MuxXpert

Real-Time Transport-Stream Multiplexer

Table of Contents

1. Introduction

The DTC-700 MuxXpert is a comprehensive software package designed to (re-)multiplex MPEG-2 Transport Streams. The MuxXpert is intended to be installed by the user on any qualifying PC and work in conjunction with DekTec input and output devices. The specifications can be found in chapter 2.

The application has integrated Transport-Stream file players for inserting local content from disk into live streams. The players can be manually controlled from the GUI and from a play list. The GUI of the MuxXpert displays information about the services, service components and bit-rates present in the input and output Transport Streams. Chapter 5 gives an overview of the GUI and clarifies the options.

Flexible configuration allows the user to specify the newly created Transport Stream, including PID-remapping, PID-filtering, service-remapping, descriptor filtering and addition, generation and insertion of PSI/SI tables and more. Chapter 6 explains how to create your own configuration.

The MuxXpert is the ideal application for many scenarios requiring (re-)multiplexing. In head ends of all sizes the application is perfectly suited for generating new Transport Streams from existing live streams. In the lab one can generate fully customized Transport Streams from files, in any mixture with live feeds. In all applications it doesn't matter whether the Transport Streams are carried on DVB-ASI, IP or RF modulated.

2. Specifications and Minimum Requirements

2.1. Key Attributes

2.1.1. Maximum configurations

Number of TS inputs	Number of File Players	Number of TS Outputs
0 .. 64*	0 .. 8	1
0 .. 48*	0 .. 12	1
0 .. 32	0 .. 16	1 .. 2
0 .. 16	0 .. 8	1 .. 4
0 .. 16	0 .. 4	1 .. 8
0 .. 24	0	1 .. 8
0 .. 8	0 .. 32	1 .. 2
0 .. 8	0 .. 16	1 .. 4
0 .. 4	0	1 .. 64*

* In combination with DekTec TS-over-IP and ASI adapters

Note: A limited number of channels are shown in the GUI.

Parameter	Value
Sum of Input Rates	0…300 Mbit/s
Sum of Output Rates	0…300 Mbit/s

2.2. PC Requirements

Platform	Windows XP/2k3/Vista/7/8/10 with .NET v2.0
Processor	P4@3.0Ghz*; Core 2* or Core i7* if the number of inputs, outputs and file players is more than 24.
RAM	2GB 3GB if the number of inputs, outputs and file players is more than 24

* Or equivalent AMD processor

2.3. Supported DekTec Adapters

The MuxXpert supports a mix of DekTec input and output adapters. The following DekTec PCI cards are supported:

Type	Description	Input(s)	Output(s)
PCI			
DTA-100	DVB-ASI Output Adapter		1x ASI
DTA-102	DVB-SPI (LVDS) Output Adapter		1x SPI
DTA-105	Dual DVB-ASI Output Adapter		2x ASI
DTA-107	DVB-S Modulator		1x L band
DTA-107S2	DVB-S2 Modulator		1x L band
DTA-110	ATSC/QAM modulator		1x UHF
DTA-111	Multi-standard modulator		1x VHF,UHF
DTA-112	ATSC/DVB-T/QAM modulator		1x VHF,UHF
		1x ASI*	
DTA-115	Multi-standard modulator		1x VHF,UHF
		1x ASI*	
DTA-116	Multi-standard modulator		1x 36-MHz I/F, digital I/Q
		1x ASI*	
DTA-117	Multi-standard modulator		1x 44-MHz I/F, digital I/Q
		1x ASI*	
DTA-120	DVB-ASI Input Adapter	1x ASI	
DTA-122	DVB-SPI (LVDS) Input Adapter	1x SPI	
DTA-124	Quad ASI/SDI Input Adapter	4x ASI	
DTA-140	DVB-ASI Input+Output Adapter	1x ASI	1x ASI
DTA-145	Dual ASI/SDI Adapter	2x ASI**	
DTA-160	Gigabit TS-over-IP + Triple ASI Ports	1x TSoIP*** 3x ASI*	
PCI Express			
DTA-2107	Multi-standard L-band modulator		1x L band
DTA-2111	Multi-standard modulator		1x VHF,UHF
DTA-2115	All-standard, all band modulator		1x VHF,UHF, L band / 8x VHF,UHF
DTA-2131	Multi-standard SDR Receiver	1x VHF,UHF	
DTA-2135	DVB-T Receiver	2x VHF,UHF	
DTA-2136	Dual-Channel QAM Receiver	2x VHF,UHF	2x ASI
DTA-2137(C)	Dual-Channel DVB-S2 Receiver	2x L band	2x ASI
DTA-2138(B)	DVB-C2/DVB-T2 Receiver	1x VHF,UHF	
DTA-2139	Twelve-Channel QAM Receiver	12x VHF,UHF	
DTA-2144	Quad ASI/SDI Adapter	4x ASI*	
DTA-2145	Dual ASI/SDI Adapter	2x ASI**	
DTA-2152	Dual HD-SDI Adapter	2x ASI**	

DTA-2154	Quad HD-SDI Adapter	4x ASI**	
DTA-2160	Gigabit TS-over-IP + Triple ASI Ports	1x TSoIP*** 3x ASI*	
DTA-2162	Dual Gigabit TS-over-IP	2x TSoIP***	
DTA-2174	Quad 3G-SDI Adapter	4x ASI**	
DTA-2180	H.264 HD contribution encoder	1x Encoder	
USB*			
DTU-205****	FantASI ASI/SDI output		1x ASI
DTU-215****	Multi-standard VHF+UHF modulator		1x VHF,UHF
DTU-225****	FantASI ASI/SDI input	1x ASI	
DTU-245****	FantASI ASI/SDI input+output	1x ASI	1x ASI
DTU-315****	All-standard, all band modulator for USB-3		1x VHF,UHF, L band
Networked Adapters			
DTE-3100	Networked ASI Output		1x ASI
DTE-3120	Networked ASI Input	1x ASI	
DTE-3137	Networked DVB-S2 Receiver	1x L band	

* Each port can be used as Output or Input
** 2x Output, or 1x Input + 1x Output
*** The Gig-E can carry multiple TS Inputs and TS Outputs
**** For evaluation and demos, not recommended for operational use.

3. MuxXpert Software Installation

For the MuxXpert Software installation and the MuxXpert license installation instructions, see the 'DTC-700 Installation' document, which is included in the install package.

4. MuxXpert API

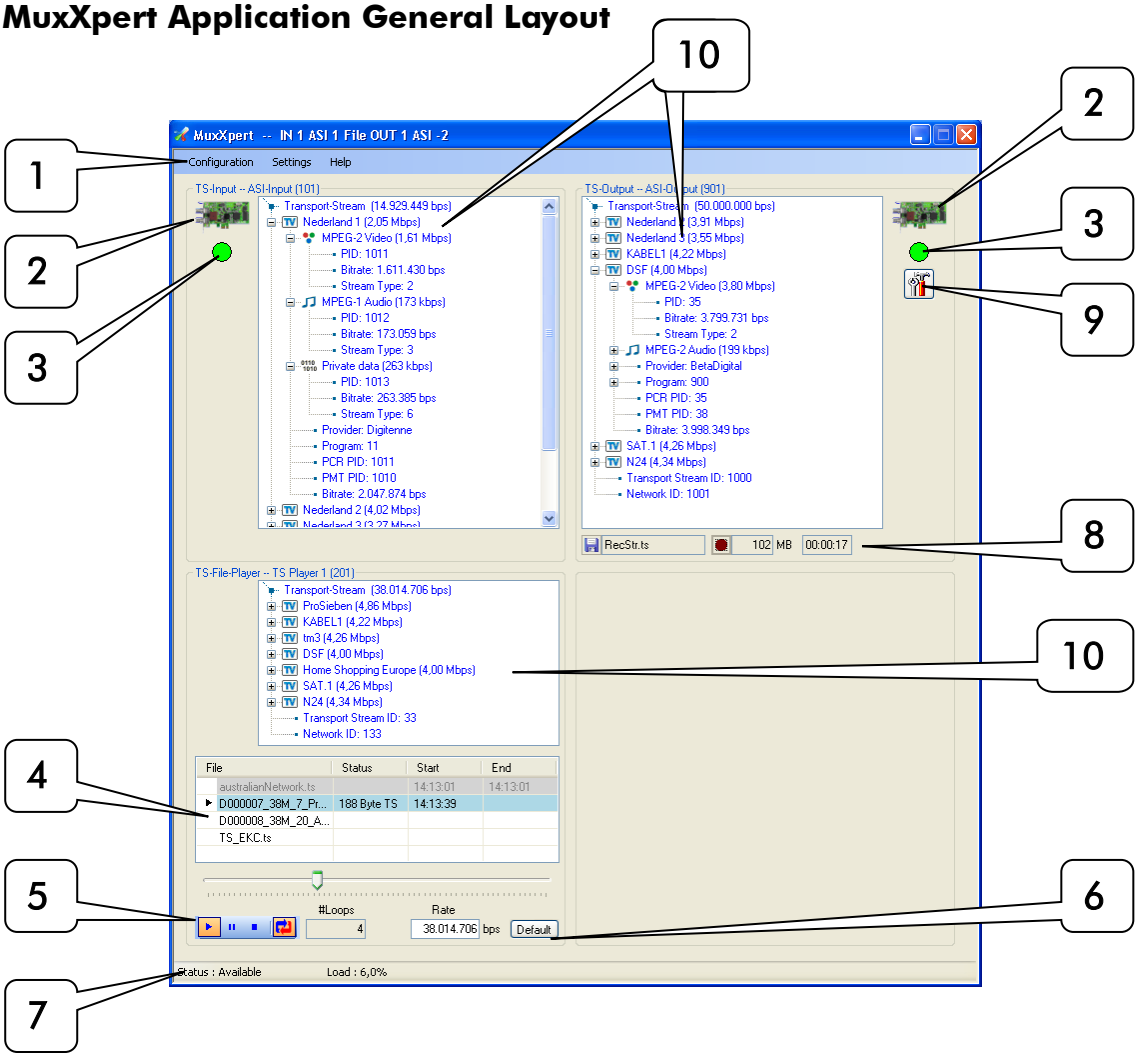
The MuxXpert API is intended for developers integrating real-time Transport-Stream multiplexing functions into their own application. It provides a number of class libraries that allow users to use the MuxXpert multiplexing functions. For the MuxXpert API documentation, see the 'MuxXpert-API Help', which is installed when during installation the MuxXpert-API SDK feature is selected.

5. MuxXpert Overview

5.1. Launching the MuxXpert

Start the MuxXpert program from the Start Menu: **start** >> **All Programs** >> **DEKTEC** >> **MuxXpert**. This will launch the MuxXpert software.

5.2. MuxXpert Application General Layout



1. Menu Bar: The top area of the MuxXpert application contains three menus: Configuration, Settings and Help.

2. Adapter Info: This area displays the information of the adapter used for the outgoing/incoming Transport Stream

3. Stream Status: This area displays the status of the outgoing/incoming Transport Streams.

4. Play-list Pane: This area provides control of the play-list. It also contains information regarding the current play-out operation.

