abbreviations	3	5.3. Section Indicator Type	12
1.6. References.....	3	5.3.1. Section Repetition Rule	13
2. TR 101 290	4	5.3.2. Section Location Rule	13
3. Location of Profile Files	5	5.4. Table Indicator Type.....	14
3.1. Schema Definition File	5	5.5. Miscellaneous Types.....	15
3.2. Standard Profiles	5	5.5.1. Indicator Log Settings Type.....	15
3.3. Custom Profiles	5	5.5.2. Section Identifier Type.....	16
4. XML Structure	6	5.5.3. Table Identifier Type	16
4.1. Profile	6	5.5.4. PID Filter Type	17
4.2. Priority levels	6	5.5.5. Simple Selection Type	18
4.2.1. Priority1	6	5.5.6. Range Selection Type.....	18
4.2.2. Priority2.....	7	5.6. Data Types	18
		5.6.1. Override Behavior Data Type	18

1. Introduction

1.1. Purpose of this Document

This document defines and explains the structure of the TR 101 290 Profiles XML files.

The StreamXpert (=DTC-320) software uses TR 101 290 profiles internally to validate if the received Transport-Stream complies too the TR 101 290 requirements defined by the profile.

1.2. Intended Audience

This document is intended to be used by StreamXpert users that wish to define custom TR 101 290 profiles.

1.3. Interface Specification

TR 101 290 profiles 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: [Tr101290Config.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 profile files 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

This first chapter provides a quick introduction to this document. The remaining chapters provide a detailed description of the XML profile files. Chapter 2 starts with a short introduction to TR 101 290. Chapter 3 describes the file locations. Chapter 4 provides a description of the TR 101 290 profile file's XML structure. Chapter 5 defines all the custom XML types used in the XML file.

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

[TR 101 290]

Digital Video Broadcasting (DVB) Measurement guidelines for DVB systems, ETSI, TR 101 290 V1.2.1, May, 2001

2. TR 101 290

This section provides a short introduction about TR 101 290.

The ETSI TR 101 290 document describes a number of common recommendations and measurement techniques for MPEG-2 systems. The goal of it is to standardise the way transport-stream parameters are measured and to allow different measurement results, from different measurement equipment, to be compared with each other.

For verifying the syntax and information consistency of MPEG-2 Transport-Stream the TR 101 290 document describes a set of rules/recommendations, called indicators, for which a transport-stream should be tested.

These indicators are divided into three Priority levels:

- Priority Level 1: specifies the minimum set of indicators a Transport-stream should comply with to be able to decode. Failing to comply with one of these indicators means the transport-stream cannot be decoded.
- Priority Level 2: specifies a set of indicators that are highly recommended for smooth decoding of the transport-stream contents. Failing to comply with one of these indicators will very likely influence the decodability.
- Priority Level 3: specifies a set of indicators that have no influence on the decodability. The indicators at this priority level are intended to ensure the availability required/recommended (P)SI information.

For a detailed description of all the indicators per priority level please refer to the ETSI TR 101 290 document.

3. Location of Profile Files

3.1. Schema Definition File

The schema definition file `Tr101290Config.xsd` is included in the StreamXpert installation package. It will be installed in the following directory:

```
<streamxpert_install_dir>\Profiles\Schema
```

3.2. Standard Profiles

The StreamXpert includes two standard TR 101 290 profiles namely: "Default_DVB_Profile" and "Default_ATSC_Profile".

The default profiles can be overridden by defining a custom profile which is derived from one of the two default profiles.

3.3. Custom Profiles

Custom template files can be copied to:

```
<streamxpert_install_dir>\Profiles
```

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

4. XML Structure

This section describes the XML structure of a TR 101 290 Profile file.

The top-level element of each profile file is **Tr101290Config** element. The syntax of this element is defined entirely by the **Tr101290Config.xsd** schema.

Tr10290Config

This is the root element for a profile XML file. It contains one or more **Profile** child elements. The **Profile** elements contain the actual definition of a TR 101 290 indicators/rules.

NOTE: this element must contain at least one **Profile** child element.

4.1. Profile

Each TR 101 290 Profile document contains one or more profiles.

Profile

A profile defines a set of TR 101 290 rules that the StreamXpert should check for. A **Profile** element contains a sequence of four child elements: **Priority1**, **Priority2**, **Priority3** and a **GlobalLogSettings** element.

name

Type: **string**, Use: **required**

Name used for the profile. Note: the name must uniquely identify a profile.

base_profile

Type: **string**, Use: **optional**

Name of the base profile (i.e. the profile this profile is derived from). Use this attribute when you want to overrule an existing profile (e.g. one of the default profiles).

