XML SI configuration manual

for
"Scienta MPTS Muxer" Win application
and
"OCTOPUS Multiplexer"

2015

ScientaMedia multiplexers are capable of Service Information (SI) tables configuration via XML files. The XML structure enables writing of any descriptor into any loop / entry / table supported by the multiplexer.

The tables and entries MUST already exist and have appropriate IDs being configured via the WEB control or WIN app GUI, otherwise descriptors will not be written, because the multiplexer won't be able to locate the target tables.

Configuring Scienta MPTS Muxer Windows Application

The Windows app is configured via the si_config.xml file which is located in the application's working directory.

Configuring Octopus Multiplexer via WEB UI

To apply the configuration for Octopus muxer via WEB UI, edit the XML file and upload it to the multiplexer via the WEB control page. The XML file can later on be downloaded from the multiplexer. The “Manage SI XML” section is located on the main page of the multiplexer (see the screen shot below). In case if file with existing name is uploaded, the existing file is overwritten.

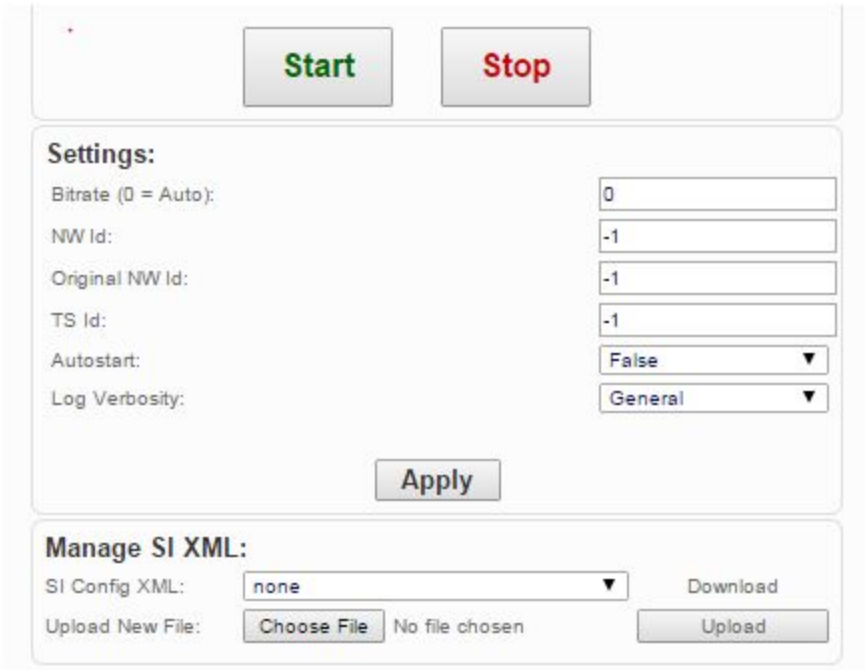

Fig. 1 – SI management section - screenshot

Important! : To address correctly to the NIT / SDT tables and their entries, make sure the TS Id, NW Id and Original NW Id parameters are set correctly.

SI XML configuration file structure.

The <table> element.
The <table> element requires the attribute "id" which specifies the DVB table ID according to EN 300 468 "table_id" and the attribute "pid" which serves to address PMTs carried by different PIDs.

Descriptors can be specified for either outer or inner loops of a table.

To specify outer loop descriptors, use the **<outer_loop>** element carrying one or more child <descriptor> elements.
The <descriptor> element is equal for both outer and inner loops.
The <descriptor> element syntax will be described further in the document.

To specify inner loop descriptors, use the **<inner_loop>** element. The <inner_loop> element MUST have <entry> element as a child item to address particular entry carried by the table.
The <entry> element has two attributes: "id" (required) and "id2" (optional). The "id" attribute corresponds to the main entry identifier (e.g. PID for PMT, service_id for SDT).
The "id2" attribute is required if table entry is identified by two identifiers (e.g. NIT entries: the "id" attribute corresponds to "transport_stream_id" and "id2" corresponds to "original_network_id").

The <descriptor> element.
The <descriptor> element is a complex element specifying a descriptor according to it's syntax.
The XML structure of the element enables writing any descriptor of any syntax (not only limited to DVB or ISO 13818-1 descriptors).
The only restriction which shall be maintained is the total length of data fields (<=256).
Essentially, the "descriptor" is a map of bit or byte fields: the lenghts and values.
Each descriptor may have the attribute "name" (optional). It's purpose is to make the XML easily readable. The value of this attribute is not used by the multiplexer.
The required element <tag> specifies the attribute tag value (integer).

The actual data fields of a descriptor are specified by a sequence of <field> elements. The <field> element may have an optional attribute "name" (used to make the xml readable, not used by the multiplexer).
The <field> element MUST have one of the three following child elements:
- <string>;
- <bits>;
- <bytes>.

The <string> simple element specifies a sequence of characters (a string).
The multiplexer writes the <string> value into the descriptor as it is, the length of the field is equal to the length of the string.

If field's length is specified by either <bits> or <bytes>, then <field> element MUST have child element <value>.
The <bits> and <bytes> elements specify that the field's length in bits or bytes, respectively.
The simple element <value> defines the actual value of the field, except the keyword *NEXT* which is used to simplify specification of "length" - "value" pairs in the descriptors
(e.g. "service_name_length" and consecutive "service_name" fields).

If the *NEXT* is specified, then the multiplexer uses the length of the next field as a value of the *current* field. It simplifies the process as it's not necessary to know the actual length of the related field length and minimizes mistake probability.

The example of the service descriptor syntax is given below.

```
<descriptor name="Service Descriptor">
	<tag>72</tag>
	<field name="service_type">
		<bits>8</bits>
		<value>22</value>
	</field>
	<field name="length">
		<bits>8</bits>
		<value>*NEXT*</value>
	</field>
	<field name="provider_name">
		<string>TV PROVIDER</string>
	</field>
	<field name="length">
		<bytes>1</bytes>
		<value>*NEXT*</value>
	</field>
	<field name="program_name">
		<string>Canale Uno</string>
	</field>
</descriptor>
```

Important! : It's customer's responsibility to specify correct descriptor syntax and the right tags for descriptors - the multiplexer does not validate the semantics!