5. Play-out Control: This area provides control of the play-out process.

6. Play-out Bit-rate: This area of the application displays and allows you to modify play-out bit-rate.

7. Status Bar: This area displays the status of the re-multiplexing application.

8. Stream Recording: This area provides control of the Transport Stream recording.

9. Adapter Specific Settings: This button provides control of the adapter specific settings.

10. Stream Analysis Window Pane: This area displays the contents and the basic parameters of the incoming/outgoing Transport Stream

5.3. MuxXpert Application Control

5.3.1. Menu Bar

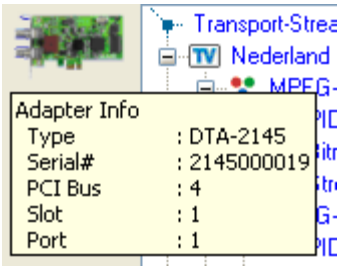
The menu bar contains three menus:

1. Configuration. From the configuration menu the user can browse to and select a (new) re-multiplexer configuration file. In case a new or different configuration of input and output ports is used, the application has to restart.
2. Settings. From the settings menu the user can perform the following settings:
    a. Enable/disable the display of event messages. Event messages show unexpected events and can indicate possible configuration errors.
    b. Detailed logging can be enabled by selecting a log-level. Setting a log-level will decrease the performance of the application. Therefore, it should not be used in normal operation.
    c. Enable/disable remote control of the file players in the MuxXpert and specifying the IP port at which the file players will listen.
    d. Enable/disable the MuxXpert API and specifying the IP port that will be used.
    e. Character Code Table selection. The selected Character Code Table will be used for decoding of the service name and service provider.
3. Help. From the help menu the user can start the License Manager to show the currently available licenses and to install new licenses. Further, the version of the currently running MuxXpert can be displayed.

5.3.2. Adapter Info

This are displays the information of the adapter used for the outgoing/incoming Transport Stream.

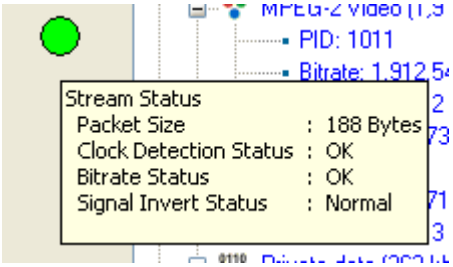
When the mouse is moved over the adapter picture, detailed information about the adapter is shown in a tooltip.



5.3.3. Stream Status

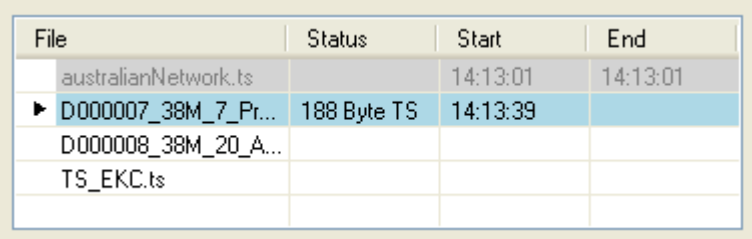
This area indicates the status of the incoming/outgoing Transport Stream. The status of the Transport Stream is indicated by a globe. A green globe defines a valid Transport-Stream input or output. A yellow globe with an exclamation mark defines a bad input or output condition and needs attention.

When the mouse is moved over the Stream status box, detailed information about the stream is shown in a tooltip.

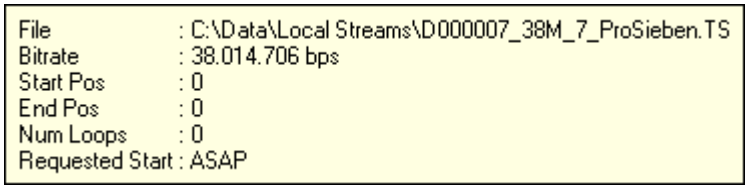


5.3.4. Play-list Pane

The play list pane shows the status of the played-out files, their start and end times. The currently playing file is marked light blue, files that were played out in the past are marked light grey and files that will be played out in the future are not marked.

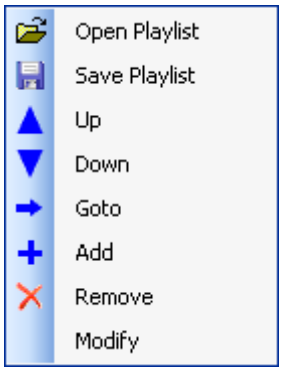


Detailed information in a tooltip will be displayed when the mouse cursor is moved over a file in the play-list.



5.3.4.1. Play-list Control

To see the options for the play-list control, place the mouse cursor in the play-list pane and press the right-mouse button. Invalid options are light grey and can not be selected.



5.3.4.2. Open Playlist Command

The Open Playlist command will prompt you to browse for a play-list file on your PC. The MuxXpert will, by default, look for files of *.xml extensions. However, you may select a play-list file with any extension by selecting "All Files" in the file of type pull-down. Keep in mind that although the file does not need to have a *.xml extension, it must be a valid play-list file. See appendix A.

In case the content of a selected play-list file is modified, the new play-list is loaded automatically.

5.3.4.3. Save Playlist Command

The Save Playlist command will prompt you to browse for the file on your PC, were the current play-list is stored.

5.3.4.4. Up Command

The Up command will move the selected file in the play-list one position up. When the position of the selected file becomes earlier than the currently playing-out file, then the selected file is marked grey and it will not be played-out. In case a start-time is attached to the selected file then it is not possible to move this file up such that its start-time is later than the start-time of a file later in the play-list.

5.3.4.5. Down Command

The Down command will move the selected file in the play-list one position down. When the position of the selected file becomes after the currently playing-out file, then the possible grey mark is removed and the selected file will be played-out again. In case a start-time is attached to the selected file then it is not possible to move this file down such that its start-time is earlier than the start-time of a file earlier in the play-list.

5.3.4.6. Goto Command

The Goto command will stop the play-out of the current file, and will start playing-out the selected file.

5.3.4.7. Add Command

The **Add** command will allow you to browse for a file on your PC that has to be added to the play-list. You must enter play-out bit-rate for the selected file.

The Start Position and End Position allow you to select a part of the file to be played-out. If the End Position is set to "0" the file is played-out till the end of the file.

The Repetitions field allows you to select the number of times the file has to be repeated.

If you check the Select Date and Time checkbox, you can enter a start-date and start-time for the selected file. In case the checkbox is unchecked the file will be played-out as soon as possible when the files earlier in the play-list are finished.

5.3.4.8. Remove Command

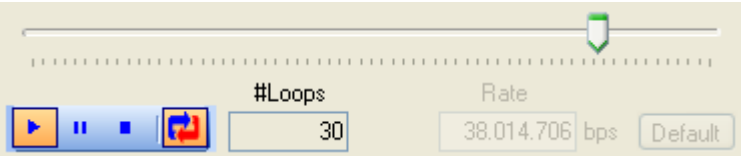
The **Remove** command will remove the selected file from the play-list. If the currently playing-out file is removed, the next file in the play-list will start.

5.3.4.9. Modify Command

The **Modify** command will allow you to modify the settings of the selected file. Items as file, bit-rate, start-position and end-position, number of repetitions and start date and time can be modified.

5.3.5. Play-out Control

The Play-out Control portion of the MuxXpert application is where you can manipulate the play-out operation of a file. This area also provides information regarding play-out progress.



5.3.5.1. Sliding Pointer

The Sliding Pointer serves two purposes. The first is to provide a reference indicating the position in the file that is playing. The second is to allow you to start play-out at a specific location. To define a starting location, stop file play-out, and place the mouse cursor over the pointer and hold the left-mouse button down while dragging the pointer. Press the play button to start playing. You may also change the play-out point while the file is playing.

5.3.5.2. Play, Pause and Stop Buttons

You will left-mouse-click on these buttons to start, pause, or stop a play-out operation.

5.3.5.3. Loop Button

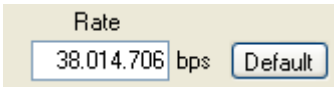
This button allows you to loop a file or the complete play-list.

5.3.5.4. Number Loops

The number loops shows you how many times a file has looped since file play-out was originally started.

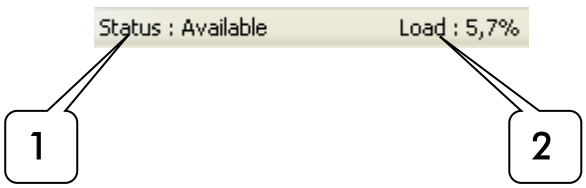
5.3.6. Play-out Bit-rate

This area of the application displays and allows you to modify play-out bit-rate of the file player. The play-out bit-rate textbox shows the currently used play-out bit-rate. Default this is the bit-rate as set in the play-list. A user can overrule the bit-rate in play-list by entering a new bit-rate in the play-out bit-rate textbox. In that case, the textbox becomes yellow. To return to the default play-out bit-rate as entered in the play-list, the user can use the **Default** button.



5.3.7. Status Bar

The Status Bar appears in the lower right corner of the MuxXpert application. The Status Bar gives you a brief overview of re-multiplexing status.



1. **Status:** This box indicates the status of the re-multiplexing operation. Possible statuses:

a. "Initializing". Indicating that the re-multiplexer is starting up.
b. "Available". Indicating that the re-multiplexer is up and running.
c. "Failed". An unrecoverable error has occurred. Possible causes: not enough memory installed, adapter already in use, total input rate is much too high.
d. "Configuration Mismatch". Indicating a possible installation problem.
e. "NoLicense". Indicating that no valid license is found.

2. **Load:** The re-multiplexing capabilities are amongst others bounded by processor performance. Therefore the processor load caused by the re-multiplexer is shown in the status bar.

5.3.8. Stream Recording

The MuxXpert allows you to record an outgoing Transport Stream. Each outgoing stream is provided with the option to record the stream. You can have multiple recordings at the same time. However the number of recordings and the recorded bit-rate is limited by the disk performance.

Note: The record function is an option of the live outputs. It is not possible to have a file-only output.

5.3.8.1. Output File Selection

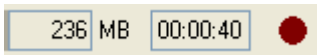
This button allows you to select an output file where recorded data is stored. The textbox next to the button shows the current selected file. When the mouse cursor is moved over the textbox is show the full path.

5.3.8.2. Start/Stop Recording

The start recording button  allows you to start the recording of the outgoing Transport Stream. When it is pressed it changes into a stop recording button  to stop the recording again.

5.3.8.3. Recording Progress

The recording progress and activity is shown in different ways: i) the number of Mega-bytes written to file so far; ii) the duration of the recording; iii) a red blinking recording busy indicator; iv) a possible ERROR indication.

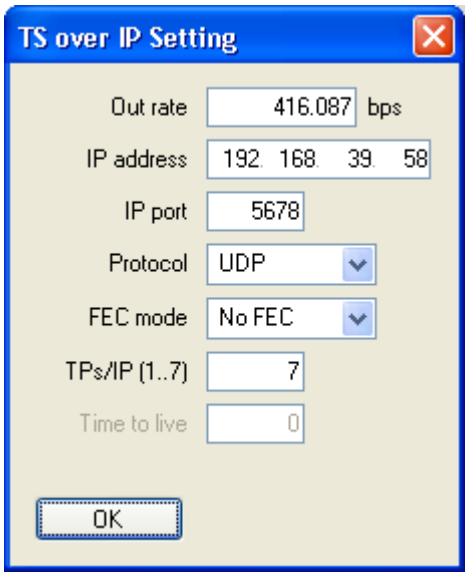


5.3.9. Adapter Specific Settings

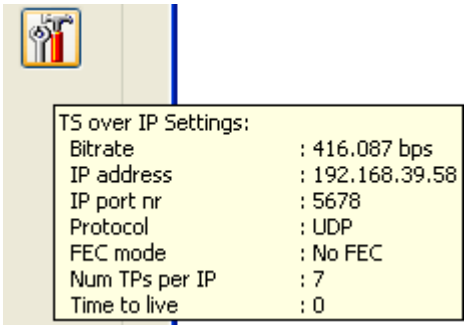
The Setting button allows you to read and set the adapter specific settings, such as: bit-rate, IP-address, RF-frequency etc.

