TSDuck

MPEG Transport Stream Toolkit
User's Guide

Version 3.19-1372
August 2019

License

Third-party libraries

TSDuck includes a few third-party libraries, either in source form, binary form or both. For more details about the licenses of these third-party libraries, see the file named LICENSE.txt in the source code repository or in the installed files.

DTAPI: On Linux and Windows, the TSDuck binary distributions contain the DTAPI library. This software is available in binary form only (see [23]). *Copyright © 2017 by Dektec Digital Video B.V.*

LibTomCrypt: Some source code was directly copied and adapted from LibTomCrypt into TSDuck (see [30]). *LibTomCrypt is public domain. As should all quality software be. Tom St Denis*.

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

AES	Advanced Encryption Standard
AIT	Application Information Table
ASI	Asynchronous Serial Interface
ATIS	Alliance for Telecommunications Industry Solutions
ATR	Answer To Reset (smartcard)
AVC	Advanced Video Coding
BDA	Broadcast Device Architecture (Microsoft Windows)
BDT	Binary Data Table
CA	Conditional Access
CAS	Conditional Access System
CAT	Conditional Access Table
CMT	CA Message Table
CP	Crypto-Period
CSA	*Conseil Supérieur de l'Audiovisuel* (French national regulator for TV)
CW	Control Word
DKMS	Dynamic Kernel Module Support (Linux)
DRM	Digital Rights Management
DTS	Decoding Time Stamp
DTTV	Digital Terrestrial Television
DTV	Digital Television
DVB	Digital Video Broadcasting
DVB-C	DVB Cable modulation
DVB-C2	DVB Cable modulation, 2nd generation
DVB-CISSA	DVB Common IPTV Software-oriented Scrambling Algorithm
DVB-CSA	DVB Common Scrambling Algorithm
DVB-S	DVB Satellite modulation
DVB-S2	DVB Satellite modulation, 2nd generation
DVB-T	DVB Terrestrial modulation
DVB-T2	DVB Terrestrial modulation, 2nd generation
EIS	Event Information Scheduler
ECM	Entitlement Control Message
ECMG	ECM Generator
EMM	Entitlement Management Message
EMMG	EMM Generator
ES	Elementary Stream
ETSI	European Telecommunications Standards Institute
FIPS	Federal Information Processing Standard
HbbTV	Hybrid broadcast/broadband Television
IDSA	IIF Default Scrambling Algorithm
IIF	IP-TV Interoperability Forum
INT	IP/MAC Notification Table
IP	Internet Protocol
ISO	International Standardization Organization
IV	Initialization Vector
MPE	Multi-Protocol Encapsulation
MPEG	Moving Picture Experts Group
MUX	Multiplexer
NIST	National Institute of Standards and Technology
NIT	Network Information Table
OUI	Organizationally Unique Identifier (IEEE assigned)
PAT	Program Association Table
PCR	Program Clock Reference
PES	Packetized Elementary Stream
PID	Packet Identifier
PLP	Physical Layer Pipe
PMT	Program Map Table

PSI	Program Specific Information
PTS	Presentation Time Stamp
RTP	Real-Time Protocol
SCS	SimulCrypt Synchronizer
SDT	Service Description Table
SI	Service Information
STB	Set-Top Box
T2-MI	DVB-T2 Modulator Interface
TDT	Time and Date Table
TID	Table Identifier
TNT	*Télévision Numérique Terrestre* (French DTTV network)
TOT	Time Offset Table
TPS	Transmission Parameter Signalling
TS	Transport Stream
UDP	User Datagram Protocol
UNT	Update Notification Table

References

[1] ISO/IEC 13818-1:2000 | ITU-T Recommendation H.262 (2000), Second edition, December 2000: “Generic coding of moving pictures and associated audio information: Systems” (also known as “MPEG-2 System Layer”)

[2] ISO/IEC JTC1/SC29/WG11 N5771, July 2003: “Generic coding of moving pictures audio: systems. Amendment 3: Transport of AVC video data over ITU-T Rec H.222.0 | ISO/IEC 13818-1 streams” (in short: “transport of MPEG-4 AVC video over MPEG-2 TS”)

[3] ISO/IEC 13818-6, July 1998: “Digital Storage Media Command & Control” (DSM-CC)

[4] ETSI, ETR 289, October 1996: “Digital Video Broadcasting (DVB); Support for use of scrambling and Conditional Access (CA) within digital broadcasting systems”.

[5] ETSI TS 103 197, V1.4.1, September 2004: “Digital Video Broadcasting (DVB); Head-end implementation of DVB SimulCrypt”.

[6] ETSI EN 300 468, V1.15.1, March 2016: “Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems”.

[7] ETSI TR 101 211 V1.6.1, May 2004: “Digital Video Broadcasting (DVB); Guidelines on implementation and usage of Service Information (SI)”.

[8] ETSI TR 101 162, V1.2.2, May 2003: “Digital Video Broadcasting (DVB); Allocation of Service Information (SI) and data broadcasting codes for DVB systems”.

[9] ETSI TS 102 006, V1.3.1, May 2005: “Digital Video Broadcasting (DVB); Specification for System Software Update in DVB Systems”.

[10] ETSI TS 102 773, V1.2.1, December 2010, “Modulator Interface (T2-MI) for a second generation digital terrestrial television broadcasting system (DVB-T2)”.

[11] ETSI EN 302 755, V1.4.1, July 2015, “Frame structure channel coding and modulation for a second generation digital terrestrial television broadcasting system (DVB-T2)”.

[12] ETSI TS 102 809, V1.3.1, June 2017: “Digital Video Broadcasting (DVB); Signalling and carriage of interactive applications and services in Hybrid broadcast/broadband environments” (HbbTV).

[13] ETSI TS 102 006, V1.4.1, June 2015: “Digital Video Broadcasting (DVB); Specification for System Software Update in DVB Systems”.

[14] ETSI EN 301 192, V1.6.1, August 2015: “Digital Video Broadcasting (DVB); DVB specification for data broadcasting”.

