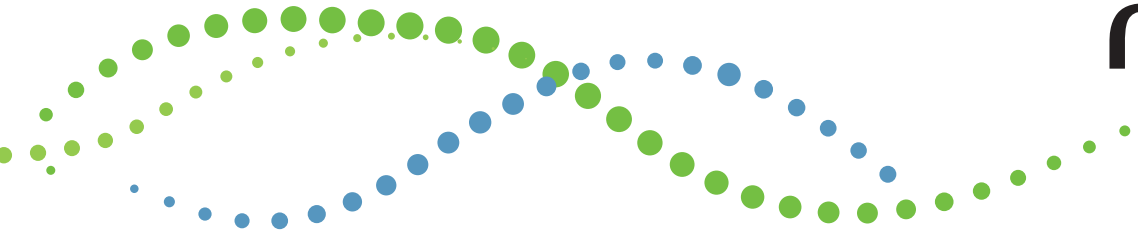

TXP Configuration

Revision: DRAFT A (4592)

2015-01-21

1 Introduction

This document describes how to interpret the configuration returned using the TXP protocol. The TXP protocol itself is described in another document. Note that this document is only valid for some Nevion products.

2 Core concepts

2.1 Schema

The schema is an XML document which contains meta-data for (almost) all parameters visible in the GUI. This might be labels, units, min/max values and more. The schema can be exported in two different modes, both XML.

```
http://<ip>/txp_get_schema
http://<ip>/txp_get_schema?format=xsd
```

The schema is built up the same way as the actual XML tree describing the data, so it's straightforward to find the schema for any given data node.

2.2 Global identifiers

Many nodes in the tree will have a field called **gid**, which is used as a Global Identifier. This simply means this identifier is unique throughout the unit regardless of the type of the node. This number if always allocated within a range for each type. The range of values valid for each type could be extracted using the txp_schema command.

```
http://<ip>/txp_get_schema?format=xsd
```

See the GID parameter for each of the relevant nodes to see the range.

2.2.1 Ranges

All elements will be allocated a GID withing the ranges defined.

Table 2.1 GID ranges

Type	Min	Max
Ethernet	10000	19999
IP interface	20000	29999
VLAN	30000	39999
ASI input	40000	49999
DVB-S/S2 input	60000	69999
TS channel	100000	299999
TSoIP input	300000	499999

2.3 Elements

The main of the configuration tree is found under /data/elements. To see all nodes under see tree invoke the command.

```
http://<ip>/txp_get_tree?path=/data/elements&depth=2
```

Make note the usage of the depth parameter. This makes the returned request much smaller, and will also be much faster to execute.

The **elements** node contain several collections which in turn contain logical elements like ports and TS channels. One important concept is that an ASI input is splitted into an ASI element and a TS element. The same goes for other input like TSoIP and Satellite.

2.3.1 Routing node

Each of the elements within the elements contains a routing node. This node is very important because it defines how the various elements are connected. Inside the routing node the **src_gid** and **dst_gid** are the most important parameters. This gives the Global Identifier of the source/destination of the element.

3 Examples

3.1 Find the TS data from an ASI Input

Start by fetching the list of ASI inputs:

```
http://<ip>/txp_get_tree?path=/data/elements/asiinput_coll&depth=2

<response request_id="1016705170">
  <status status="0" status_text="OK"/>
  <data>
    <elements>
      <asiinput_coll>
        <asiinput _id="40000" enable="false" label="" gid="40000" slot="0" flabel="ASI M.1"/>
        <asiinput _id="40001" enable="false" label="" gid="40001" slot="0" flabel="ASI M.2"/>
        <asiinput _id="40002" enable="false" label="" gid="40002" slot="0" flabel="ASI M.3"/>
        <asiinput _id="40003" enable="false" label="" gid="40003" slot="0" flabel="ASI M.4"/>
      </asiinput_coll>
    </elements>
  </data>
</response>
```

We can see from this list that there are 4 ASI inputs in the system. We would like to fetch TS data from the port labelel ASI M.1 Fetch detailed information in the routing node from ASI M.1:

```
http://<ip>/txp_get_tree?path=/data/elements/asiinput_coll/[40000]/routing

<response request_id="1188548601">
  <status status="0" status_text="OK"/>
  <data>
    <elements>
      <asiinput_coll>
        <asiinput _id="40000">
          <routing src_gid="1" src_formats="asi" src_gid_flabel="Port M.1" src_present="true"
dst_formats="tssrc" dst_gid="290000" shelf_gid="1" active="true"/>
        </asiinput>
      </asiinput_coll>
    </elements>
  </data>
</response>
```