When the Setting button is clicked a setting window pops-up. Here the adapter specific settings can be modified.

Note: The new settings are not saved. Upon (re-) loading of a re-multiplexer configuration file, these settings are lost and overwritten by the ones stored in the configuration file.



When the mouse is moved over the Setting button, the adapter specific settings are shown in a tooltip.

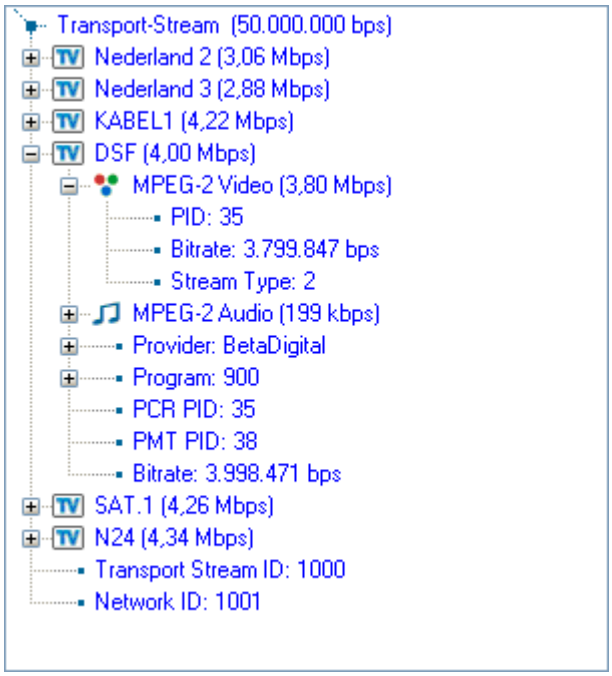


5.3.10. Stream Analysis Window Pane

The MuxXpert application includes a basic analysis feature that allows you to view the make-up of the incoming/outgoing Transport Streams. From this window one can also drag services from an input to an output.

The top line in this pane shows the bit-rate of the incoming/outgoing Transport Stream.

The Transport-Stream pane displays the services (programs) contained in the Transport Stream. You may expand the analysis down to the individual components (PID's) and even further by left-mouse clicking on the "+" signs next to the components.



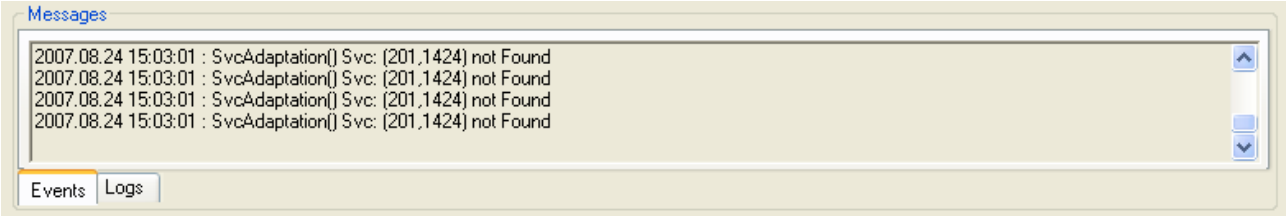
To select a service from an incoming Transport Stream for the outgoing Transport Stream, place the mouse cursor over the icon of the desired service and hold the left-mouse button down while dragging the service to the output.

To remove a service from an outgoing Transport Stream, select the service to be removed by left mouse clicking on the icon of the service and thereafter pressing the Delete key.

Note: The new service selections are not saved. Upon (re-) loading of a re-multiplexer configuration file, the service selections are lost and overwritten by the ones stored in the configuration file.

5.3.11. Message Bar

The Message Bar will be shown if the Messages are enabled in the settings menu in the menu bar (see section 5.3.1). The Message Bar can contain two tabs. The Events tab contains the event message and may indicate possible configuration errors.



Note: Not all event entries indicate an error. For example if a service is not found, this can be because:

1. The service is not in the stream. This is possibly an error, for example: no input stream or wrong service is selected
2. The service is not yet detected. This can be normal temporary situation. For example: a new configuration is loaded and the MuxXpert has not yet found the service in the stream. When the service is found, it will not show a positive event.

Detailed logging can be enabled by selecting a log-level. The Logs tab contains detailed logging information with respect to the re-multiplexing operation.

6. Creating a Re-Mux Configuration Data File

6.1. Introduction

The *Remux Configuration Data* (*RMC-Data*) is used to describe the composition of the outgoing Transport Streams of the MuxXpert application including PSI/SI tables, custom tables, descriptors and table-assembly parameters.

The RMC-Data file is a XML-file. The permissible format of the XML file is defined by XML *Schemas*, written in XML Schema Definition Language [XSDL].

The RMC-Data Schemas are defined in the RmcDataTempl00.xsd file.

The .xsd file enables:

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

The elements and attributes in the schema are described in plain text in the RmcDataTempl00.pdf file which is also included in the MuxXpert installation.

6.2. References

[DVB SI]
EN 300 468, Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems, V1.7.1 (2006-05).

[MPEG-2 SYS]
ISO/IEC 13818-1, Information technology – Generic coding of moving pictures and associated audio information: Systems, May 27th, 1999.

[XSDL]
XML Schema Definition Language, W3C, Candidate Recommendation, October 24, 2000

[RMC XSD]
RmcDataTempl00.xsd - XML Schema for RMC Data. This document defines the permissible XML format of the Remux Configuration Data file, written in XML Schema Definition Language. It will be installed in the following directory: <muxxpert_install_dir>\Doc.

[RMC DOC]
RmcDataTempl00.pdf - XML Schema for RMC Data. This document describes the elements and attributes in the RMC-Data Schemas in plain text. It will be installed in the following directory: <muxxpert_install_dir>\Doc.

6.3. Overview

The RMC-Data file is built up from a number of sections. Each section specifies a certain item. This chapter describes step by step how to create your own RMC-Data file, by explaining the different sections in the RMC-Data file and by giving some examples.

The RMC-Data is created by specifying the following items:

1. Specification of the input and output ports used in the RMC-Data file (section 6.4).
2. Specification of the input settings (section 6.5).

3. Specification of the output streams. Per stream the following is specified:
    a. Identification of output stream (e.g. transport_stream_id) (section 6.6.1)
    b. Specification of the output settings (e.g. bit-rate) (section 6.6.2)
    c. Specification of the composition of the output stream (e.g. services in the output stream) (section 6.6.3)
    d. Specification of the stream adaptations (e.g. PID remapping) (section 6.6.4)
    e. Specification of the PSI/SI assembly (e.g. how to create the PMT) (section 6.6.5)

To create a new RMC-Data file it is recommended to start with the Empty RMC-Data-file or one of the example RMC-Data-files that are included in the installation. Thereafter you can edit the different parts.

6.4. Input and Output Ports Used.

First, the user has to specify the input, output ports and the file-players used by the application. Here the relation is specified between a Logical Transport Stream Identification (LogTsId) and a physical port of a DekTec input or output adapter or a certain file-player. Hereafter, this LogTsId can be used in the RMC-Data to refer to a certain input or output stream. The LogTsId numbers can be chosen arbitrary. However, they need to be unique. Further a descriptive name can be attached to the LogTsId. This descriptive name of the port will be displayed in the MuxXpert's GUI.

```
<!--+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+= LogTsIdIoPortMappings +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
     Specification of the input and output ports used
 -->
<LogTsIdIoPortMappings>

<!-- LogTsId :101 is an ASI input port related to the first DTA140 port 1 -->
    <LogTsIdIoPortMapping LogTsId = "101" DescrName = "ASI In 1">
        <InPort>
            <ASI>
                <DTA140 RelDvcNr = "0" PortNr = "1"/>
            </ASI>
        </InPort>
    </LogTsIdIoPortMapping>

    <!--  LogTsId :901 an ASI output port related to the first DTA140 port 2 -->
    <LogTsIdIoPortMapping LogTsId = "901" DescrName = "ASI Out 1">
        <OutPort>
            <ASI>
                <DTA140 RelDvcNr = "0" PortNr = "2"/>
            </ASI>
        </OutPort>
    </LogTsIdIoPortMapping>

</LogTsIdIoPortMappings>
```

Figure 1. Example of a specification of the input and output ports used.

The example above specifies that a DTA140 ASI input port (port 1 of the first DTA140) and an ASI output port (port 2 of the first DTA140) is used. The LogTsId of the input port is: 101 and the LogTsId of the output port: 901, also a descriptive name is specified for the ports.

6.5. Input Port Settings

Some input ports require parameter setting to operate. For example the IP-settings for an IP-input port (such as IP-address used for reception) need to be specified. The input port settings are also

used to specify a play-list for a File-Player and to specify a time correction and time offset that has to be applied on the incoming EIT-events.

```
<!-- +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ TsIns +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
     Input settings
-->
<TsIns>

   <!--  Set IP parameters for input 103-->
    <TsIn LogTsId = "103">
        <TsInPars>
            <IP IpAddress="192.168.39.67" IpPortNr="5680" IpProtocol="auto" FecMode="disable"/>
        </TsInPars>
    </TsIn>

    <!--  Settings for FilePlayer 201-->
    <TsIn LogTsId = "201">
        <!-- Apply time correction to events in the EIT tables based on the TDT in the stream->
        <TsTimeCorrection>
            <ByTsTdt/>
        </TsTimeCorrection>

        <!-- Set PlayList -->
        <TsInPars>
            <File PlayList = "C:\MyPlaylists\PlayList Torino.xml"/>
        </TsInPars>
    </TsIn>

</TsIns>
```

Figure 2. Example of specification input settings.

The example above shows the input settings for an IP-Port and a File-Player. Note that the LogTsIds that are used are specified before (section 6.4).

6.6. Specification of the Output Streams

In this part of the RMC-Data a sequence of output streams is specified. The sections below have to be repeated for each output stream.

6.6.1. Identification of Output Stream

Identification of the outgoing stream, here the relation between LogTsId and network_id, transport_stream_id (and optionally the original_network_id) have to be specified. The LogTsId refers to an output port that is specified before (section 6.4). The network_id and tansport_stream_id are the DVB-attributes that uniquely identify this outgoing Transport Stream. These DVB-attributes will also be used in the tables that are created for this Transport Stream. If the original_network_id is not specified it is the same as the network_id.

An example of the identification of an output stream is given below network_id 1000 and transport_stream_id 2 is assigned to the output stream with LogTsid 901.

```
<!-- LogTsId 901 identification -->
<TsOut LogTsId = "901" NwId = "1000" TsId = "2">
```

Figure 3. Example of identification of the output stream.

6.6.2. Output Port Settings

For correct operation of the output port, parameters such as bit-rate, modulation type, IP-address etc. need to be specified. Which parameters must be specified depends on the type of output port. This can be found in the RMC-Data Schemas and the detailed description (See [RMC XSD] and [RMC DOC].

An example of a specification of an output setting is given below. The bit-rate and the IP-parameters of the IP-output port are specified.

```
<!-- IP setting of the output port -->
<TsOutPars>
    <IP Bitrate = "50000000" IpAddress = "192.168.39.54" IpPortNr = "5678" IpProtocol = "udp" FecMode = "disable"
        NumTpPerIp = "7"/>
</TsOutPars>
```

Figure 4. Example of specification of the output settings.