[15] ETSI TS 101 812, V1.3.2, August 2006: "Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.0.3".

[16] ETSI TS 103 127, V1.1.1, May 2013: “Digital Video Broadcasting (DVB); Content Scrambling Algorithms for DVB-IPTV Services using MPEG2 Transport Streams”.

[17] ANSI/SCTE 35 2017, “Digital Program Insertion Cueing Message for Cable”.

[18] ATSC A65/2013, August 2013: "ATSC Standard: Program and System Information Protocol for Terrestrial Broadcast and Cable".

[19] ATSC A69/2009, December 2009: "ATSC Recommended Practice: Program and System Information Protocol Implementation Guidelines for Broadcasters".

[20] EACEM TR 030, V1.0, February 2000: "Baseline Digital Terrestrial TV Receiver Specification".

[21] Via Eutelsat Fransat: "Set-Top-Box Specification DVB MPEG-4 HD", V0.0.7, October 2009.

[22] Dektec Digital Video B.V. corporate home page, http://www.dektec.com/

[23] Dektec drivers and SDK’s downloads, http://www.dektec.com/downloads/SDK/

[24] Linux DKMS for Dektec device drivers, https://github.com/tsduck/dektec-dkms

[25] HiDes USB DVB-T modulator adaptors, http://www.hides.com.tw/product_cg74469_eng.html

[26] Device drivers for HiDes modulators, https://github.com/tsduck/hides-drivers/

[27] Linux TV Wiki: “How to install DVB device drivers”, http://linuxtv.org/wiki/index.php/How_to_install_DVB_device_drivers

[28] VideoLAN VLC Media Player home page, http://www.videolan.org/vlc/

[29] Telxcc project page, https://github.com/petrkutalek/telxcc

[30] LibTomCrypt site, http://www.libtom.net/LibTomCrypt/

[31] TSDuck Web site, https://tsduck.io/

[32] Homebrew tap for TSDuck on macOS, https://github.com/tsduck/homebrew-tsduck

[33] TSDuck issues tracker, https://github.com/tsduck/tsduck/issues

1 Transport Stream Toolkit Overview

1.1 Purpose

The transport stream toolkit contains a set of simple but flexible command-line utilities that run on Linux, Windows and macOS. These commands are described in this document.

Through *tsp*, the *transport stream processor*, many types of analysis and transformation can be applied on live or recorded transport streams. This utility can be extended through *plugins*. Existing plugins can be enhanced and new plugins can be developed using a library of C++ classes.

Structure of this guide:

- The chapter 2 describes the data formats (transport stream, binary sections files, XML files).
- The chapter 3 describes all TSDuck utilities.
- The chapter 4 describes all *tsp* plugins.
- The chapter 5 provides some concrete examples of TSDuck usage.
- The chapter 6 describes the level of test and support for some hardware devices, mainly DVB receivers and Dektec devices.

1.2 Operating System Selection Guidelines

Here is a brief summary of pros and cons of using TSDuck on the various operating systems.

- Linux pros:
  - ⇒ Availability of a powerful shell environment. TSDuck is a light-weight *toolkit* with elementary tools and plugins which can be combined in an infinite number of ways. The user can obtain even more flexibility when combining them with the *bash* shell and all standard UNIX utilities (*grep*, *sed*, *awk*, etc.) See some complex examples in section 5.2.
- Linux cons:
  - ⇒ When used in a mobile environment, a laptop PC with Linux (or Linux/Windows dual boot) is required.
  - ⇒ Some DVB tuners are not supported on Linux. Some supported tuners do not work well on Linux. Make sure to get fully supported DVB hardware.
- Windows pros:
  - ⇒ Available on all “average user” laptop PC. Useful for transport stream capture and analysis in the field.
- Windows cons:
  - ⇒ No or limited shell environment.
  - ⇒ Some limitations in the support of DVB receiver devices (see 6.1.2.2, page 284, for more details):
    - o No standard support for DiSEqC with DVB-S/S2 tuners, which makes Windows useless when capturing behind a DiSEqC switch with multiple dishes.
    - o Impossible to retrieve the actual tuning parameters of a transport stream as detected by the tuner device.
- macOS pros:
  - ⇒ Availability of a powerful shell environment, just another UNIX system, just like Linux. Powerful user-friendly system.
- macOS cons:
  - ⇒ Currently no support for hardware DVB tuners and Dektec devices. So, macOS is recommended only when dealing with transport stream files and IP networking, not for any hardware support.

Summary: Use Linux if you can. Use Windows when you do not have Linux (typically a Windows laptop in the field). Use macOS if you have a Mac and do not need DVB or Dektec hardware.

1.3 Installing TSDuck

The TSDuck installers are available from the "Download" section of the TSDuck Web site (see [31]).

The basic installation provides all TSDuck tools and plugins. The command-line tools are directly accessible from the command prompt.

TSDuck can also be used as a large C++ library for third-party applications, outside the TSDuck tools and plugins. To do that, you must install the "TSDuck development environment". See more details on the TSDuck Web site: select "Source code", then "Doxygen documentation" and finally "Using the TSDuck library".

Windows

Binary executable installers are provided for Windows platforms, 32-bit and 64-bit versions.

The directory containing the command-line tools is automatically added to the Path. The TSDuck development environment is included in the installer but it is not installed by default. You must select it explicitly.

Note that TSDuck is supported for Windows 7 and higher only. TSDuck may work on Windows XP or Vista but without guarantee or support.

For users without privilege, *portable packages* are provided for 32 and 64 bits platforms. A portable package is simply a zip archive file which can be expanded anywhere. The TSDuck commands are located in the `bin` subdirectory and can be executed from here without any additional setup. It is probably a good idea to add this `bin` directory in the `Path` environment variable of the user.

Linux

Two flavors of packages are available: `.rpm` for Fedora systems and `.deb` for Ubuntu systems. Currently, only 64-bit packages are available.

All tools are in `/usr/bin`. There is a separate package for the TSDuck development environment.