The **dst_gid** of this ASI input is 290000. This is within the TS range so we request the TS node with key 290000:

```
http://<ip>/txp_get_tree?path=/data/elements/ts_coll/[290000]

<response request_id="3348952854">
  <status status="0" status_text="OK"/>
  <data>
    <elements>
      <ts_coll>
        <ts _id="290000" enable="false" tsmode="2" src_gid="40000" gid="290000" flabel="">
          <op spts_enable="0" adv_mon_enable="false" fun_mask="0"/>
          <routing src_gid="40000" src_formats="tssrc" src_gid_flabel="ASI M.1" src_present="true"
dst_formats="ts" dst_gid="" shelf_gid="0" active="true"/>
```

```
          <stream totrate="9765888" effrate="4755648" tsname="Cross-country Channel" onid="0"
tsid="8206"/>
          <pid_signaling/>
          <psisi/>
          <pid_coll></pid_coll>
          <service_coll></service_coll>
          <pcr_pid_coll></pcr_pid_coll>
        </ts>
      </ts_coll>
    </elements>
  </data>
</response>
```

This node contains lots of sub nodes containing more TS specific information.

3.2 Find services within a Transport Stream

Given that you already know the GID of the transport stream, it is simple to list the services present. For example if the TS channel GID is 290000:

```
http://<ip>/txp_get_tree?path=/data/elements/ts_coll/[290000]/service_coll&depth=2

<response request_id="2583407843">
  <status status="0" status_text="OK"/>
  <data>
    <elements>
      <ts_coll>
        <ts _id="290000">
          <service_coll>
            <service _id="28400" s="28400" m="100" c="101" r="1352096" rx="1361120" rn="843744"
z="0" dt="0"/>
            <service _id="28401" s="28401" m="110" c="111" r="1352096" rx="1355104" rn="840736"
z="0" dt="0"/>
            <service _id="28402" s="28402" m="120" c="121" r="1352096" rx="1356608" rn="839232"
z="0" dt="0"/>
          </service_coll>
        </ts>
      </ts_coll>
    </elements>
  </data>
</response>
```

The returned data is a list of services, where the Service Id is the key of the collection. Getting more info for a specific service is done by requesting an entry with a specific Service Id:

```
http://<ip>/txp_get_tree?path=/data/elements/ts_coll/[290000]/service_coll/[28401]

<response request_id="2583407843">
  <status status="0" status_text="OK"/>
  <data>
    <elements>
      <ts_coll>
        <ts _id="290000">
          <service_coll>
            <service _id="28401" s="28401" m="110" c="111" r="1353600" rx="1929632" rn="836224"
z="0" dt="0">
```

```
            <sdt x="true" n="Bayern 2" t="Digital radio" v="2" p="ARD BR" f="true" d="true" c="false"
r="Running"/>
            <vct x="false" n="" mj="0" mn="0" m="" f="0" ts="0" pn="28401" etm="" c="false" h="false"
hg="false" t="" v="0" src="0"/>
            <component_coll>
              <component _id="111" p="111" t="Audio" v="3" ct="2" e="true" r="335392" rx="909920"
rn="0" x="1" s="0"/>
              <component _id="2074" p="2074" t="AIT" v="5" ct="-1" e="true" r="10528" rx="10528"
rn="7520" x="61" s="0"/>
              <component _id="2177" p="2177" t="Type B" v="11" ct="20" e="true" r="1000160" rx="1001664"
rn="822688" x="62" s="0"/>
            </component_coll>
            <ecm_coll></ecm_coll>
            </service>
          </service_coll>
        </ts>
      </ts_coll>
    </elements>
  </data>
</response>
```

This lists information for the specified service, along with a list of components. The **p** value gives the PID of the component. It is possible to get more information for each component by looking up PID data in the **pid_coll**. For example PID 111 in the component collection yields:

```
http://<ip>/txp_get_tree?path=/data/elements/ts_coll/[290000]/pid_coll/[111]

<response request_id="2583407843">
  <status status="0" status_text="OK"/>
  <data>
    <elements>
      <ts_coll>
        <ts _id="290000">
        <pid_coll>
          <pid _id="111" p="111" tei="false" pri="false" ce="false" e="305" c="true" h="53317"
t="0" s="0" r="335392" rx="1094912" rn="0" ra="334339" rax="417059" ran="192812" y="false"/>
        </pid_coll>
        </ts>
      </ts_coll>
    </elements>
  </data>
</response>
```