Priority1

The **Priority1** element defines the TR 101 290 priority 1 level indicators/rules. See §4.2.1 for a detailed description.

Priority2

The **Priority2** element defines the TR 101 290 priority 2 level indicators/rules. See §4.2.2 for a detailed description.

Priority3

The **Priority3** element defines the TR 101 290 priority 3 level indicators/rules. See §4.2.3 for a detailed description.

GlobalLogSettings

The **GlobalLogSettings** element defines the global settings for the log TR 101 290 file. See §4.3 for a detailed description.

4.2. Priority levels

This section provides a description of the three priority child elements of a **Profile**.

4.2.1. Priority1

Priority1

Type: **TBasePriority**

The **Priority1** element defines the TR 101 290 priority level 1 indicators/rules. It is derived from the **TBasePriority** (see §5.1) type. It inherits all attributes from this type and extends the type by adding the following sequence of optional child elements: **TS_sync_loss**, **Sync_byte_error**, **PAT_error_2**, **Continuity_count_error**, **PMT_error_2** and **PID_error**.

TS_sync_loss

Type: **TBaseIndicator**

This element controls the **TS_sync_loss** indicator. It is a type definition of the **TBaseIndicator** type (see §5.1) and inherits all attributes and child elements from this type.

Sync_byte_error

Type: **TBaseIndicator**

This element controls the **Sync_byte_error** indicator. It is a type definition of the **TBaseIndicator** type (see

§5.1) and inherits all attributes and child elements from this type.

PAT_error_2

Type: **TSectionIndicator**

This element controls the `PAT_error_2` indicator. It is a type definition of the **TSectionIndicator** type (see §5.3) and inherits all its attributes and child elements from this type.

Continuity_count_error

Type: **TBaseIndicator**

This element controls the `Continuity_count_error` indicator. It is a type definition of the **TBaseIndicator** type (see §5.1) and inherits all attributes and child elements from this type.

PMT_error_2

Type: **TSectionIndicator**

This element controls the `PMT_error_2` indicator. It is a type definition of the **TSectionIndicator** type (see §5.3) and inherits all its attributes and child elements from this type.

PID_error

Type: **TBaseIndicator**

This element controls the `PID_error` indicator. It is a type definition of the **TBaseIndicator** type (see §5.1) and inherits all attributes and child elements from this type. Next to the inherited attributes and child elements it extends the **TBaseIndicator** type with optional list of **StreamTypeTimeout** child element and the `time_out_ms` attribute.

time_out_ms

Type: **nonNegativeInteger**, Use: **optional**

Specifies the default time-out value (in ms) for all `stream_types` not specified in the list of **StreamTypeTimeout** elements.

If a referenced PID is not encountered within the time specified here a `PID_error` violation will be reported.

StreamTypeTimeout

This element can be used to specify a timeout for the `PID_error` indicator on the basis of a `PIDs stream_type`.

stream_type

Type: **nonNegativeInteger**, Use: **required**

Specifies the `stream_type` for which the timeout is specified.

time_out_ms

Type: **nonNegativeInteger**, Use: **required**

Specifies the time-out value (in ms) for a specific `stream_type`.

If a referenced PID, with the specified `stream_type`, is not encountered within the time specified here a `PID_error` violation will be reported.

4.2.2. Priority2

Priority2

Type: **TBasePriority**

The **Priority2** element defines the TR 101 290 priority level 2 indicators/rules. It is derived from the **TBasePriority** (see §5.1) type. It inherits all attributes from this type and extends the type by adding the following sequence of optional child elements: **Transport_error**, **CRC_error**, **PCR_repetition_error**, **PCR_discontinuity_indicator_error**, **PCR_accuracy_error**, **PTS_error** and **CAT_error**.

Transport_error

Type: **TBaseIndicator**

This element controls the `Transport_error` indicator. It is a type definition of the **TBaseIndicator** type (see §5.1) and inherits all attributes and child elements from this type.

CRC_error

Type: **TBaseIndicator**

This element controls the `CRC_error` indicator. It is a type definition of the **TBaseIndicator** type (see §5.1) and inherits all attributes and child elements from this

type. It further extends the **TBaseIndicator** type by adding an optional list of **SectionIdentifier** child elements.

SectionIdentifier

Type: **TSectionIdentifier**

This element identifies a section/section range for which the CRC should be checked.

The **SectionIdentifier** is a type definition of the **TSectionIdentifier** type. Refer to §5.5.2 for a detailed description of the **TSectionIdentifier** type.