MacOS

On macOS, TSDuck is installed using the Homebrew packaging and delivery system (see [32]).

All tools are accessible from `/usr/local/bin` (standard installation structure for Homebrew).

The development environment is always installed with TSDuck using Homebrew.

2 Data Formats

2.1 Transport Stream Format

Transport streams shall conform to the MPEG-2 system layer format as defined in ISO 13818-1 [1].

2.1.1 Live transport streams

Live transport streams can be read by TSDuck from:

- Live sources using specialized hardware, cheap DVB tuners or Dektec ASI devices.
- UDP/IP using various encapsulations (the encapsulation of TS packets in UDP packets does not matter since TSDuck automatically retrieves the TS packets inside UDP packets and simply ignores everything in between).
- HTTP or HTTPS streams without encapsulation (ie. raw TS streams, but not manifest-based formats such as DASH or HLS).
- HLS (HTTP Live Streaming) with transport stream segments (not fMP4).

See the documentation of the plugins *dvb*, *dektec*, *ip*, *http*, *hls* for more details on the reception of live transport streams.

The same plugins can also transmit live streams on Dektec ASI and modulator devices and on UDP/IP streams (multicast or unicast).

2.1.2 Stored transport streams

Transport streams can be read from and written to binary files, called *TS files*.

TS files must contain contiguous 188-byte TS packets without any encapsulation. All TS packets shall start with the MPEG-defined synchronization byte `0x47`. Any packet not starting with this synchronization byte is considered invalid and rejected.

When dealing with non-conformant TS files coming from outside, the utility *tsresync* can be used to extract the TS packets and recreate a pure 188-byte TS file. Here is a sample list of common non-conformant TS files which can be processed by *tsresync* :

- Raw capture of TS packets with the 16-byte trailing Reed-Solomon correction code.
- M2TS files where each packet is preceded by a 4-byte time stamp. This format is found on Blu-Ray discs and some DVB or IP-TV recorders.
- Network capture files as produced by tools like Wireshark. Such files contain network packets, containing IP packets, containing UDP packets, containing TS packets.

In all these cases, *tsresync* can extract all TS packets and recreate a “pure” TS file which can be manipulated by the various utilities and plugins from the TSDuck suite.

2.2 PSI/SI Signalization Storage Format

TSDuck can manipulate PSI/SI sections and tables outside of transport streams. Sections and tables can be extracted from a transport stream, saved and manipulated in various file formats and injected in other transport streams.

There are two main file formats for PSI/SI: binary section files and XML text files.

These two formats are documented in the next sections. In the general case, tools which extract PSI/SI sections and tables can save in any format and tools which use PSI/SI can read them from any format as well. The utility *tstabcomp*, the table compiler, can translate between the two formats.

Some key differences between the two formats are:

- Binary section files contain collections of individual sections in any order, not necessarily complete tables. XML files contain complete tables only.

- Binary section files contain the exact representation, byte by byte, of sections which were extracted from a transport stream. XML files contain a higher-level representation.
- Binary section files are not easily modifiable. XML files contain text which can be manually edited using any text editor or XML tool.

2.2.1 PSI/SI binary format

A PSI/SI binary file contains one or more sections in a simple binary format. Each section is directly written in the file without any encapsulation or synchronization information. All sections are contiguous in the file.

A binary file must be read from the beginning. The header of each section contains the section length. Using this length information, it is possible to locate the next section, starting right after the current section, and so on down to the end of the file.

2.2.1.1 Creating PSI/SI binary files

PSI/SI binary files can be extracted from live streams or TS files using *tstables* or the plugin *tables*. The extracted sections are identical, byte by byte, to the transported sections. By default, all sections of a given table are contiguously saved in the binary file, in increasing order of section number. Thus, a complete table can be easily rebuilt by reading sections one by one.

With the option `--all-sections`, *tstables* and the plugin *tables* save all individual sections in their order of reception. In that case, the order and repetition of sections in the binary files are not defined.

PSI/SI binary files can also be created by *tstabcomp*, the table compiler. Tables are described in XML format (see 2.2.2) and compiled into a binary file. Since *tstabcomp* processes complete tables, all sections of a table are also contiguously saved in the binary file, in increasing order of section number, just like *tstables* by default.

2.2.1.2 Using PSI/SI binary files

The content of binary section files can be viewed using *tstabdump*. This utility displays the content of each individual section in a human-readable format, regardless of the order of sections in the file.

Binary section files can be used to packetize or inject sections in a stream (command *tspacketize* and plugin *inject*). The sections are packetized or injected in their order of appearance in the file.

Finally, binary section files can also be decompiled by *tstabcomp* to recreate the corresponding XML files from the binary tables. But note that XML files contain complete tables only. This means that tables can be recreated only when their sections are contiguous and in increasing order of section number in the binary file.

2.2.2 PSI/SI XML format

An XML file containing PSI/SI tables for TSDuck uses `<tsduck>` as root node. The root node contains any number of tables.

Unlike binary files which may contain individual sections, XML files can only contain complete tables. The XML format represents a higher level view of a table, regardless of the binary implementation in one or more sections.

The following sample XML file contains the definition for simple (and incomplete) PAT and PMT.

```
<?xml version="1.0" encoding="UTF-8"?>
<tsduck>

  <PAT version="8" transport_stream_id="0x0012" network_PID="0x0010">
    <service service_id="0x0001" program_map_PID="0x1234"/>
    <service service_id="0x0002" program_map_PID="0x0678"/>
  </PAT>

  <PMT version="4" service_id="0x0456" PCR_PID="0x1234">
    <CA_descriptor CA_system_id="0x0777" CA_PID="0x0251"/>
    <component elementary_PID="0x0567" stream_type="0x12">
```

```
        <CA_descriptor CA_system_id="0x4444" CA_PID="0x0252"/>
        <ISO_639_language_descriptor>
          <language code="fre" audio_type="0x45"/>
          <language code="deu" audio_type="0x78"/>
        </ISO_639_language_descriptor>
      </component>
    </PMT>

  </tsduck>
```

All XML files shall be encoded in UTF-8 format to allow international character sets in service names or event descriptions for instance. The initial declaration line "*<?xml version="1.0" encoding="UTF-8"?>*" is optional but recommended.