6.6.3. Composition of the Output Stream

This part of the RMC-Data file allows the user to specify which services, service components and other streams have to be included in the output stream.

Composition of the output stream: The building of an outgoing Transport Stream from selected services and components from incoming Transport Streams.

There are two levels of composition:

1. The Transport-Stream wide composition allows the user to specify a collection of services, service components, EMM-streams and unreferenced Elementary-Streams for the output stream.
2. The Service wide composition allows the user to refine the collection of service components within a single service.

Note that when a service is selected to be included output stream, default this includes all its service components and its PSI/SI information.

6.6.3.1. Transport-Stream Wide Composition

The Transport-Stream wide composition allows the user to specify a collection of services, service components, EMM-streams and unreferenced Elementary-Streams for the output.

The figure below illustrates the Transport-Stream wide composition. It can be seen as a series of filters, that passes (or drops) the services and components from different sources to the output stream.

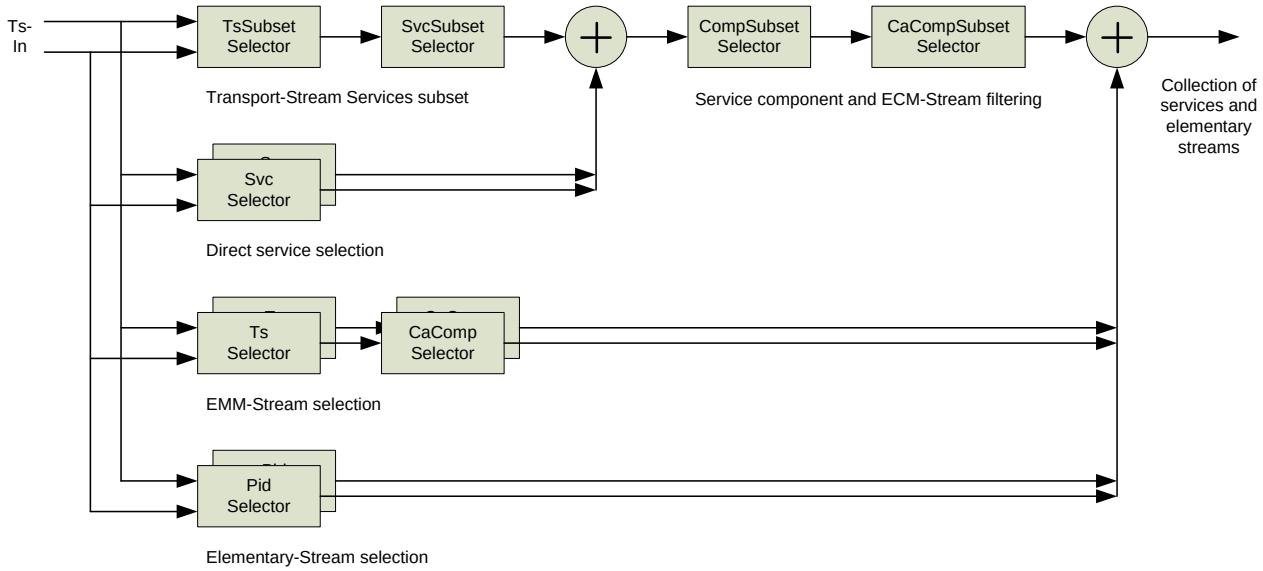


Figure 5. Illustration of the Transport-Stream wide composition.

The filtering is achieved with so-called selectors and subset selectors. See also [RMC DOC] and [RMC XSD] for the selector details.

1. There are two methods to add services to the output stream. A mix of both methods can be used. If a service or service component is selected multiple times via different specifications, the service and the service components will appear only once in the output stream.

   a. Transport Stream services subset. This method is useful for in case all services of an input stream or all services except for some services have to be passed. First a Transport Stream subset selector is used to get all services from the selected incoming Transport Streams. Thereafter a service-subset selector can be specified to drop (or pass) certain services from the collection.

   b. Direct services selection. Specifies each service that has to be included in the output stream. This is a very common way of specifying the composition of the output stream.

   After the services have been selected for the output, the user may specify: i) service component filtering and ii) ECM-filtering.

   Service component filtering; optionally a component subset selector can be specified to drop certain service components globally over all services in the collection of services. For example this can be used to drop certain teletext languages in all services. If it is not specified, all service components are passed

   ECM-filtering; optionally a CA-component subset selector can be specified to drop (or pass) certain ECM-Streams globally over all services in the collection of services. If it is not specified, all ECM-Streams are passed.

2. EMM-Stream selection. Can be used to add EMM-streams from a particular input stream and from a particular conditional-access system to the output stream.

3. Elementary-Stream selection. Can be used to add unreferenced elementary streams (streams that are not referenced in PSI) to the output stream.

Examples of Transport-Stream wide compositions are given below.

The first example below shows the selection of a subset of services within a subset of input streams. First a subset of input streams is specified; thereafter a service subset is specified.

```
<!-- Specificaion of a subset of services from a subset of input streams-->
<TsSvcsSubset>

 <!--  1. Transport Stream (sub)selection  from which to get all services-->
 <TsSubsetSelector SelectionMethod="positive">

   <!-- The input stream LogTsId=101 is selected and -->
  <TsSelector>
   <ByLogicalTsId LogTsId="101"/>
  </TsSelector>

   <!-- the input stream identified with the DVB-attributes network_id=1000, transport_stream_id=1 is selected -->
  <TsSelector>
   <DvbQualified OnwId="1000" TsId="1"/>
  </TsSelector>

 </TsSubsetSelector>


 <!-- 2. Service (sub)selection specification of a  from  the (sub)selection of the Transport Streams
        (because SelectionMethod=negative, this subselection is used for dropping services)  -->
 <SvcSubsetSelector SelectionMethod="negative">

   <!-- drop service-identified with the DVB-attributes network_id=1000, transport_stream_id=1 and service-id=1 -->
  <SvcSelector>
   <DvbQualified OnwId="1000" TsId="1" SvcId="1" />
  </SvcSelector>

   <!-- drop service with service-id=1 and LogTsId=101 -->
  <SvcSelector>
   <Short LogTsId="101" SvcId="1"/>
  </SvcSelector>

 </SvcSubsetSelector>

</TsSvcsSubset>
```

Figure 6. Example of selection of a subset of services within a subset of input streams

The second example below shows the direct selection of services; it specifies each service that has to be included in the output stream.

```
<!-- Specification of a direct selected service-->
<DirectSelectedSvcs>

   <!-- Add service with service-id=11 from LogTsId=102 to the output -->
 <SvcSelector>
  <Short LogTsId = "102" SvcId = "11"/>
 </SvcSelector>

 <!-- Add service-identified by network_id=999, transport_stream_id=2 and service-id=3 to the output -->
 <SvcSelector>
  <DvbQualified OnwId="999" TsId="2" SvcId="3"/>
 </SvcSelector>

</DirectSelectedSvcs>
```

Figure 7. Example of direct service selection

The example below shows the filtering of service components and ECM-Streams. This example specifies the dropping of service components and the passing of certain ECM-Streams.

```
<!-- Specification which components not to include (negative selection method)    in the output-->
<CompSubsetSelector SelectionMethod="negative">

<!-- drop all components were the Teletext language is only Dutch-->
  <CompSelector SingleGroupFlag="group">
    <CombinedSelection>
      <TeletextLanguage LanguageCode="dut"/>
    </CombinedSelection>
  </CompSelector>

</CompSubsetSelector>

<!-- Specification which ECM-Streams to include (positive slection method) in the output-->
<CaCompSubsetSelector SelectionMethod="positive">
  <!-- pass the ECM-Streams that use CA-System-Id 18012 and 18013.-->
  <CaCompSelector CasId="18012"/>
  <CaCompSelector CasId="18013"/>
</CaCompSubsetSelector>
```

Figure 8. Example of service component and ECM-filtering.

The example below shows the selection of certain EMM-Streams for the output stream. This example specifies an input stream and the CA-System-Ids of the EMM-streams to be included in the output stream.

```
<!-- Specification which EMM-streams to include  (positive slection method) in the output -->
<EmmStreams>
  <EmmsSubset>

    <!--Use input LogTsId=101 for the EMM selection-->
   <TsSelector>
     <ByLogicalTsId LogTsId="101"/>
   </TsSelector>

   <!-- Selects EMM streams with CA-system-id 18012 and 18013 (from input 101) -->
   <CaCompSubsetSelector SelectionMethod="positive">
     <CaCompSelector CasId="18012"/>
     <CaCompSelector CasId="18013"/>
   </CaCompSubsetSelector>

  </EmmsSubset>
</EmmStreams>
```

Figure 9. Example of EMM-Stream selection

The example below shows the selection of certain elementary streams for the output stream. This example specifies PIDs of input streams to include in the output stream.

```
 <!-- Specification which unreferenced elementary streams to include in the output-->
<ElemStreams>

  <!-- add elementary stream  with PID=1200 from LogTsId=101 to the output -->
 <PidSelector>
   <ByLogicalTsId LogTsId="101" Pid="1200"/>
 </PidSelector>

  <!-- add elementary streams  with PIDs =1300..1399 from LogTsId=101 to the output -->
 <PidSelector>
   < ByLogicalTsIdRange LogTsId="101" MinPid="1300" MaxPid="1399"/>
 </PidSelector>

</ElemStreams>
```

Figure 10. Example of elementary stream selection

6.6.3.2. Service Wide Composition (optional)

In many cases the Transport-Stream wide composition is sufficient. Then this section can be omitted. However, in some case a user may want to refine a service that was selected for the output stream in the "Transport-Stream Wide Composition" or a user may want to create a new service from scratch. In such cases the service wide composition can be repeated for each service that needs to be composed or refined.

The figure below illustrates the service wide composition. It can be seen as a series of filters that drops or adds components from different sources to the selected service.

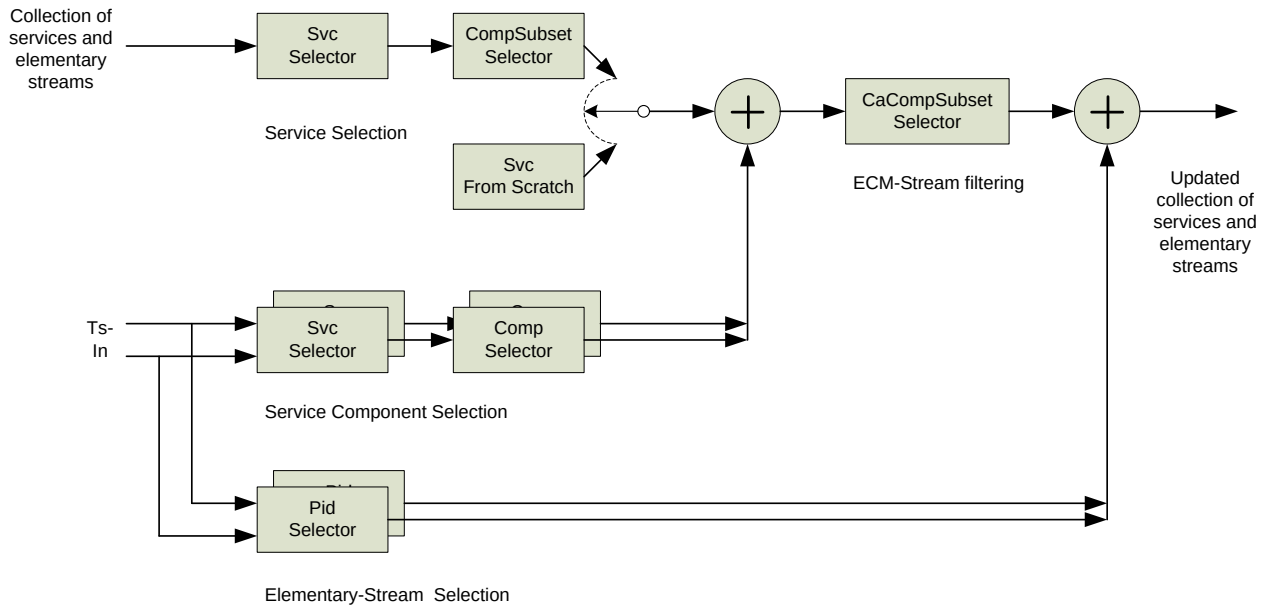


Figure 11. Illustration of the service wide composition.