NOTE: the **SectionNum** and **SectionNumRange** child elements are ignored.

PCR_repetition_error

Type: **TBaseIndicator**

This element controls the **PCR_repetition_error** indicator. It is a type definition of the **TBaseIndicator** type (see §5.1) and inherits all attributes and child elements from this type. It extends the type adding following attribute: **max_interval_ms**.

max_interval_ms

Type: **nonNegativeInteger**, Use: **optional**

Specifies the maximum allowed interval between two consecutive PCRs, on the same PID, in ms.

If the interval between two PCRs exceeds the period specified here a **PCR_repetition_error** violation will be reported.

PCR_discontinuity_indicator_error

Type: **TBaseIndicator**

This element controls the **PCR_discontinuity_indicator_error** indicator. It is a type definition of the **TBaseIndicator** type (see §5.1) and inherits all attributes and child elements from this type. It extends the type adding following attribute: **max_difference_ms**.

max_difference_ms

Type: **nonNegativeInteger**, Use: **optional**

Specifies the maximum allowed difference between two consecutive PCR values, on the same PID, in ms.

If the difference between two PCR values exceeds the value specified here a **PCR_discontinuity_indicator_error** violation will be reported.

PCR_accuracy_error

Type: **TBaseIndicator**

This element controls the **PCR_accuracy_error** indicator. It is a type definition of the **TBaseIndicator** type (see §5.1) and inherits all attributes and child elements from this type. It extends the type adding following attribute: **max_error_ns**.

max_error_ns

Type: **nonNegativeInteger**, Use: **optional**

Specifies the maximum allowed PCR accuracy error in ns.

If the PCR accuracy exceeds the value specified here a **PCR_accuracy_error** violation will be reported.

CAT_error

Type: **TSectionIndicator**

This element controls the **CAT_error** indicator. It is a type definition of the **TSectionIndicator** type (see §5.3) and inherits all its attributes and child elements from this type.

4.2.3. Priority3

Priority3

Type: **TBasePriority**

The **Priority3** element defines the 101 290 priority level 3 indicators/rules. It is derived from the **TBasePriority** (see §5.1) type. It inherits all attributes from this type and extends the type by adding the following sequence of optional child elements:

NIT_actual_error, **NIT_other_error**,
SI_repetition_error, **Buffer_error**,

Unreferenced_PID, **SDT_actual_error**,
SDT_other_error, **EIT_actual_error**,
EIT_other_error, **EIT_PF_error**,
RST_error, **TDT_error**,
Empty_buffer_error and
Data_delay_error.

NIT_actual_error

Type: **TSectionIndicator**

This element controls the `NIT_actual_error` indicator. It is a type definition of the **TSectionIndicator** type (see §5.4) and inherits all its attributes and child elements from this type.

NIT_other_error

Type: **TSectionIndicator**

This element controls the `NIT_other_error` indicator. It is a type definition of the **TSectionIndicator** type (see §5.3) and inherits all its attributes and child elements from this type.

SI_repetition_error

Type: **TTableIndicator**

This element controls the `SI_repetition_error` indicator. It is a type definition of the **TTableIndicator** type (see §5.3) and inherits all its attributes and child elements from this type.

Buffer_error

Type: **TBaseIndicator**

This element controls the `Buffer_error` indicator. It is a type definition of the **TBaseIndicator** type (see §5.1) and inherits all attributes and child elements from this type.

Unreferenced_PID

Type: **TBaseIndicator**

This element controls the `Unreferenced_PID` indicator. It is a type definition of the **TBaseIndicator** type (see §5.1) and inherits all attributes and child elements from this type. It extends this type by adding the `time_out_ms` attribute and an

optional list of **ExcludedPid** and/or **ExcludedPidRange** child elements.

time_out_ms

Type: **nonNegativeInteger**, Use: **optional**

This attribute defines the time-out (in ms) period for an unreferenced PID.

PIDs which are defined in the list of **ExcludedPid/ ExcludedPidRange** list are excluded from the time-out check.

If a PID is not referenced from the (P)SI information within the time specified here a `Unreferenced_PID` violation will be reported.

ExcludedPid

Type: **TSimpleSelection**

This element defines a single PID which should be excluded from the time-out check. This element is type definition of the **TSimpleSelection** type (see §5.5.5).

ExcludedPidRange

Type: **TRangeSelection**

This element defines a range of PIDs which should be excluded from the time-out check. This element is type definition of the **TRangeSelection** type (see §5.5.6).