The complete definition of the XML model can be found in Appendix C, page 296.

3 Transport Stream Utilities

The transport stream toolkit provides a number of command-line utilities. The main one is *tsp*, the transport stream processor. The other utilities are small tools which work on transport stream files.

With a few exceptions, the transport stream files are continuous streams of 188-byte TS packets. These files can also be pipes. With the help of the *tsp* and its input and output plugins, the TS packets can be piped from and to various devices and protocols (files, DVB-ASI, DVB-S, DVB-C, DVB-T, multicast IP, etc.)

The Table 1 lists all transport stream utilities:

Table 1: TSDuck utilities

Utility	Description
`tsanalyze`	Analyze a TS file and display various information about the transport stream and each individual service and PID.
`tsbitrate`	Evaluate the original bitrate of a TS based on the analysis of the PCR's and the number of packets between them.
`tscmp`	Compare the binary content of two TS files.
`tsdate`	Display the date & time information (TDT & TOT) from a TS file.
`tsdektec`	Control a Dektec device.
`tsdump`	Dump the content of a TS file.
`tsecmg`	DVB SimulCrypt-compliant ECMG stub for system integration and debug.
`tsemmg`	DVB SimulCrypt-compliant EMMG stub for system integration and debug.
`tsfixcc`	Fix continuity counters in a TS file.
`tsftrunc`	Truncate a TS file, removing extraneous bytes (last incomplete TS packet) or truncating after a specified TS packet.
`tsgenecm`	Generate one ECM using any DVB SimulCrypt compliant ECMG.
`tshides`	List HiDes modulator devices.
`tslsdvb`	List DVB receiver devices.
`tsp`	General-purpose TS processor: receive a TS from a user-specified input plugin, apply MPEG packet processing through several user-specified packet processor plugins and send the processed stream to a user-specified output plugin.
`tspacketize`	Packetize PSI/SI tables in a transport stream PID.
`tspsi`	Display the PSI (PAT, CAT, NIT, PMT, SDT) from a TS file.
`tsresync`	Resynchronize a captured TS file: locate start of first packet, resynchronize to next packet after holes, convert to 188-byte packets (if captured with 204-byte packets).
`tsscan`	Scan frequencies in a DVB network.
`tssmartcard`	List or reset smart-card reader devices.
`tsstuff`	Add stuffing to a TS file to reach a target bitrate.
`tsswitch`	Transport stream input source switch using remote control.
`tstabcomp`	PSI / SI table compiler from / to XML files.
`tstabdump`	Dump binary tables files, as previously saved by `tstables`.
`tstables`	Collect specified PSI/SI tables from a TS file. Either display them or save them in binary files.
`tsterinfo`	Compute or retrieve various DVB-T (terrestrial) information.
`tsversion`	Check version, download and upgrade TSDuck.

3.1 Command line syntax

3.1.1 Command line options

All utilities are simple command-line tools. They accept *options* and *parameters*. The syntax of options follows the GNU `getopt_long(3)` conventions. See the corresponding Linux manual page for details.

In short, this means that all options have a "long name" preceded by a double dash and optionally a short name (one dash, one letter). Long options can be abbreviated if there is no ambiguity.

Although this syntax is inspired by Linux and the GNU utilities, the same syntax is used on Windows.

As an example, consider a utility which accepts the two options `--verbose` (short name `-v`) and `--version` (no short name). Then, the verbose mode can be equally triggered by `-v`, `--verbose`, `--verb` but not `--ver` since there is an ambiguity with `--version`.

3.1.2 Integer values in command line options

When an option or parameter is documented to require an integer value (PID, identifier, etc.), this value can be uniformly specified in decimal or hexadecimal format (with the `0x` prefix).

In decimal values, the commas which are used as separators for groups of thousands are ignored. Since most commands display large values with separators in order to improve the readability, these values can be simply copied / pasted in subsequent command lines.

Example: The following options are equivalent:

```
--count 3,100,456
--count 3100456
```

When the same option is allowed to be specified several times in one command, it is possible to use ranges of integer values (two values, separated with a dash) instead of specifying all values individually.

Example: The following sets of options are equivalent:

```
--pid 0 --pid 0x20 --pid 0x21 --pid 0x22 --pid 0x23 --pid 0x24 --pid 0x25 --pid 0x40
--pid 0 --pid 0x20-0x25 --pid 0x40
```

3.1.3 Predefined common options

All commands accept the following common options:

`--debug[=`*`N`*`]`

Produce verbose debug output. Specify an optional debug level *N*. Do not use this option in normal operation.

Without this option, no debug output is produced. When the option is specified but not the level *N*, the default debug level is 1, that is to say a reasonable amount of information. The higher the debug level is, the more output is produced.

The amount of debug information depends on the command. Some commands do not generate any debug information.

`--help`

The utility displays its syntax and exits.

If either the standard output or the standard error is a terminal, the help text is "paged" through a system utility such as `less` or `more`, whichever is available. The environment variable `PAGER` can be used to specify an alternate pager command with its parameters.

To redirect the help text to a file, you must redirect both the standard output and standard error. Otherwise, since at least one of the two is a terminal, the pager will be used. Example:

```
tsp --help &>help.txt
```

`--verbose`

Display verbose information.

--version

The utility displays the TSDuck version and exits.

All *tsp* plugins accept the option `--help` which provides help on this specific plugin.

3.1.4 Using a pager command

Some commands which produce a very verbose output are automatically redirected to a "pager" command such as `less` or `more`, whichever is available. The redirection is performed only when the standard output is a terminal.

The environment variable `PAGER` can be used to specify an alternate pager command with its parameters.

These commands always define a `--no-pager` option to disable the redirection even when the standard output is a terminal.