1. First the service that needs to be refined must be specified. This can be:
   a. A service that is selected from the Transport-Stream Wide composition. Thereafter, optionally a component subset selector can be specified to drop certain service components from the selected service.
   b. A new service that is created from scratch. Initially this is a service without any service components. Components and elementary streams can be added to this service.
   Notes:
   In case of newly created services the user needs to specify the PCR-stream, service type, PMT-descriptors and SDT-flags and descriptors. This can be specified in the section "Service Adaptation" (see section 6.6.4.3).
   The user can refer to the new service by using the service-id of the new service and the identifying attributes of the output stream.
2. The user can specify service components that must be added to this service, by specifying a service that needs to be selected from an input stream and a component selector that selects a component from the service. Note that the selection of a service component includes the selection of the related PSI and possible related ECM-Streams. The MuxXpert can use this PSI information when the PMT is created for this service.

3. Further, a CA-component subset selector can be specified to drop (or pass) certain ECM-Streams within this service. If it is not specified, all ECM-Streams are passed.
4. Finally, the user can specify unreferenced elementary streams that must be added to this service, by specifying a PID that must be included in this service.

   Note: The addition of an unreferenced elementary stream does not include PSI. Possibly the user needs to specify the stream type and the PMT-ES-info descriptors. This can be specified in the section "Service Wide Component Adaptation" (see section 6.6.4.2).

Below an example is given of the service wide composition. The example below shows the refinement of a service that was selected for the output stream in the Transport-Stream wide composition. The selected service is extended with a service component from another service.

```
<!-- Specification of one service wide composition-->
<SvcWideComposition>

 <!-- First select the service that is composed/refined
      this is a service from the Tranpsort Stream Wide composition collection
      identified by service-id=11 from input LogTsId=102 -->
 <SelectedSvc>
  <SvcFromTsComposition>
   <SvcSelector>
    <Short LogTsId="102" SvcId="11"/>
   </SvcSelector>
  </SvcFromTsComposition>
 </SelectedSvc>

 <!-- A service component is added to this service
      The service component added is a dutch teletext service component (streamtype=6 and language=dut)
      taken from service with service-id 99 from input LogTsId=102
      (only appropriate if the services have the same time-base)-->
 <SvcComponents>
  <SvcComponent>
   <SvcSelector>
    <Short LogTsId="102" SvcId="99"/>
   </SvcSelector>
   <CompSelector SingleGroupFlag="single">
    <CombinedSelection>
     <StreamType StreamType="6"/>
     <TeletextLanguage LanguageCode="dut"/>
    </CombinedSelection>
   </CompSelector>
  </SvcComponent>
 </SvcComponents>

</SvcWideComposition>
```

Figure 12. Example of service wide composition

6.6.4. Adaptations of the Output Stream (optional)

After the output stream has been composed, the contents of the outgoing Transport Stream (services and components) can be *adapted*:

1. For *existing* services and components, the user may specify adaptations to change fields and descriptors in the PSI and SI describing those services and components.
2. For *new* services and components, the PSI and SI fields to be inserted for those services and components may be specified.

In many cases the MuxXpert will adapt the fields autonomously, for example in case the service-ids in the re-multiplexed stream are conflicting.

The MuxXpert will also perform the *attribute relocation*. This is the function to update references to DVB-objects when that object is re-multiplexed and its attributes are changed. The primary example of relocation is changing the `service_id` field in PSI/SI tables and descriptors when the referenced

service is re-multiplexed onto a different service-id. The MuxXpert also performs the attribute relocation for: `PID`, `component_tag`, `network_id`, `original_network_id` and `transport_stream_id`.

Default the MuxXpert allocates a free `PID` to each elementary stream that is included in the output stream likely this differs from the incoming `PID`. To suppress the re-allocation of `PIDs`, the KeepOrigPid attribute can be set to "true" then the outgoing `PID` values are equal to the original `PID` values if the `PID` values are unique and if the `PID` value is not overruled by PID-assignments.

Note: The user should stay using the original DVB-attributes for identifying a DVB-object, after these attributes have been changed.

The user can specify the following kind of adaptation:

1. Service component adaptation. There are two levels of service component adaptation:
    a. The Transport-Stream wide component adaptation allows the user to specify a component adaptation that will be applied on the selected service components from the entire collection of services (that will become the output stream).
    b. The service wide component adaptation allows the user to specify a component adaptation that will be applied on the selected service components from a single service.
2. Service adaptation.
3. PCR adaptation.
4. Transport-Stream wide elementary stream adaptation.
5. Transport-Stream wide EMM-Stream adaptation
6. Service wide ECM-Stream adaptation

The figure below illustrates the output adaptations. It can be seen as a series of optional adaptors that selects certain objects from the output stream, adapts them and merges them with the output stream again. At the end it outputs an adapted collection of services, service components and elementary streams.

Note that adaptations modify the PSI/SI describing the service components. The contents of the component itself are not modified, with the exception of changing the PID.

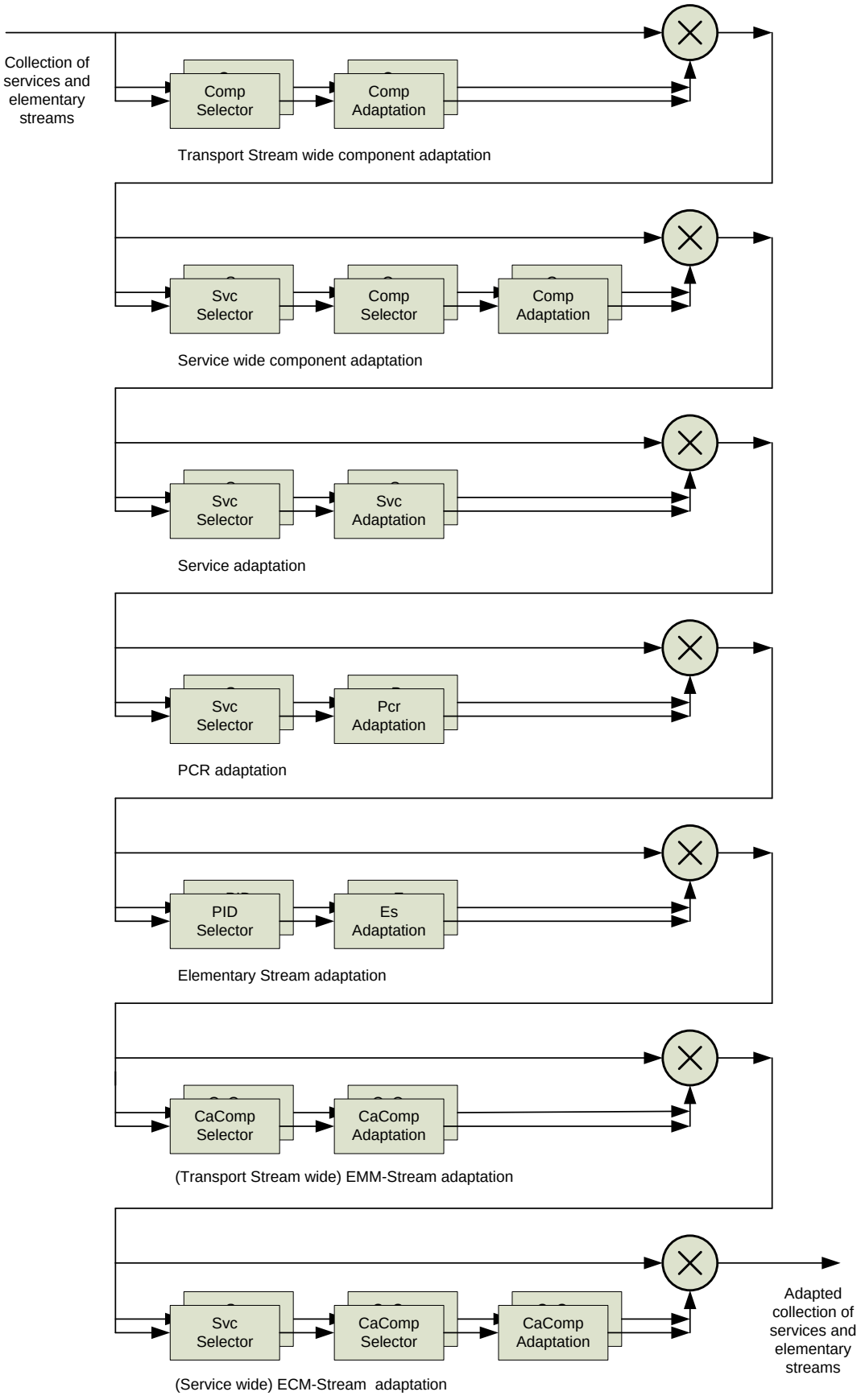


Figure 13. Illustration of adaptation of the output stream.

6.6.4.1. Transport-Stream Wide Component Adaptations (optional)

The Transport-Stream wide component adaptations allow the user to specify adaptations that have to operate on all selected components in the output stream. For example it allows the user to specify: for all Hungarian audio streams in the output stream, add descriptor XYZ to the ES-info section in the PMT.

The component adaptations allow a user to specify:

1. RemoveLevel. The RemoveLevel controls the multiplex behaviour in case the sum of the incoming bit rates exceeds the outgoing bit rate for too long. The RemoveLevel can be used to assign a priority to a stream such that "unimportant" streams are removed first. Default this value is set by the MuxXpert (see section 6.7).
2. Stream-type. Specifying a (new) value for `stream_type` field for this component (PMT 2nd loop). A user should specify a value in case an unreferenced stream is added to the service (see section 6.6.3.2).
3. PID. Specifying a (new) value for the PID. This can be specified for a single component only. Default the MuxXpert allocates a free PID for each component (see section 6.7.2).
4. Component-tag. Specifying a (new) value for the `component_tag` field. The component-tag (or: Stream Identifier) is used to establish a link between PMT and EIT. This can be specified for a single component only. Default the MuxXpert allocates a free component-tag for each component (see section 6.7.3).
5. PMT-ES-Info-descriptors. Allows the user to add, remove or to replace PMT-ES-Info-descriptors. A user needs to specify a value in case an unreferenced stream is added to the service (see section 6.6.3.2.)