SDT_actual_error

Type: **TSectionIndicator**

This element controls the `SDT_actual_error` indicator. It is a type definition of the **TSectionIndicator** type (see §5.3) and inherits all its attributes and child elements from this type.

SDT_other_error

Type: **TSectionIndicator**

This element controls the `SDT_other_error` indicator. It is a type definition of the **TSectionIndicator** type (see §5.3) and inherits all its attributes and child elements from this type.

EIT_actual_error

Type: **TSectionIndicator**

This element controls the `EIT_actual_error` indicator. It is a type definition of the **TSectionIndicator** type (see §5.3) and inherits all its attributes and child elements from this type.

EIT_other_error

Type: **TSectionIndicator**

This element controls the `EIT_other_error` indicator. It is a type definition of the **TSectionIndicator** type (see §5.3) and inherits all its attributes and child elements from this type.

EIT_PF_error

Type: **TBaseIndicator**

This element controls the `EIT_PF_error` indicator. It is a type definition of the **TBaseIndicator** type (see §5.1) and inherits all attributes and child elements from this type.

RST_error

Type: **TSectionIndicator**

This element controls the `RST_error` indicator. It is a type definition of the **TSectionIndicator** type (see §5.3) and inherits all its attributes and child elements from this type.

TDT_error

Type: **TSectionIndicator**

This element controls the `TDT_error` indicator. It is a type definition of the **TSectionIndicator** type (see §5.3) and inherits all its attributes and child elements from this type.

Empty_buffer_error

Type: **TBaseIndicator**

This element controls the `Empty_buffer_error` indicator. It is a type definition of the **TBaseIndicator** type (see §5.1) and inherits all attributes and child elements from this type.

Data_delay_error

Type: **TBaseIndicator**

This element controls the `Data_delay_error` indicator. It is a type definition of the **TBaseIndicator** type (see §5.1) and inherits all attributes and child elements from this type.

4.3. Global Log Settings

GlobalLogSettings

This element defines the global log file settings. Using this element one can configure the number of log files, maximum log file size and the log duration per file settings.

only_log_time_and_indicator

Type: **boolean**, Use: **optional**

By setting this attribute to 'true' one indicates that the log file should only contain the time/date and indicator that was violated. Setting this attribute to 'false' indicates that the time/date, violated indicator and an error message should be logged.

NOTE: if this attribute is set to 'true' it can be overruled by the **only_log_time_and_indicator** attribute of each indicators **LogSettings** child element (see §5.2 and §5.5.1).

max_file_size_MB

Type: **nonNegativeInteger**, Use: **optional**, Valid Values: **1-1024**

Specified the maximum allowed size (in MB) for a log file. If the maximum size is reached, logging to the file will be suspended (i.e. no log entries will be added to the file).

NOTE: if multiple log files are used, logging will resume as soon as the log file is switched (as defined by the **switch_file_every_X_hours** attribute).

max_num_files

Type: **nonNegativeInteger**, Use: **optional**, Valid Values: **1-16**

Specifies the maximum number of log files a maintained by the StreamXpert (see **switch_file_every_X_hours** attribute).

switch_file_every_X_hours

Type: **nonNegativeInteger**, Use: **optional**,
Valid Values: **0, 1, 2, 4, 8 or 24**

The StreamXpert has the capability to maintain a list of X (=max_num_files) log files as a kind of trace buffer. The **switch_file_every_X_hours** attribute defines the maximum hours of log stored in each log file maintained.

The StreamXpert will switch to the next log file every Xth hour. Upon the file switch the StreamXpert will first check how many files it has in its 'trace buffer'. If the number of files is less than the maximum a new log file is created add it is added to the list of files. If during the current session the number of log files already has reached the maximum allowed the StreamXpert will replace the oldest existing log file with a new one.

Set the **switch_file_every_X_hours** attribute to zero to switch to a new file if the maximum log size is reached instead one the bases of a log duration.

NOTE: If the maximum number of files is set to 1 the StreamXpert will never switch to a new file and will continue logging until the maximum file size is reached.

NOTE: the StreamXpert automatically names the files it maintains. The log file name has the following format: **prefix_date_#**. The **prefix** is user defined. The date is the current date with the following format YYYY_MM_DD (e.g. 2005_07_05). The **#** is the file number for the day. An example file name is:
Tr101290Log_2005_07_05_2.txt.

override_behavior

Type: **tOverrideBehavior**, Use: **optional**

This attribute controls the behavior when overriding the global log settings indicator (i.e. when deriving a profile from another profile).