3.1.5 Partial command line redirection from a file

In any command, it is possible to read some or all options and parameter from a file. The syntax is "`@filename`" where *filename* is a text file containing options and parameters.

In the text file, each line must contain exactly one item (option name, option value or parameter).

Sample command:

```
tsp -v @dvb.txt -P until --seconds 20 -P analyze -o out.txt -O drop
```

The file *dvb.txt* contains a list of command line items, one per line. The content of the file *dvb.txt* exactly replaces the expression "`@dvb.txt`".

Sample content of this file:

```
-I
dvb
--frequency
12,169,000,000
--symbol-rate
27,500,000
--fec-inner
3/4
--polarity
horizontal
--delivery-system
DVB-S2
--modulation
8-PSK
```

Note that each line contains exactly one command line item. Spaces or special characters are not filtered or interpreted. Using that kind of command can be useful in several situations:

- When a custom application generates long and complicated TSDuck commands.
- When the options or parameters contain special characters, spaces or any other sequence which must be properly escaped with some shells, possibly differently between shells or operating systems.

Command line parameter redirections can be nested. When one line of such a text file contains a pattern "`@filename`", the second file is inserted here.

Finally, if a parameter really starts with a @ character (which can be possible in a service name for instance), use a double @ to indicate that this is a literal @ character and not a redirection.

Consider the following command:

```
tsp -v @dvb.txt -P zap @@home -O drop
```

This command reads parameters from the file *dvb.txt* to find the tuning options and extracts the service named "`@home`" (with one @). The double @ has been used to indicate that this is a literal @.

And since redirections can be nested, the initial @@ escape sequence can also be used inside text files containing parameters.

3.1.6 Default options from the TSDuck configuration file

It is possible to specify default command line options or alternate options in a global configuration file. This configuration file is specific per user. See Appendix A, page 291, for a complete reference of the TSDuck configuration file.

The rest of this chapter documents all TSDuck utilities, in alphabetical order.

tsanalyze

Transport Stream Analysis

This utility analyzes a transport stream. It reports either a full analysis of the transport stream, services and PID's (either in human readable format or normalized format for automatic analysis) or selected individual information.

The output can include full synthetic analysis (options `--*-analysis`), full normalized output (option `--normalized`) or a simple list of values on one line (options `--*-list`). The second and third type of options are useful to write automated scripts.

If output control options are specified, only the selected outputs are produced. If no such option is given, the default is:

```
--ts-analysis --service-analysis --pid-analysis --table-analysis
```

See also the *analyze* plugin for tsp for the equivalent tool in the context of tsp.

Usage

```
tsanalyze [options] [input-file]
```

Input file

MPEG transport stream, either a capture file or a pipe from a live stream. Must be a binary stream of 188-byte packets. If omitted, standard input is used.

General purpose options

-b *value*
--bitrate *value*

Specifies the bitrate of the transport stream in bits/second (based on 188-byte packets). By default, the bitrate is evaluated using the PCR in the transport stream. If no bitrate can be determined (no user-specified value, no PCR), the analysis will not report the bitrates of the individual services and PID's.

--no-pager

Do not send output through a pager process. By default, if the output device is a terminal, the output is paged. See 3.1.4 for more details.

Analysis control options

--atsc

Assume that the transport stream is an ATSC one.

ATSC streams are normally automatically detected from their signalization. This option is only useful when ATSC-related stuff are found in the TS before the first ATSC-specific table. For instance, when a PMT with ATSC-specific descriptors is found before the first ATSC MGT or VCT.

--default-charset *name*

Default character set to use when interpreting DVB strings without explicit character table code.

According to DVB standard ETSI EN 300 468, the default DVB character set is ISO-6937. However, some bogus signalization may assume that the default character set is different, typically the usual local character table for the region. This option forces a non-standard character table.

The available table names are: ISO-6937, ISO-8859-1, ISO-8859-10, ISO-8859-11, ISO-8859-13, ISO-8859-14, ISO-8859-15, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, UNICODE, UTF-8.

--europe

A synonym for `'--default-charset ISO-8859-15'`. This is a handy shortcut for commonly incorrect signalization on some European satellites. In that signalization, the character encoding is ISO-8859-15, the most common encoding for Latin & Western Europe languages. However, this is not the default DVB character set and it should be properly specified in all strings, which is not

the case with some operators. Using this option, all DVB strings without explicit table code are assumed to use ISO-8859-15 instead of the standard ISO-6937 encoding.

--suspect-max-consecutive *value*

Specifies the maximum number of consecutive *suspect* packets. The default value is 1. If set to zero, the suspect packet detection is disabled.

Suspect packets are TS packets which are technically correct but which may be suspected of being incorrect, resulting in analysis errors. Typically, in the middle of a suite of packets with uncorrectable binary errors, one packet may appear to have no such error while it has some errors in fact. To avoid adding this type of packets in the analysis, a packet is declared as *suspect* (and consequently ignored in the analysis) when:

- its PID is unknown (no other packet was found in this PID)
- it immediately follows a certain amount of packet containing errors (see option `--suspect-min-error-count`)
- it immediately follows no more than the specified number consecutive suspect packets.

--suspect-min-error-count *value*

Specifies the minimum number of consecutive packets with errors before starting "suspect" packet detection. See also option `--suspect-max-consecutive`. The default value is 1. If set to zero, the suspect packet detection is disabled.

Output control options

--ts-analysis

Report global transport stream analysis.

--service-analysis

Report analysis for each service.

--pid-analysis

Report analysis for each PID.

--table-analysis

Report analysis for each table.

--error-analysis

Report analysis about detected errors.

--normalized

Complete report about the transport stream, services, PID's and tables in a normalized output format (see details below). This type of output is useful for automatic analysis in scripts.

--service-list

Report the list of all service ids.

--pid-list

Report the list of all PID's.

--global-pid-list

Report the list of all global PID's, that is to say PID's which are not referenced by a specific service but are standard DVB PSI/SI PID's or are referenced by them. This include, for instance, PID's of the PAT, EMM's, EIT's, stuffing, etc.