3.3 Find alarms for a given TS

All TS/ASI/TSoIP nodes have an alarm node mounted at the top. The alarm node consists of several sub-nodes.

```
http://<ip>/txp_get_tree?path=/data/elements/ts_coll/[290000]/alarms

<response request_id="2583407843">
  <status status="0" status_text="OK"/>
  <data>
    <elements>
      <ts_coll>
        <ts _id="290000">
          <alarms>
```

```
            <config filter_name="__subid__100007"/>
            <status>
              <severities top_severity="3" n_total="4" n_non_filtered="10" n_critical="0" n_major="0"
n_minor="0" n_warning="2" n_filtered="2" change_cnt="240"/>
              <current_alarms_coll>
                <aa _id="71160" seqence_no="71160" alarm_id="1140" subid1="1" subid2="100007"
subid3="101" cat="2" sev="3" on_time="2015-01-20 18:57:59;691" name="CC error [PID: 101]" src_text="IP
1 > TS" details="PID 101, Jumps: 15=>7,15=>7,15=>7,15=>7,15=>7,15=>7,15=>7,15=>7,15=>7,15=>7,15=>7"/>
                <aa _id="71161" seqence_no="71161" alarm_id="1140" subid1="1" subid2="100007"
subid3="111" cat="2" sev="3" on_time="2015-01-20 18:57:59;692" name="CC error [PID: 111]" src_text="IP
1 > TS" details="PID 111, Jumps: 4=>14,4=>14,4=>14,4=>14,4=>14,4=>14,4=>14,4=>14,4=>14,4=>14,4=>14"/>
              </current_alarms_coll>
              <filtered_alarms_coll>
                <aa _id="71042" seqence_no="71042" alarm_id="1801" subid1="1" subid2="100007"
subid3="0" cat="2" sev="1" on_time="2015-01-20 18:47:34;284" name="TS-ID incorrect" src_text="IP
1 > TS" details="Current TS-ID 1093, expected 0"/>
                <aa _id="71043" seqence_no="71043" alarm_id="1812" subid1="1" subid2="100007"
subid3="0" cat="2" sev="1" on_time="2015-01-20 18:47:34;285" name="TS rate too high" src_text="IP
1 > TS" details="Measured max 38048.2, limit 0 [kbit/s]"/>
              </filtered_alarms_coll>
            </status>
          </alarms>
        </ts>
      </ts_coll>
    </elements>
  </data>
</response>
```

Under the status node there is the severities node containing a top level summary, giving the number of current alarms of a given severity. The **current_alarms_coll** contain a list of all the currently active alarms for this element. The **filtered_alarms_coll** contains a list of active, but filtered alarms. Filtered alarms are not visible, nor counted when determining the top level severity. Alarms are filtered due to user configurations.

The config node contains a field called **filter_name**. This gives the name of the alarm config, which may be found be querying:

```
http://<ip>/txp_get_tree?path=/data/alarms/filtering/filterset_coll/[__subid__100007]

<response request_id="2583407843">
  <status status="0" status_text="OK"/>
  <data>
  <alarms>
    <filtering>
      <filterset_coll>
        <filterset _id="__subid__100007" name="__subid__100007" parent_name="TS Default" type="gid">
          <f_coll>
            <f _id="0" num="0" alarm_id="1120" subid1="-1" subid2="-1" subid3="-1" o_flg="1"
sev="4" offtime="10.000000" lim1="0.000000" lim2="0.000000" log="true" trap="true"/>
          </f_coll>
        </filterset>
      </filterset_coll>
    </filtering>
  </alarms>
  </data>
</response>
```

The filterset collections gives a list of all alarm settings that has been overriden by the user relative to the default filter.

4 Tips

4.1 Show bindings in GUI

On the TXP page in the GUI there is an option called Show TXP path info as seen in **Figure 4.1**.

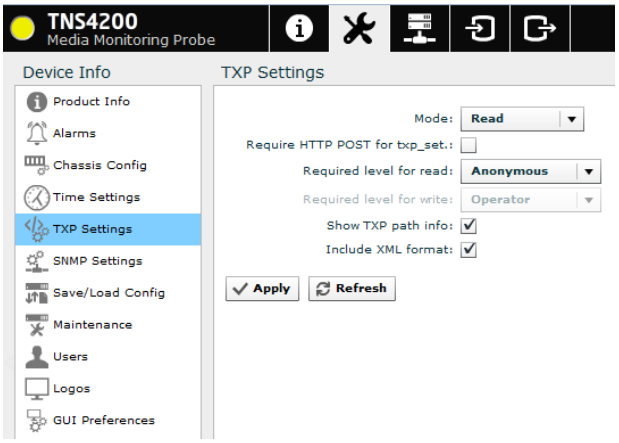

Figure 4.1 TXP settings

By enabling this option you may do a mouseover on many fields to detect their TXP paths. This will be shown in the botton right of your GUI.

The other option, Include XML format, will when clicked copy the TXP request to your clipboard when making a configuration change.