Setting this attribute to **extend_existing** indicates that only attributes mentioned in the XML description for the global log settings

will be replaced, all unmentioned attributes will retain their original values (i.e. the value assigned in the Base Profile).

Setting this attribute to **replace_existing** indicates that the all global log settings should be replaced by the ones defined here. All attributes are required now since we want to completely replace the existing **GlobalLogSettings**.

See §5.6.1 for a description of the **tOverrideBehavior** data type.

5. Base XML Types

This section describes the user defined base types used in the XSD schema definition for the TR 101 290 Profiles

5.1. Base Priority Type

TBasePriority

This is the base element for all priority elements (e.g. **Priority1**, **Priority2**, **Priority3**). All Priority elements are derived from this **complexType**.

enabled

Type: **boolean**, Use: **required**

Global enable flag for all indicators/rules defined within the priority. Set this flag to "false" to disable all indicators. If the flag is set to "true" the individual enable flag of each indicator can overrule this value.

5.2. Base Indicator Type

TBaseIndicator

This is the base type for all indicators defined in the schema. The **TBaseIndicator** contains the following sequence of optional child elements: **LogSettings** and **Filter**.

enabled

Type: **boolean**, Use: **optional**

Enable/disable flag for the indicator. Set to "false" to disable the indicator. Setting this attribute to "true" will enable the indicator and the StreamXpert will check for violations of this indicator/rule.

NOTE: the value of this attribute overrules the **TBasePriority** enable attribute only if it was set to "true".

override_behavior

Type: **tOverrideBehavior**, Use: **optional**

This attribute controls the behavior when overriding an indicator (i.e. when deriving a profile from another profile). Setting this attribute to **extend_existing** indicates that only attributes mentioned in the XML description for the indicator will be replaced, all un-

mentioned attributes will retain their original values (i.e. the value assigned in the Base Profile).

Setting this attribute to **replace_existing** indicates that the indicator replaces the existing indicator values as defined in the base profile. All attributes are required now since we want to completely replace the existing indicator.

NOTE: the value of this attribute acts as the default value for the child elements (e.g. **LogSettings** or **Filter**). The child elements can overrule this by specifying their local **override_behavior** value.

See §5.6.1 for a description of the **tOverrideBehavior** data type.

LogSettings

Type: **TIndicatorLogSettings**

The **LogSettings** element defines the log settings for the indicator. This element is a type definition of the **TIndicatorLogSettings** type (see §5.5.1).

Filter

Type: **TPidFilter**

The **Filter** element can be used to apply an include- or exclude-filter to the indicator (i.e. only check the indicator/rule for a specific set of PIDs). If the Filter is empty the indicator will be verified for all PIDs.

This element is type definition of the **TPidFilter** type. Refer to §5.5.4 for a description of **TPidFilter** type.

5.3. Section Indicator Type

TSectionIndicator

Type: **TBaseIndicator**

This is the base type for all section related indicators. It is derived from the **TBaseIndicator** type (see §5.2) and inherits all attributes and child elements from this type. It further extends the type by adding the following sequence of optional child el-

ements: **SectionRepetitionRule** and **SectionLocationRule**.

5.3.1. Section Repetition Rule

SectionRepetitionRule

This element defines a repetition requirement for a specific section or group of sections. The **SectionRepetitionRule** contains the following sequence of child elements: **Filter** (=optional) and **SectionIdentifier**.

max_interval_ms

Type: **nonNegativeInteger**, Use: **required**
 Defines the maximum allowed interval (in ms) between two consecutive sections.

min_interval_ms

Type: **nonNegativeInteger**, Use: **optional**
 Defines the minimum allowed interval (in ms) between two consecutive sections.

required

Type: **boolean**, Use: **optional**
 Defines whether or not the section is required to be present. If the section is required and no section is detected within the time specified by **max_interval_ms** an error will be reported.

override_behavior

Type: **tOverrideBehavior**, Use: **optional**
 This attribute controls the behavior when overriding a section repetition rule (i.e. when deriving a profile from another profile).

If the value is set to **extend_existing** only the attributes defined in the derived profile will replace will be replaced. All undefined attributes will retain their original value (i.e. value as defined in the base profile).

Setting this attribute to **replace_existing** indicates that all existing attributes values (i.e. values as defined in the base profile) will be replaced by the values in the derived profile.

NOTE: the **max_interval_ms**, **min_interval_ms** and **required** attributes must be defined if the **override_behavior** is set the **replace_existing**.

NOTE: if the **override_behavior** attribute is not defined the override behavior of the parent element is used.

See §5.6.1 for a description of the **tOverrideBehavior** data type.