--unreferenced-pid-list

Report the list of all unreferenced PID's, that is to say PID's which are neither referenced by a service nor known as or referenced by the standard DVB PSI/SI.

--service-pid-list *value*

Report the list of all PID's which are referenced by the specified service id.

--pes-pid-list

Report the list of all PID's which are declared as carrying PES packets (audio, video, subtitles, etc).

--title *string*

Display the specified string as title header.

--prefix *string*

For one-line displays (options `--*-list`), prepend the specified string to all values. For instance, options `--global --prefix -p` outputs something like `'-p 0 -p 1 -p 16'`, which is an acceptable option list for the tsp `filter` plugin.

-w
--wide-display

Use a wider grid display with more information on each line.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N***]**

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

--verbose

Produce verbose messages.

--version

Display the version number.

Normalized output format

In normalized output, each line describes one *object* (service, PID, table, etc). The format of each line is:

```
type:name[=value]:...
```

The *type* identifies the kind of object which is described by the line. The *name* identifies a characteristics for the object with an optional *value*. There is no space characters. All integer values are in decimal format.

The normalized syntax can be used to search for specific objects with specific characteristics.

Example: The following sample command extracts the list of EMM PID's for the SafeAccess CAS. The object *type* is `pid` (at beginning of line) and the two selected characteristics are `emm` (no value) and `cas` with SafeAccess DVB-assigned CA_system_id value (`0x4ADC`, which is `19164` in decimal).

```
tsanalyze --normalize ... | \
grep '^pid:' | grep ':emm:' | grep ':cas=19164:' | \
sed -e 's/.*:pid=//' -e 's/:.*//'
```

Other more complex examples of automated scripts are available in chapter 5.

Normalized object types

The list of *type*, at beginning of lines, is the following:

`ts:`	Global transport stream description. There is always one single `ts` line.
`global:`	Summary of global PID's, ie. not attached to a specific service. There is always one single `global` line.
`unreferenced:`	Summary of unreferenced PID's, ie. neither global nor attached to a specific service. There is always one single `unreferenced` line.
`service:`	Description of one service. There is one `service` line per service.
`pid:`	Description of one PID. There is one `pid` line per PID.

`table:`	Description of one table on one PID. There is one `table` line per unique table per PID.
`time:`	Time description, either from the TDT/TOT tables or from the running system.

Normalized transport stream characteristics

The characteristics in `ts:` lines are:

`:id=`*`int`*`:`	Optional. Transport stream id, when found.
`:packets=`*`int`*`:`	Total number of TS packets.
`:bytes=`*`int`*`:`	Total number of bytes.
`:services=`*`int`*`:`	Number of services.
`:clearservices=`*`int`*`:`	Number of clear (not scrambled) services.
`:scrambledservices=`*`int`*`:`	Number of scrambled services.
`:pids=`*`int`*`:`	Number of PID's.
`:clearpids=`*`int`*`:`	Number of clear (not scrambled) PID's.
`:scrambledpids=`*`int`*`:`	Number of scrambled PID's.
`:pcrpids=`*`int`*`:`	Number of PID's with PCR's.
`:unreferencedpids=`*`int`*`:`	Number of unreferenced PID's.
`:invalidsyncs=`*`int`*`:`	Number of TS packets with invalid synchronization byte.
`:transporterrors=`*`int`	Number of TS packets with *transport error indicator*.
`:suspectignored=`*`int`	Number of suspect TS packets which were ignored in the analysis.
`:bitrate=`*`int`*`:`	Best value for transport stream bitrate in b/s.
`:bitrate204=`*`int`*`:`	Same as previous, based on 204-byte packets.
`:userbitrate=`*`int`*`:`	User-specified value for transport stream bitrate in b/s. Zero if none. When used within *tsp* plugin, the user-specified bitrate comes from previous plugins in the chain.
`:userbitrate204=`*`int`*`:`	Same as previous, based on 204-byte packets.
`:pcrbitrate=`*`int`*`:`	Estimated transport stream bitrate in b/s, based on PCR analysis. Zero if unable to analyze PCR (no or not enough PCR, too many discountinuities, etc.)
`:pcrbitrate204=`*`int`*`:`	Same as previous, based on 204-byte packets.
`:duration=`*`int`*`:`	Duration of transmission in seconds, based on TS bitrate.
`:country=`*`name`*`:`	Optional. First region name in TOT.

Normalized global and unreferenced PID's summary characteristics

The characteristics in `global:` and `unreferenced:` lines are:

`:pids=`*`int`*`:`	Total number of global or unreferenced PID's.
`:clearpids=`*`int`*`:`	Number of clear (not scrambled) global or unreferenced PID's.
`:scrambledpids=`*`int`*`:`	Number of scrambled global or unreferenced PID's.
`:packets=`*`int`*`:`	Total number of TS packets in global or unreferenced PID's.
`:bitrate=`*`int`*`:`	Total bitrate of global or unreferenced PID's.
`:bitrate204=`*`int`*`:`	Same as previous, based on 204-byte packets.
`:access=`*`type`*`:`	Value is `scrambled` if there is at least one scrambled PID in the category and `clear` otherwise.
`:pidlist=`*`int`*`,`*`int`*`,...:`	List of global or unreferenced PID's.