In the example below for all Hungarian audio streams in the output stream, a private descriptor is added to the ES-info section in the PMT. Further, for these streams the `RemoveLevel` is set to 1.

```
<!-- Specification of one Transport-Stream wide component adadaptation-->
  <ComponentAdaptation>

    <!-- do for all components in the Transport Stream that have streamtype=3 (audio) and language=hun-->
    <CompSelector SingleGroupFlag="group">
      <CombinedSelection>
        <StreamType StreamType="3"/>
        <Language LanguageCode="hun"/>
      </CombinedSelection>
    </CompSelector>

    <!-- Set Remove Level for these type of streams to 1 -->
    <RemoveLevel RemoveLevel="1"/>

    <!-- Add a private descriptor to the PMT-ES-Info loop -->
    <PmtEsInfoDescriptorComposition>
      <NewDescriptors>
        <Descriptor DescrInsertMode="add" DescrTagScope="private"  PrivDataSpecifier="12345" DescrTag="200"
                    DescrData="0A 0B 0C"/>
      </NewDescriptors>
    </PmtEsInfoDescriptorComposition>

  </ComponentAdaptation>
```

Figure 14. Example of component adaptation

6.6.4.2. Service Wide Component Adaptations (optional)

The service wide component adaptations allow the user to specify adaptations that have to operate on one or all selected component(s) from one single service in the output stream. For example it

allows the user to specify: for the Hungarian audio streams in service 1234 in the output stream, add descriptor XYZ to the ES-info section in the PMT.

The service wide component adaptations are similar to the Transport Stream wide component adaptations with difference that the service wide component adaptations operate on all selected components within a single service.

6.6.4.3. Service Adaptations (optional)

The service adaptations allow the user to specify adaptations that have to operate on a single service. The output is the same service, but with adapted PSI/SI. Service adaptation allows a user to specify:

1. Service-id. The user can specify a (new) service-id for a certain service. Default the MuxXpert allocates a free service-id (see section 6.7.4).
2. Service-type. The user can specify a new service-type. This new value will be used in the SDT, NIT and BAT tables. For newly created services the user should specify a value. Default, the MuxXpert doesn't alter the service-type.
3. PCR-Stream. The user can specify which (new) PCR-Stream must be used by a certain service. In case of a newly created service the user should specify a PCR-Stream. Default, the MuxXpert doesn't alter the PCR-Stream.
4. PMT-PID. Specifying a (new) value for the PMT-PID. Default the MuxXpert allocates a free PID for each PMT (see section 6.7.2).
5. PMT-First-Loop-descriptors. Allows the user to add, remove or to replace PMT-first-loop-descriptors. In case of a newly created service the user should specify these descriptors. Default, the MuxXpert doesn't alter the descriptors, except from the attribute relocations.
6. SDT-Actual-Adaptation. Allows the user to specify fields and descriptors in the SDT-loop related to a certain service. This can be used to change the service name, by specifying a new service-descriptor. In case of a newly created service the user should specify these fields and descriptors. Default, the MuxXpert doesn't alter the fields and descriptors, except from the attribute relocations.

An example of a service adaptation is shown below. The service-id of a selected service is adapted and the EIT-flags related to this service in the SDT-loop are cleared. Further a new service descriptor is specified in the SDT-Loop, specifying a new service-name and provider-name for this service.

```
<!--Specifying a service adaptation-->
<SvcAdaptation>
 <!--The service to be adapted is identified by service_id=13 and LogTsId=101 -->
 <SvcSelector>
  <Short LogTsId="101" SvcId="13"/>
 </SvcSelector>

 <!--New value for the service_id=103-->
 <NewSvcId Assignment="mandatory" SvcId="103"/>

 <!--New values for the EIT-schedule=0 and EIT-present-following=0 flags in the SDT-Loop-->
 <SdtActualAdaptation>
  <NewEitSchedFlag Flag="0"/>
  <NewEitPfFlag Flag="0"/>

  <SdtLoopDescriptorComposition>
   <!-- Replace service descriptor in sdt-->
   <NewDescriptors>
    <Descriptor DescrInsertMode = "replace" DescrTagScope = "public" DescrTag = "72"
                DescrData = "01  0A &quot;MyProvider&quot;  09 &quot;MyProgram&quot;"/>
   </NewDescriptors>
  </SdtLoopDescriptorComposition>

 </SdtActualAdaptation>

</SvcAdaptation>
```

Figure 15. Example of service adaptation

6.6.4.4. PCR Adaptations (optional)

The PCR adaptations allow the user to specify a fixed PID and the RemoveLevel for a certain PCR stream.

Note: After the PCR Adaptations, the service(s) will keep using the same PCR streams however with a possible different PID value. (Use the Service Adaptation section 6.6.4.3, to change the PCR stream within a service.)

The RemoveLevel controls the multiplex behaviour in case the sum of the incoming bit rates exceeds the outgoing bit rate for too long. The RemoveLevel can be used to assign a priority to a stream such that “unimportant” streams are removed first.

Default the PID and the RemoveLevel are determined by the MuxXpert (see section 6.7). However, the user can overrule this default by specifying a new value.

Below an example is shown, a PCR stream from a service is selected for which a new PID and RemoveLevel is specified.

```
<!--PCR Adaptation-->
<PcrAdaptation>

 <!--The PCR to be adapted is part of the service that is identified by service_id=12 and LogTsId=101 -->
 <SvcSelector>
  <Short LogTsId="101" SvcId="12"/>
 </SvcSelector>

 <!--New value for the PID = 123-->
 <NewPid Assignment="mandatory" Pid="123"/>

 <!--New value for the RemoveLevel=1 -->
 <RemoveLevel RemoveLevel ="1"/>

</PcrAdaptation>
```

Figure 16. Example of elementary stream adaptation

6.6.4.5. Transport-Stream Wide Elementary Stream Adaptations (optional)

The elementary stream adaptations allow the user to specify a fixed PID and the RemoveLevel for a certain elementary stream.

The RemoveLevel controls the multiplex behaviour in case the sum of the incoming bit rates exceeds the outgoing bit rate for too long. The RemoveLevel can be used to assign a priority to a stream such that "unimportant" streams are removed first.

Default the PID and the RemoveLevel are determined by the MuxXpert (see section 6.7). However, the user can overrule this default by specifying a new value.

Below an example is shown, an elementary stream from the output stream is selected for which a new PID and RemoveLevel is specified.

```
<!--Elementary Stream Adaptation-->
<EsAdaptation>

 <!--The Elementary Stream to be adapted is identified by network_id=1001, transport_stream_id=1 and PID=1234-->
 <PidSelector>
  <DvbQualified OnwId="1001" TsId="1" Pid="1234"/>
 </PidSelector>

 <!--New value for the PID = 5678-->
 <NewPid Assignment="mandatory" Pid="5678"/>

 <!--New value for the RemoveLevel=1 -->
 <RemoveLevel RemoveLevel ="1"/>

</EsAdaptation>
```

Figure 17. Example of elementary stream adaptation

6.6.4.6. Transport-Stream Wide EMM-Stream Adaptations (optional)

The EMM-Stream Adaptations allow the user to specify a fixed PID and the RemoveLevel for a certain EMM-Stream, similar to the elementary stream adaptation (see section 6.6.4.5).

6.6.4.7. Service Wide ECM-Stream Adaptations (optional)

The ECM-Stream Adaptations allow the user to specify a fixed PID and the RemoveLevel for a certain ECM-Stream, similar to the elementary stream adaptation (see section 6.6.4.5).

6.6.5. Table Assembly

This section describes the specification of *table assembly* (= composition + adaptation) of PSI/SI. The Table Assembly allows the user to specify:

1. Which PSI/SI (sub-) tables will be included in the output
2. how are these tables constructed
3. and what adaptations are applied to the incoming tables

The specification of the table assembly is not the same for all tables. In this section most common options are explained. For the detailed table assembly options see the RMC-Data specification [RMC XSD] and [RMC DOC].

For all tables the user can specify:

1. That the table is not generated at all. (MuxXpert's default).
2. The section data of the Tables as a sequence of "raw" data bytes.
3. The table cycle properties (cycle rate, priority, etc.) See [RMC DOC]. Default the MuxXpert will use the DVB-recommended cycle-rates.
4. Descriptor sorting. Specifying the descriptor sort order for each descriptor-tag scope. Default, the MuxXpert will not change the descriptor order.

   Descriptor sorting is not applied to tables which section data is specified directly (as in 2.).

For the PAT, CAT, PMT, SDT-Actual, EIT-Present-Following-Actual and EIT-Schedule-Actual tables the user can specify:

5. That the table is automatically generated from the output stream composition and the output stream adaptation specification (See section 6.6.3 and section 6.6.4). The PSI/SI information is taken from the incoming streams and the specified compositions and adaptations are applied.

For the NIT-Actual, NIT-Other, BAT, SDT-Other, EIT-Present-Following-Other, EIT-Schedule-Other and TOT tables the user can specify:

6. The table assembly instructions according to which the table has to be created. For example the user can specify:
   a. That the generated (sub-) table is derived from an incoming (sub-) table (e.g. NIT-Other -> NIT-Actual or NIT-Actual -> NIT-Other conversion)
   b. The fields and descriptors to build up a (sub-) table.
   c. Or by combinations of a. and b.

For the TDT table the user can specify:

7. That the TDT table must be generated and inserted.

The following pages will show some examples of PSI and SI table assembly specifications.

Below an example of the specification of the PAT, CAT and PMT table assembly. Typically these tables are automatically generated from the output stream composition and the output stream adaptation specification. In this example PAT and PMT are derived from the composition, no CAT is generated, default cycle-rates are used and no descriptor sorting is applied.

```
<!-- Specification of the MPEG PSI table assembly -->
<MpegTables>

 <Pat>
  <FromTsComposition/>
 </Pat>

 <Cat>
  <TableNotGenerated/>
 </Cat>

 <Pmt>
  <FromTsComposition/>
 </Pmt>

</MpegTables>
```

Figure 18. Example of PSI table (PAT, CAT and PMT) assembly

Below an example of the specification of a custom table assembly is given. The section data of the custom table is specified as a sequence of "raw" data bytes. A new cycle-rate of the custom table is specified. Multiple custom tables with different table_ids can be specified.

```
<!-- -.-.-.-.-.-. specification of a Custom Table (RST table table_id 113) -.-.-.-.-.-.-. -->
<Custom TableId="113">
 <ExternallySuppliedTable>
  <ExternalTableProperties Pid="19" VersionMgmtFlag="false" AddCrcFlag="false"
                           SyntaxIndicator="0" PrivateIndicator="1" />
  <SubTables>
   <SubTable>
    <!-- SectionData; without first 3 bytes (table_id, section_syntax_indicator and section_length) -->
    <Section SectionData="0001 1001 0382 0006 FC
                          0001 1001 0383 0006 FC
                          0001 1001 0384 0007 FC">
    </Section>
   </SubTable>
  </SubTables>
 </ExternallySuppliedTable>
 <!-- -.-.-.-.-.-.-.-. specification of the Custom Table cycle properties -.-.-.-.-.-.-.-. -->
 <TableCycleProperties MinimumCycleRate="9" TargetCycleRate="11"/>
</Custom>
```

Figure 19. Example of a custom table assembly

Below a typical example of the specification of the SDT-Actual is given. Typically the SDT-Actual is derived from the composition and adaptation specification. The default cycle-rates are used and no descriptor sorting is applied.

```
<!-- -.-.-.-.-.-.-. specification of the SDT actual table -.-.-.-.-.-.-.-.-. -->
<SdtActual>
  <FromTsComposition/>
</SdtActual>
```

Figure 20. Example of SDT-Actual table assembly

Below another example of the specification of the SDT-Actual is given. Here the raw table data of the SDT is specified.

```
<!-- -.-.-.-.-.-.-. specification of the SDT actual table -.-.-.-.-.-.-.-.-. -->
<SdtActual>
  <ExternallySuppliedTable>
    <ExternalTableProperties VersionMgmtFlag="false" AddCrcFlag="true" PrivateIndicator="0"/>
    <SubTables>
      <SubTable>
        <!--  SectionData; without first 3 bytes (table_id, section_syntax_indicator and section_length) and without CRC -->
        <Section SectionData="0457 71 00 00 0001 FF
              0001 F3 0015 48 13 01 07 &amp;quot;TV Prov&amp;quot; 09 &amp;quot;Program 1&amp;quot;
              0002 F3 0015 48 13 01 07 &amp;quot;TV Prov&amp;quot; 09 &amp;quot;Program 2&amp;quot;
              0003 F3 0015 48 13 01 07 &amp;quot;TV Prov&amp;quot; 09 &amp;quot;Program 3&amp;quot;
              0004 F3 0015 48 13 01 07 &amp;quot;TV Prov&amp;quot; 09 &amp;quot;Program 4&amp;quot;">
        </Section>
      </SubTable>
    </SubTables>
  </ExternallySuppliedTable>
</SdtActual>
```

Figure 21. Example of SDT-Actual table assembly by using ExternallySuppliedTable

Below a typical example of the specification of a NIT-Actual table assembly is shown. The NIT-Actual table is built from scratch by specifying the descriptors in the NIT's first loop and thereafter specifying the fields of the NIT's second loop (network_id and transport_stream_id) and the related descriptors.