Figure 1 Example XML for Section Repetition Rule

```
// Repetition rule for the NIT-Actual.
// Sections with table_id 64 (NIT-Actual)
// should occur once every 10s.
// Only check repetition rule if the section
// is found on PID 16
<SectionRepetitionRule
    max_interval_ms = "10000"
    required = "true">
    <Filter>
        <Pid value = "16"/>
    </Filter>
    <SectionIdentifier>
        <TableId value = "64"/>
    </SectionIdentifier>
</SectionRepetitionRule>
```

Filter

Type: **TPidFilter**
 The optional **Filter** element can be used to limit the section repetition rule to only check for sections on the set of PIDs specified by this filter.

This element is type definition of the **TPidFilter** type. Refer to §5.5.4 for a description of **TPidFilter** type.

SectionIdentifier

Type: **TSectionIdentifier**
 This element identifies the section / section range the rule applies to. The **SectionIdentifier** is a type definition of the **TSectionIdentifier** type. Refer to §5.5.2 for a detailed description of the **TSectionIdentifier** type.

5.3.2. Section Location Rule

SectionLocationRule

This element defines a location rule for a specific section or group of sections. A location rule is a rule that specifies which section(s) are allowed to occur on a specific

PID. The **SectionLocationRule** contains a list **SectionIdentifier** child elements.

NOTE: The **SectionNum** or **SectionNumRange** child elements of a **SectionIdentifier** element are not used.

pid

Type: **nonNegativeInteger**, Use: **required**
 Defines the PID the section(s) should be located on.

override_behavior

Type: **tOverrideBehavior**, Use: **optional**
 This attribute controls the behavior when overriding a section location rule (i.e. when deriving a profile from another profile).

If the value is set to **extend_existing** only the attributes defined in the derived profile will be replaced. All undefined attributes will retain their original value (i.e. value as defined in the base profile).

Setting this attribute to **replace_existing** indicates that all existing attributes values (i.e. values as defined in the base profile) will be replaced by the values in the derived profile.

NOTE: if the **override_behavior** attribute is not defined the override behavior of the parent element is used.

See §5.6.1 for a description of the **tOverrideBehavior** data type.

Figure 2 Example XML for Section Location Rule

```

// Location rule for the NIT. Sections with
// table_id 64 (NIT-Actual), 65 (NIT-Other)
// and 114 (Stuffing Table) are allowed on
// PID 16
<SectionLocationRule pid = "16">
  <SectionIdentifier>
    <TableIdRange min_val = "64"
                  max_val = "65"/>
  </SectionIdentifier>
  <SectionIdentifier>
    <TableId value = "114"/>
  </SectionIdentifier>
</SectionLocationRule>
    
```

SectionIdentifier

Type: **TSectionIdentifier**

This element identifies the section / section range the rule applies to. The **SectionIdentifier** is a type definition of the **TSectionIdentifier** type. Refer to §5.5.2 for a detailed description of the **TSectionIdentifier** type.

5.4. Table Indicator Type

TTableIndicator

Type: **TBaseIndicator**
 This is the base type for all table related indicators. It is derived from the **TBaseIndicator** type (see §5.2) and inherits all attributes and child elements from this type. It further extends the type by adding the optional list **TableRepetitionRule** child elements.

TableRepetitionRule

This element defines a repetition requirement for a specific table or group of tables. The **TableRepetitionRule** contains the following sequence of child elements: **Filter** (=optional) and **TableIdentifier**.

max_interval_ms

Type: **nonNegativeInteger**, Use: **required**
 Defines the maximum allowed interval (in ms) between two consecutive tables.

min_interval_ms

Type: **nonNegativeInteger**, Use: **optional**
 Defines the minimum allowed interval (in ms) between two consecutive tables.

required

Type: **boolean**, Use: **optional**
 Defines whether or not the table is required to be present. If the table is required and no table is detected within the time specified by **max_interval_ms** an error will be reported.

name

Type: **string**, Use: **required**
 Unique name for the repetition rule.

To override a specific repetition rule this name should be set to the name of the rule to be overridden.

NOTE: The override behaviour is specified by the **TTableIndicator's** `override_behavior` attribute.

Filter

Type: **TPidFilter**

The optional **Filter** element can be used to limit the table repetition rule to only check for tables on the set of PIDs specified by this filter.

This element is type definition of the **TPidFilter** type. Refer to §5.5.4 for a description of **TPidFilter** type.

TableIdentifier

Type: **TTableIdentifier**

This element identifies the table/ table range the rule applies to. The **TableIdentifier** is a type definition of the **TTableIdentifier** type. Refer to §5.5.3 for a detailed description of the **TTableIdentifier** type.