Normalized service characteristics

The characteristics in `service:` lines are:

`:id=`*int*`:`	Service id.
`:tsid=`*int*`:`	Transport stream id.
`:orignetwid=`*int*`:`	Original network id.
`:servtype=`*int*`:`	Service type.
`:access=`*type*`:`	Value is `scrambled` if there is at least one scrambled PID in the service and `clear` otherwise.
`:pids=`*int*`:`	Number of PID's in the service. Note that ECM PID's are also included.
`:clearpids=`*int*`:`	Number of clear (not scrambled) PID's in the service.
`:scrambledpids=`*int*`:`	Number of scrambled PID's in the service.
`:packets=`*int*`:`	Total number of TS packets in the service.
`:bitrate=`*int*`:`	Total bitrate of the service in b/s.
`:bitrate204=`*int*`:`	Same as previous, based on 204-byte packets.
`:ssu:`	Optional. Indicate that the service carries a System Software Update PID.
`:t2mi:`	Optional. Indicate that the service carries a T2-MI (DVB-T2 Modulator Interface) PID.
`:pmtpid=`*int*`:`	Optional. PID of the service's PMT.
`:pcrpid=`*int*`:`	Optional. PCR PID of the service, as declared in the PMT.
`:pidlist=`*int*`,`*int*`,...:`	List of PID's in the service.
`:provider=`*name*`:`	Service provider name.
`:name=`*name*	Service name. Note that this is always the last item in the line. The value is not terminated by a colon ('`:`'). So, if a colon is present, it is part of the service name.

Normalized PID characteristics

The characteristics in `pid:` lines are:

`:pid=`*int*`:`	PID number.
`:pmt:`	Optional. Indicate that this is a PMT PID.
`:ecm:`	Optional. Indicate that this is an ECM PID.
`:emm:`	Optional. Indicate that this is an EMM PID.
`:cas=`*int*`:`	Optional. Related *CA_system_id* for ECM or EMM PID's.
`:operator=`*int*`:`	Optional. Related CA system operator id, when applicable, for ECM or EMM PID's.
`:access=`*type*`:`	Value is `scrambled` if there is at least one scrambled packet in the PID and `clear` otherwise.
`:cryptoperiod=`*int*`:`	Optional. Average crypto-period duration in seconds for scrambled PID's, when it can be evaluated.
`:streamid=`*int*`:`	Optional. PES *stream_id* in PES packet headers when the PID carries PES packets and all PES packets have the same *stream_id*.
`:audio:`	Optional. Indicate that this is an audio PID.
`:video:`	Optional. Indicate that this is a video PID.
`:language=`*name*`:`	Optional. Indicate the language for the PID. Can be found on audio or subtitles PID's.
`:servcount=`*int*`:`	Number of services which reference this PID.
`:unreferenced:`	Optional. Indicate that this is an unreferenced PID.
`:global:`	Optional. Indicate that this is a global PID.
`:servlist=`*int*`,`*int*`,...:`	Optional. List of *service_id* which reference this PID.
`:ssuoui=`*int*`,`*int*`,...:`	Optional. List of manufacturers OUI for System Software Update data

	PID's.
`:t2mi:`	Optional. Indicate that the PID carries a T2-MI stream.
`:plp=`*`int`*`,`*`int`*`,...:`	Optional. List of T2-MI PLP (Physical Layer Pipe) id.
`:bitrate=`*`int`*`:`	Bitrate for this PID in b/s.
`:bitrate204=`*`int`*`:`	Same as previous, based on 204-byte packets.
`:packets=`*`int`*`:`	Total number of TS packets in this PID.
`:clear=`*`int`*`:`	Number of clear (not scrambled) TS packets in this PID.
`:scrambled=`*`int`*`:`	Number of scrambled TS packets in this PID.
`:af=`*`int`*`:`	Number of TS packets with adaptation field in this PID.
`:pcr=`*`int`*`:`	Number of TS packets with PCR in this PID.
`:discontinuities=`*`int`*`:`	Number of discontinuities in this PID.
`:duplicated=`*`int`*`:`	Number of duplicated TS packets in this PID.
`:invalidscrambling=`*`int`*`:`	Number of TS packets in this PID with invalid scrambling control value.
`:pes=`*`int`*`:`	Optional. Number of PES packets, for PID's carrying PES.
`:invalidpesprefix=`*`int`*`:`	Optional. Number of invalid PES prefix, for PID's carrying PES.
`:unitstart=`*`int`*`:`	Optional. Number of PUSI (*payload unit start indicator*), for PID's not carrying PES.
`:description=`*`string`*	Human-readable description of this PID. Note that this is always the last item in the line. The value is not terminated by a colon ('`:`'). So, if a colon is present, it is part of the description.

Normalized table and sections characteristics

The characteristics in `table:` lines are:

`:pid=`*`int`*`:`	PID number on which the table is found.
`:tid=`*`int`*`:`	Table id.
`:tidext=`*`int`*`:`	Optional. Table id extension, for long sections only.
`:tables=`*`int`*`:`	Total number of occurences of the table.
`:sections=`*`int`*`:`	Total number of sections for this table.
`:repetitionms=`*`int`*`:`	Optional. Average repetition rate in milliseconds (can be computed only if the transport stream bitrate is known).
`:minrepetitionms=`*`int`*`:`	Optional. Minimum repetition rate in milliseconds (can be computed only if the transport stream bitrate is known).
`:maxrepetitionms=`*`int`*`:`	Optional. Maximum repetition rate in milliseconds (can be computed only if the transport stream bitrate is known).
`:repetitionpkt=`*`int`*`:`	Average repetition rate in TS packets interval.
`:minrepetitionpkt=`*`int`*`:`	Minimum repetition rate in TS packets interval.
`:maxrepetitionpkt=`*`int`*`:`	Maximum repetition rate in TS packets interval.
`:firstversion=`*`int`*`:`	Optional. Version number of first occurrence of the table. For long sections only.
`:lastversion=`*`int`*`:`	Optional. Version number of last occurrence of the table. For long sections only.
`:versions=`*`int`*`,`*`int`*`,...:`	Optional. List of all version numbers of the table. For long sections only.

Normalized time characteristics

The characteristics in `time:` lines are:

`:utc:`	Optional. The specified time is UTC.
`:local:`	Optional. The specified time is local time.