Note that for the Transport Streams that are created by the MuxXpert (NativeNitTss) the user can refer to an output stream and the MuxXpert can automatically generate a service_list_descriptor.

```
<!-- -.-.-.-.-.-. specification of the NIT actual table from scratch -.-.-.-.-.-.-. -->
<NitActual>

 <AssembledTable>

  <NitActualFromScratch>

   <!-- Specification of the NIT first loop descriptors -->
   <NitFirstLoopDescriptors>
    <!-- Addition of the network_name_descriptor -->
    <Descriptor DescrInsertMode = "add" DescrTagScope = "public" DescrTag = "64"
                DescrData = "&amp;quot;MyNetworks&amp;quot;"/>
    <!-- Add possible additional NIT first loop descriptors here -->
   </NitFirstLoopDescriptors>


   <!-- Specification of NIT second loop -->

   <!-- Specification of the "elsewhere" created Transport Streams in this network -->
   <ForeignNitTss>
    <!-- none -->
   </ForeignNitTss>

   <!-- Specification of the "here" created Transport Streams in this network.
        create the  service_list_descriptors automatically -->
   <NativeNitTss AutoGenerateSvcListDescriptorFlag = "true">

    <NativeNitTsDescription>

     <!-- Specify the native TSS -->
     <TsSelector>
      <ByLogicalTsId LogTsId = "901"/>
     </TsSelector>

     <!-- Additional NIT 2nd loop descriptors for this TS-->
     <TsDescriptors>
      <!-- Addition of  the cable_delivery_system_descriptor-->
      <Descriptor DescrInsertMode = "add" DescrTagScope = "public" DescrTag = "68"
                  DescrData = "04470000 FFF2 03 0052740 F"/>

      <!-- Addition of  the logical_channel_descriptor-->
      <Descriptor DescrInsertMode = "add" DescrTagScope = "private" PrivDataSpecifier = "40" DescrTag = "131"
                  DescrData = "6DCE FC15 6DCF FC16 0383 FC17 0384 FC18" />
     </TsDescriptors>

    </NativeNitTsDescription>

   </NativeNitTss>

  </NitActualFromScratch>

 </AssembledTable>

</NitActual>
```

Figure 22. Example of NIT-Actual table assembly from scratch

Below another example of the specification of a NIT-Actual table assembly is shown. The NIT-Actual table is built from an incoming NIT-Actual by specifying incoming Transport Stream from which the NIT-Actual is taken. Thereafter the original NIT's second loop information related to the incoming Transport Stream is removed and new NIT second loop information is added. This way of specifying a NIT-Actual can be useful in case only the modulation type of a Transport Stream is changed and therefore a small NIT adaptation is needed.

```
<!-- -.-.-.-.-.-. specification of the NIT actual table from incoming NIT actual -.-.-.-.-.-.-. -->
<NitActual>

 <AssembledTable>

  <!-- NIT actual is derived from an incoming NIT actual-->
  <NitActualFromNitActual>
   <!-- NIT actual from input 201 is used-->
   <TsInSelector>
    <ByLogicalTsId LogTsId="201"/>
   </TsInSelector>

   <!-- No modifications to NIT 1st loop-->

   <!-- Remove the original NIT 2nd loop related to the incoming Transport Stream -->
   <SecondLoopTsSubsetSelector SelectionMethod="negative">
    <TsSelector>
     <ByLogicalTsId LogTsId="201"/>
    </TsSelector>
   </SecondLoopTsSubsetSelector>

   <!-- No modification to the other NIT 2nd loops -->


   <!-- Add the "here" created  Transport Stream's NIT 2nd loop information.
        Create the  service_list_descriptors automatically -->
   <NativeNitTss AutoGenerateSvcListDescriptorFlag = "true">

    <NativeNitTsDescription>

     <!-- Specify the native TS -->
     <TsSelector>
      <ByLogicalTsId LogTsId = "901"/>
     </TsSelector>

     <!-- Additional NIT 2nd loop descriptors for this TS-->
     <TsDescriptors>
      <!-- Addition of  the cable_delivery_system_descriptor-->
      <Descriptor DescrInsertMode = "add" DescrTagScope = "public" DescrTag = "68"
                  DescrData = "04470000 FFF2 03 0052740 F"/>
     </TsDescriptors>

    </NativeNitTsDescription>

   </NativeNitTss>
  </NitActualFromNitActual>
 </AssembledTable>

</NitActual>
```

Figure 23. Example of NIT-Actual table assembly from an incoming NIT-Actual

Below a typical example of the specification of a SDT-Other table assembly is shown. The SDT-Other table is generated by converting a SDT-Actual Tables from the input and output stream into SDT-Other sub-tables. Note that the SDT-Other assembly may also include sub-tables generated from scratch and from SDT-Others. A new cycle rate is specified for this table and a descriptor order is specified.

```
<!-- -.-.-.-.-.-. specification of the SDT other table -.-.-.-.-.-.-.-. -->
<SdtOther>

  <AssembledTable>

    <!-- SDT Actual on input 101 becomes an SDT other subtable-->
    <SdtOtherSubTable>
      <SdtOtherFromSdtActual>
        <TsInOutSelector>
          <ByLogicalTsId LogTsId="101"/>
        </TsInOutSelector>
      </SdtOtherFromSdtActual>
    </SdtOtherSubTable>

    <!-- SDT Actual on output 902 becomes an SDT other subtable-->
    <SdtOtherSubTable>
      <SdtOtherFromSdtActual>
        <TsInOutSelector>
          <ByLogicalTsId LogTsId="902"/>
        </TsInOutSelector>
      </SdtOtherFromSdtActual>
    </SdtOtherSubTable>

  </AssembledTable>

  <!-- New cycle rates. The target rate is once per 10 seconds-->
  <TableCycleProperties TargetCycleRate="0.1" MinimumCycleRate="0.09"/>

  <!-- Apply descriptor sorting; first de service_descriptor thereafter the data_broadcast_descriptor  -->
  <TableDescriptorSorting>
    <ScopeDescrSorting DescrTagScope="public">
      <Tag DescrTag = "72"/>
      <Tag DescrTag = "100"/>
    </ScopeDescrSorting>
  </TableDescriptorSorting>

</SdtOther>
```

Figure 24. Example of SDT-Other table assembly

Below a typical example of the specification of the EIT-Present-Following-Actual and the EIT-Schedule-Actual table assembly is shown. In this example the EITs are derived from the composition and adaptation specification. The EIT-events related to the services in the outgoing Transport Stream are taken from the EIT information in the incoming Transport Streams.

```
<!-- -.-.-.-.-.-. specification of the EIT P/F actual table -.-.-.-.-.-.-.-. -->
<EitPfActual>
  <AssembledTable>
    <DefaultEitActualSubTable>
      <FromComposition/>
    </DefaultEitActualSubTable>
  </AssembledTable>
</EitPfActual>

<!-- -.-.-.-.-.-. specification of the EIT Schedule actual table -.-.-.-.-.-.-.-. -->
<EitSchedActual>
  <AssembledTable>
    <DefaultEitActualSubTable>
      <FromComposition/>
    </DefaultEitActualSubTable>
  </AssembledTable>
</EitSchedActual>
```

Figure 25. Example of EIT-Actual table assembly

Note: The MuxXpert deletes old EIT-events. In case of recording from the past it is often desired to transfer these "old" EIT-event into "new" EIT-events. This can be achieved by specifying a time correction, see section 6.5.

Below another example of the specification of the EIT-Present-Following-Actual assembly is shown. In this example the EIT-Present-Following-Actual sub-tables are derived from the composition and adaptation specification except from one sub-table. For this sub-table that is related to a certain service, a new EIT-P/F-Actual sub-table is specified.

```
<!-- -.-.-.-.-.-. specification of the EIT P/F actual table -.-.-.-.-.-.-. -->
 <EitPfActual>
  <AssembledTable>

   <!-- Specification of the EIT P/F subtable related to ServiceId 4164-->
   <EitActualSubTable>
    <EitActualFromScratch>
     <SvcSelector>
      <Short LogTsId="201" SvcId="4164"/>
     </SvcSelector>

     <!-- Specification of present and following event -->
     <EventDescriptions>

      <!-- Present event for the same service -->
      <Event>
       <EventPars  EventId="0806" EventType="present" StartTime="2007-06-12T12:00:00" Duration="PT30M"
                   RunningStatus="1" FreeCaMode="0"/>
       <EventDescriptors>
        <Descriptor DescrInsertMode="add" DescrTagScope="public" DescrTag="77"
                    DescrData="&quot;eng&quot; 04 &quot;News&quot; 12 &quot;Todays news events&quot;"/>
       </EventDescriptors>
      </Event>

      <!-- Following event for the same service -->
      <Event>
       <EventPars EventId="0807" EventType="following" StartTime="2007-06-12T12:30:00" Duration="PT1H45M"
                  RunningStatus="1" FreeCaMode="0"/>
       <EventDescriptors>
        <Descriptor DescrInsertMode="add" DescrTagScope="public" DescrTag="77"
                    DescrData="&quot;eng&quot; 06 &quot;Sports&quot; 13 &quot;Live football match&quot;"/>
       </EventDescriptors>
      </Event>

     </EventDescriptions>
    </EitActualFromScratch>
   </EitActualSubTable>

   <!-- Default the EIT P/F subtables are derived from the composition-->
   <DefaultEitActualSubTable>
    <FromComposition/>
   </DefaultEitActualSubTable>

  </AssembledTable>
 </EitPfActual>
```

Figure 26. Example of EIT-Present-Following-Actual table assembly

Below a typical example of the specification of the TDT and TOT table assembly is shown. In this example TDT is generated. The TOT is created and the descriptors are taken from an incoming TOT the local time offset descriptor is replaced.

```
<!-- -.-.-.-.-.-.-.-. specification of the TDT actual table -.-.-.-.-.-.-.-. -->
<Tdt>
  <AssembledTable>
    <TdtFromScratch/>
  </AssembledTable>
</Tdt>

<!-- -.-.-.-.-.-.-.-. specification of the TOT actual table -.-.-.-.-.-.-.-. -->
<Tot>
  <AssembledTable>
    <!-- Get the TOT descriptors from the TOT on input 101 -->
    <TotFromTot>
      <TsInSelector>
        <ByLogicalTsId LogTsId = "101"/>
      </TsInSelector>

      <TotDescriptors>
        <!-- Replace the local_time_offset_descriptor-->
        <Descriptor DescrInsertMode="replace" DescrTagScope="public" DescrTag ="88"
                    DescrData ="&quot;nld&quot; 02 0200 D481020000 0100
                                &quot;deu&quot; 02 0200 D481020000 0100"/>
      </TotDescriptors>

    </TotFromTot>
  </AssembledTable>
</Tot>
```

Figure 27. Example of TDT and TOT table assembly

6.7. Allocation Strategies

6.7.1. Assignment of RemoveLevels

The MuxXpert supports a graceful-degradation algorithm to handle bit rate overflows. To control this algorithm, the MuxXpert sets the value of the RemoveLevel attribute, for each PID in each outgoing Transport Stream.