5.5. Miscellaneous Types

5.5.1. Indicator Log Settings Type

TIndicatorLogSettings

With indicator log settings one can specify the log settings for a specific indicator/rule.

max_log_per_10_sec

Type: **nonNegativeInteger**, Use: **optional**

Specifies the maximum allowed number of log message to be written to the log file per 10 seconds. This attribute can be used to prevent the log file to be clogged with one specific error which occurs regularly.

max_log_per_60_sec

Type: **nonNegativeInteger**, Use: **optional**

Specifies the maximum allowed number of log message to be written to the log file per 60 seconds. Like the **max_log_per_10_sec** attribute this attribute is intended to limit the number of log messages writing to the log file.

This attribute overrules the **max_log_per_10_sec** attribute. This means that if the number of errors logged in the last 10seconds does not exceed the value specified by the **max_log_per_10_sec** attribute, but the number of errors logged in the last 60 seconds is exceeded the error will not be logged.

only_log_time_and_indicator

Type: **boolean**, Use: **optional**

By setting this attribute to 'true' one indicates that the log file should only contain the time/date and indicator that was violated. Setting this attribute to 'false' indicates that the time/date, violated indicator and an error message should be logged.

This attribute will overrule the **only_log_time_and_indicator** attribute value of the Global Log Settings (see §4.3) only if the global attribute is set to 'true'.

override_behavior

Type: **iOverrideBehavior**, Use: **optional**

This attribute controls the behavior when overriding the indicator log settings (i.e. when deriving a profile from another profile).

If the value is set to **extend_existing** only the attributes defined in the derived profile will be replaced. All undefined attributes will retain their original value (i.e. value as defined in the base profile).

Setting this attribute to **replace_existing** indicates that all existing attributes values (i.e. values as defined in the base profile) will be replaced by the values in the derived profile.

NOTE: the **max_log_per_10_sec**, **max_log_per_60_sec** and **only_log_time_and_indicator** attributes must be defined if the **override_behavior** is set the **replace_existing**.

NOTE: if the **override_behavior** attribute is not defined the override behavior of the parent element is used.

See §5.6.1 for a description of the **tOverrideBehavior** data type.

Figure 3 Example XML for Indicator Log Settings

```
// Log no more than 10 messages per 10 sec
// Log no more than 30 message per 60 sec
// Include error message in log
<LogSettings
  max_log_per_10_sec = "10"
  max_log_per_60_sec = "30"
  only_log_time_and_indicator = "false"/>
```

5.5.2. Section Identifier Type

TSectionIdentifier

A section identifier is used to identify a specific section or range of sections. It contains a choice of a **TableId** or **TableIdRange** child elements followed by the optional choice of a **SectionNum** or **SectionNumRange** child element.

Figure 4 Example XML for Section Identifier

```
// Identifies PMT section (table_id = 0x02)
<SectionIdentifier>
  <TableId value = "2"/>
</SectionIdentifier>

// Identifies the PMT section 1 (table_id =
0x02, section_number = 1)
<SectionIdentifier>
  <TableId value = "2"/>
  <SectionNum value = "1"/>
</SectionIdentifier>
```

TableId

Type: **TSimpleSelection**

The **TableId** element defines a single `table_id` to identify the section selected by the **TSectionIdentifier** element. This element is type definition of the **TSimpleSelection** type (see §5.5.5).

TableIdRange

Type: **TRangeSelection**

The **TableIdRange** element defines a range of `table_ids` to identify the tables selected by the **TSectionIdentifier** element. This element is type definition of the **TRangeSelection** type (see §5.5.6).

SectionNum

Type: **TSimpleSelection**

The **SectionNum** element can be used to limit the sections identified by the **TSectionIdentifier** element to section with one specific section number. This element is type definition of the **TSimpleSelection** type (see §5.5.5).

SectionNumRange

Type: **TRangeSelection**

The **SectionNumRange** element can be used to limit the sections identified by the **TSectionIdentifier** element to sections with a section number within a specified range. This element is type definition of the **TRangeSelection** type (see §5.5.6).

5.5.3. Table Identifier Type

TTableIdentifier

A table identifier is used to specify a specific table or range of tables. It contains a choice of a **TableId** or **TableIdRange** child element followed by the optional choice of a **TableType** or **TableTypeRange** child element.