The RemoveLevel controls the multiplex behaviour in case the sum of the incoming bit rates exceeds the outgoing bit rate for too long. The RemoveLevel can be used to assign a priority to a stream such that "unimportant" streams are removed first. Streams with `RemoveLevel=1` are removed first, streams with `RemoveLevel=4` last and streams whit `RemoveLevel=0` should not be removed at all.

The Remux Controller derives the value of `RemoveLevel` from the characteristics of the elementary stream, according to the table below. The default remove level can be overruled through the Remux Configuration Data.

Table 1. Default value of `RemoveLevel`, as a function of elementary stream characteristics.

Elementary-Stream Characteristics	*RemoveLevel*
Elementary stream contains time stamps synchronised to PCR (stream carries DTS and/or PTS), or elementary stream carries PCRs	0
Elementary does not contain time stamps (data stream)	4
PSI (PAT, PMT, CAT)	2
SI (NIT, SDT, ...)	1
Custom table	1
Unreferenced PID	4

6.7.2. Allocation of PIDs

The MuxXpert assigns a PID to each elementary stream that is included in the output stream.

PID allocation strategy of the MuxXpert is defined below as a series of rules that are executed sequentially. Once a certain rule applies, a PID value is assigned and the PID cannot be overruled by subsequent rules.

1. Assignment of PIDs for which the user has specified a *mandatory* PID assignment. These assignments will overrule user reserved PIDs.
2. If the KeepOrigPid option is set, output PIDs will be fixed (equal) to the incoming PID. These assignments will not overrule user reserved PIDs.
3. Assignment of unreferenced PIDs, their output PID will be fixed (equal) to the incoming PID. These assignments will overrule user reserved PIDs.
4. Assignment of PIDs of PSI/SI tables for which DVB or MPEG has defined mandatory PIDs. These assignments will overrule user reserved PIDs.
5. Keep PIDs constant. When the configuration changes or input stream changes, the MuxXpert assigns for each elementary stream the same PID as before the change. This only succeeds if the PID is not yet assigned and is not a user reserved PID.

6. Assignment of PIDs for which the user has specified a preferred PID assignment. This only succeeds if the PID is not yet assigned and is not a user reserved PID.
7. Assignment of PIDs that have a preferred PID assignment that could not be honoured (PID in use); the MuxXpert assigns the first unassigned/unreserved PID value (with wrap-around to 0) above the specified *preferred* value.
8. Assignment of PIDs of remaining elementary streams. For each elementary stream that did not get a PID assigned through one of the other PID-allocation rules the MuxXpert assigns a PID value. If the KeepOrigPid option is set the MuxXpert assigns the first unassigned/unreserved PID value (with wrap-around to 0) above the specified incoming PID value. If the KeepOrigPid option is not set the MuxXpert assigns the first unassigned/unreserved PID value starting at zero.

In case the PID of an elementary stream changes, implies that the MuxXpert updates all references to that elementary stream.

6.7.3. Allocation of Component-tags

The *Component-tag* (or: Stream Identifier) is used to establish a link between PMT and EIT. Component-tag is a field in the stream-identifier descriptor.

The scope for component-tag allocation is a service. Different services may use the same set of component-tag values. This is also true for shared components. A shared component may have a different component-tag value for each service in which it is included.

The rules for component-tag allocation are similar to PID-allocation rules. The following rule set is applied sequentially.

1. Assignment of component-tags for which the user has specified a *mandatory* Component-Tag assignment.
2. Assignment of component-tags for which the user has specified *preferred* Component-Tag assignment. This only succeeds if the component-tag is not yet assigned.
3. Keep incoming component-tags untouched. The MuxXpert assigns to each component the same component-tag value as it had in the incoming stream. This only succeeds if the component-tag is not yet assigned.
4. Assignment of component-tags that have a preferred assignment that could not be honoured (tag in use), the MuxXpert assigns the first unassigned tag value (with wrap-around to 0) above the specified *preferred* value.
5. For incoming components that cannot keep their original component-tag, (tag in use), the MuxXpert assigns the first unassigned tag value (with wrap-around to 0) above the original value.
6. Assignment of component-tags of remaining components. For each component that did not get a tag assigned through one of the other rules, the MuxXpert assigns the first tag that is still unassigned, starting at zero.

Changing the component-tag of a re-multiplexed component implies that the MuxXpert updates all references to the component.

6.7.4. Allocation of Service-ids

The *Service-identifier* uniquely identifies a service within a Transport Stream.

The rules for `service-id` allocation are similar to PID- and component-tag allocation rules. The following rule set is applied sequentially.

1. Assignment of service-ids for which the user has specified a *mandatory* service-id assignment.
2. Assignment of service-ids for which the user has specified a *preferred* service-id assignment. This only succeeds if the service-id is not yet assigned.
3. Keep incoming service-ids untouched. The MuxXpert assigns to each service the same service-id value as it had in the incoming stream. This only succeeds if the service-id is not yet assigned.
4. Allocation of remaining services. For each service without service-id assignment, and for each incoming service that has a service-id conflicting with another service-id in the same output stream, the MuxXpert assigns a service-id value that is still unassigned, starting at the lowest unallocated value.

Changing the service-id of a re-multiplexed service implies that the MuxXpert updates all references to the service.

Appendix A: Play-list XML Syntax

The sequence of files to be played out by the file-player can be specified in a play-list. The play-list is a XML-formatted file. This appendix specifies the syntax of the XML-file.

The top-level element in a play-list file is the PlayList element.

PlayList

Root-element of a play-list document. The PlayList contains the following child elements:

1. PlaySettings; Optional element that specifies the settings of the file-player.
2. A sequence of PlayListItem elements, each element specifies one item in the play-list.

Below the graphical diagram of the PlayList element is shown.

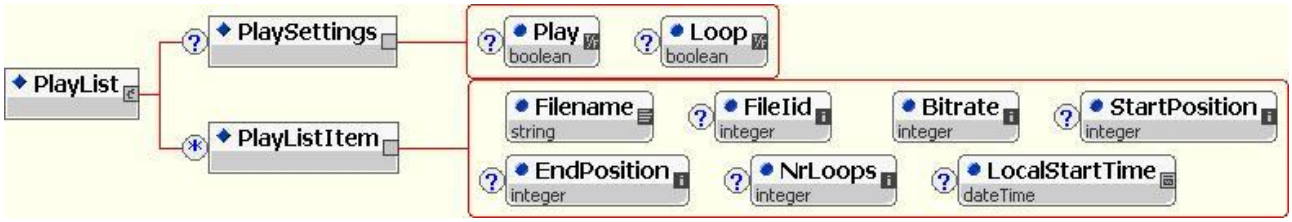


Figure 28. Graphical diagram of PlayList.

PlayList/PlaySettings

Optional element, used for specifying the state of the play-out operation after loading the play-list. It has the following attributes:

Play

Type: boolean, Use: optional, Default: true (=play)
Specifies whether the file-player starts or continues the play-out operation.

Loop

Type: boolean, Use: optional, Default: false (=not looping)
Specifies whether the file-player starts play-out of a file to occur in a continuous loop.

LoopList

Type: boolean, Use: optional, Default: false (=not looping)
Specifies whether the file-player starts play-out of the playlist to occur in a loop.

PlayList/PlayListItem

Element that can occur multiple times, the PlayListItem element is used for specifying one item in the play-list. It has the following attributes:

Filename

Type: string, Use: required
Specifies the path of the file to be played-out. The path must start with a drive-letter.

FileIid

Type: integer, Use: optional
The FileIid uniquely identifies an instance of a file (play-list-item) within the play-list and within two subsequent loaded versions of the play-list. The file-iids used in a new play-list determine the behaviour of the file-player. There are three possible cases:

a. The file-iid of the currently playing file is still in the new play-list: the file-player will continue the play-out operation of the current file without interruption.
b. The file-player finished the previous play-list and the file-iid of the lastly played-out file is still in the new play-list: the file-player will start playing-out the file coming after the lastly played-out file.
c. Other situations: the file-player will start playing-out the first file in the new play-list.

Specifying the file-iid, enables the user to extend the play-list with "new" play-list-items and to remove the "old" play-list-items, without interrupting the current play-out operation.

Note(s):
In two subsequent versions of the play-list the user should not re-use the same file-iid for another instance of a file.
The user must specify a new file-iid in case one of the play-list-item-attributes of the file changes (a new play-list-item is created).

Bitrate

Type: integer, Use: required
Specifies the bit-rate used for playing-out the file. In case VarTsRate is set to "true" Bitrate specifies the peak bit-rate.

VarTsRate

Type: boolean, Use: optional, Default: false (=constant rate)
Specifies whether the file is contains a variable rate single program Transport Stream.

StartPosition

Type: integer, Use: optional, Default: 0 (= begin of file)
Specifies a byte offset from the start of the file at which to start the file play-out.

EndPosition

Type: integer, Use: optional, Default: 0 (= end of file)
Specifies a byte offset from the start of the file at which to stop the file play-out. If the EndPosition is set to "0" the file is played-out till the end of the file.

NrLoops

Type: integer, Use: optional, Default: 0
Specifies the number of times the play-out of the file is repeated. If the NrLoops is set to "-1" this file is repeated until the play-out start-time of the next file in the play-list.

LocalStartTime

Type: dateTime, Use: optional, Default: As Soon As Possible.
Specifies the play-out start-time of this file. Subsequent play-list-items that specify a start-time must have increasing start-times. The format of the LocalStartTime attribute is based on ISO 8601, using local time. If this attribute is absent, the file will be played-out as soon as possible after the previous file in the list is completed.

Below an example of a play-list is given. The play-out is started without looping; two play-list-items are specified.

```
<PlayList>
  <PlaySettings Play="true" Loop="false" LoopList="false" />
  <PlayListItem Filename="C:\Data\Local Streams\torino2006_1.TS" FileIid="123" Bitrate="14929412" />
  <PlayListItem Filename="C:\Data\Local Streams\torino2006_2.TS" FileIid="124" Bitrate="14929412"
                StartPosition="0" EndPosition="465493828" NrLoops="0" LocalStartTime="2006-10-18T13:02:18" />
</PlayList>
```

Figure 29. Example of a play-list.