Figure 5 Example XML for Table Identifier

```
// Identifies the PAT (table_id = 0x00)
<TableIdentifier>
  <TableId value = "0"/>
</TableIdentifier>

// Identifies the PSIP EIT-0 (table_id =
0xCB, table_type = 0x0100)
<TableIdentifier>
  <TableId value = "203"/>
  <TableType value = "256"/>
</TableIdentifier>
```

TableId

Type: **TSimpleSelection**

The **TableId** element defines a single `table_id` to identify the table selected by the **TTableIdentifier** element. This element is type definition of the **TSimpleSelection** type (see §5.5.5).

TableIdRange

Type: **TRangeSelection**

The **TableIdRange** element defines a range of `table_ids` to identify the tables selected by the **TTableIdentifier** element. This element is type definition of the **TRangeSelection** type (see §5.5.6).

TableType

Type: **TSimpleSelection**

The **TableType** element can be used to limit the tables identified by the **TSectionIdentifier** element to a specific ATSC table type (i.e. the type defined in the MGT table). This element is type definition of the **TSimpleSelection** type (see §5.5.5).

TableTypeRange

Type: **TRangeSelection**

The **TableTypeRange** element can be used to limit the tables identified by the **TSectionIdentifier** element to a specific range of ATSC table types. This element is type definition of the **TRangeSelection** type (see §5.5.6).

5.5.4. PID Filter Type

TPidFilter

A PID filter is a list of PIDs that should be included or excluded from a specific test. The PIDs in the filter are defined by the **Pid** and/or **PidRange** child elements.

type

Type: **string**, Use: **optional**

Setting this attribute to the `include_filter` value indicates that all PIDs in the filter list are included (enabled). Setting this attribute to the `exclude_filter` value indicates that all PIDs in the filter list are excluded (i.e. disabled).

override_behavior

Type: **tOverrideBehavior**, Use: **optional**

This attribute controls the behavior when overriding a PID filter (i.e. when deriving a profile from another profile).

When the value is set to `extend_existing` the PIDs defined in the **Pid** and **PidRange** child elements will be added to the existing PIDs (i.e. the PIDs defined in the base pro-

file) and/or the `type` attribute can be replaced by the new value.

Setting this attribute to `replace_existing` indicates that all existing PIDs in the filter should be removed and replaced by the PIDs defined in the **Pid** and **PidRange** child elements.

NOTE: the `type` attribute is required when setting the `override_behavior` is set to `replace_existing`.

NOTE: the **Pid** and **PidRange** child elements are always optional, regardless of the value of the `override_behavior` attribute. No **Pid** or **PidRange** elements indicate that there are no PIDs in the filter.

NOTE: if the `override_behavior` attribute is not defined the override behavior of the parent element is used.

See §5.6.1 for a description of the **tOverrideBehavior** data type.

Figure 6 Example XML for a Pid Filter

```
// Filter which includes PID 0, 1 and 3
<Filter type>
  <Pid value = "0"/>
  <Pid value = "1"/>
  <Pid value = "2"/>
</Filter>

// Filter which excludes the NULL-packet PID
<Filter type = "exclude_filter">
  <Pid value = "8191"/>
</Filter>
```

Pid

Type: **TSimpleSelection**

The **Pid** element defines a single PID that should be added to the PID filter. This element is type definition of the **TSimpleSelection** type (see §5.5.5).

PidRange

Type: **TRangeSelection**

The **PidRange** element defines a range of PIDs that should be added to the PID filter. This element is type definition of the **TRangeSelection** type (see §5.5.6).

5.5.5. Simple Selection Type

TSimpleSelection

This element defines a simple integer selection.

value

Type: **integer**, Use: **required**
Selected integer value.

5.5.6. Range Selection Type

TRangeSelection

This element defines an integer selection range.

min_val

Type: **integer**, Use: **required**
Minimum integer value included in the selection range.

max_val

Type: **integer**, Use: **required**
Maximum integer value included in the selection range.

5.6. Data Types

This section describes all user defined XML data types used in the XSD schema.

5.6.1. Override Behavior Data Type

The **tOverrideBehavior** type is used to define the behavior when deriving a profile from another profile. This type is an enumeration with the following possible values:

extend_existing: indicates that the existing settings (as defined in the base profile) should remain intact. They are only replaced if they are redefined in the new profile. This behavior is ideal for modifying a specific part of existing indicator (e.g. changing the maximum allowed interval for a specific section) or adding an additional PID to an indicators **Filter** element.

replace_existing: indicates that the existing settings (as defined in the base profile) are completely replaced with the settings for the indicator in the new profile. In most cases one is required to define all attributes, since we are completely existing indicator settings. This be-

havior is ideal to completely overrule an indicator in the base profile.