`:tdt:`	Optional. The specified time is extracted from a TDT.
`:tot:`	Optional. The specified time is extracted from a TOT.
`:system:`	Optional. The specified time is an operating system time, not extracted from the transport stream.
`:first:`	Optional. The specified time is the first one in its category (first TDT or TOT, system time of first packet).
`:last:`	Optional. The specified time is the last one in its category (last TDT or TOT, system time of last packet).
`:date=`*dd*`/`*mm*`/`*yyyy*`:`	Date part of the time, example: "`24/11/2008`".
`:time=`*hh*`h`*mm*`m`*ss*`s:`	Hour, minute and second part of time, example: "`14h12m45s`".
`:secondsince2000=`*int*`:`	Number of seconds since 1st January 2000. Can be used to compute duration, to compare time values, etc.
`:country=`*name*`:`	Optional. First region name in TOT, if the time comes from a TOT.

tsbitrate

Bitrate Evaluation from PCR

This utility evaluates the original bitrate of a transport stream based on an analysis of the PCR's (Program Clock Reference timestamps) and the interval between them. This is especially useful for captured files where the transmission bitrate information is lost.

Usage

```
tsbitrate [options] [input-file]
```

Input file

MPEG transport stream, either a capture file or a pipe from a live stream. Must be a binary stream of 188-byte packets. If omitted, standard input is used.

Options

-a
--all

Analyze all packets in the input file. By default, stop analysis when enough PCR information has been collected.

-d
--dts

Use DTS (Decoding Time Stamps) from video PID's instead of PCR (Program Clock Reference) from the transport layer.

-f
--full

Full analysis. The file is entirely analyzed (as with `--all`) and the final report includes a complete per PID bitrate analysis.

-i
--ignore-errors

Ignore transport stream errors such as discontinuities.

When errors are not ignored (the default), the bitrate of the original stream (before corruptions) is evaluated. When errors are ignored, the bitrate of the received stream is evaluated, missing packets being considered as non-existent.

--min-pcr *value*

Stop analysis when that number of PCR's are read from the required minimum number of PID's (default: stop after 64 PCR's on 1 PID).

--min-pid *value*

Minimum number of PID to get PCR's from (default: stop after 64 PCR's on 1 PID).

-v
--value-only

Display only the bitrate value, in bits/seconds, based on 188-byte packets. Useful to reuse the value in command lines.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N*]

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

`--verbose`

Produce verbose messages.

`--version`

Display the version number.

tscmp

Transport Stream Files Comparison

This utility compares the binary content of two transport stream files. Selected fields may be omitted in the comparison to allow comparing files which went through different PID remapping or resynchronization process.

Usage

```
tscmp [options] filename-1 filename-2
```

Input files

MPEG transport stream files to be compared.

Options

--buffered-packets *value*

Specifies the files input buffer size in TS packets. The default is 10,000 TS packets.

-b *value*
--byte-offset *value*

Start reading the files at the specified byte offset (default: zero).

--cc-ignore

Ignore continuity counters when comparing packets. Useful if one file has been resynchronized.

-c
--continue

Continue the comparison up to the end of files. By default, stop after the first differing packet.

-d
--dump

Dump the content of all differing packets. Also separately dump the differing area within the packets.

-n
--normalized

Report in a normalized output format (useful for automatic analysis).

-p *value*
--packet-offset *value*

Start reading the files at the specified TS packet (default: zero).

--payload-only

Compare only the payload of the packets, ignore header and adaptation field.

--pcr-ignore

Ignore PCR and OPCR when comparing packets. Useful if one file has been resynchronized.

--pid-ignore

Ignore PID value when comparing packets. Useful if one file has gone through a remapping process.

-q
--quiet

Do not output any message. The process simply terminates with a success status if the files are identical and a failure status if they differ.

-s
--subset

Specifies that the second file is a subset of the first one. This means that the second file is expected to be identical to the first one, except that some packets may be missing. When a difference is found, the first file is read ahead until a matching packet is found. Without this option, missing packets in the second file cause all the rest of the file to be considered as different.

See also `--threshold-diff`.

-t *value*
--threshold-diff *value*

When used with `--subset`, this value specifies the maximum number of differing bytes in packets to declare them equal. When two packets have more differing bytes than this threshold, the packets are reported as different and the first file is read ahead. The default is zero, which means that two packets must be strictly identical to declare them equal.

If you find this explanation unclear, try it with a second file which contains both missing and corrupted packets...

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N*]

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

--verbose

Produce verbose messages.

--version

Display the version number.

tsdate

Date and Time Extraction

This utility extracts date and time information from s transport stream, namely the TDT (Time and Data Table) and the TOT (Time Offset Utility).

Usage

```
tsdate [options] [input-file]
```

Input file

MPEG transport stream, either a capture file or a pipe from a live stream. Must be a binary stream of 188-byte packets. If omited, standard input is used.

Options

-a
--all

Report all TDT/TOT tables (default: report only the first table of each type).

--notdt

Ignore Time & Date Table (TDT).

--notot

Ignore Time Offset Table (TOT).

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N*]

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

--verbose

Produce verbose messages.

--version

Display the version number.

tsdektec

Dektec Device Control

This utility controls Dektec devices, which include input and / or output DVB-ASI devices, QPSK or QAM modulators (see [22]).

Usage

```
tsdektec [options] [device]
```

Device

Device index, from 0 to N-1 (with N being the number of Dektec devices in the system). The default is 0. Use option `--list-all` (or `-a`) to have a complete list of devices in the system.

Options

-a
--list-all

List all Dektec devices available on the system.

-i *port-number*
--input *port-number*

Set the specified port in input mode. This applies to bidirectional ports which can be either set in input or output mode. The port number of each channel can be seen using the command "`tsdektec -av`".

-l *state*
--led *state*

Set the state of the LED on the rear panel. Useful to identify a Dektec device when more than one is present. The state is one of "`off`", "`green`", "`red`", "`yellow`", "`blue`", "`hardware`". See also option `--wait` (the led state is automatically returned to "`hardware`" after exit).

-n
--normalized

With `--all`, list the Dektec devices in a normalized output format (useful for automatic analysis).

-o ***port-number***
--output ***port-number***

Set the specified port in output mode. This applies to bidirectional ports which can be either set in input or output mode. The port number of each channel can be seen using the command "`tsdektec -av`".

-r
--reset

Reset the device.

-w *seconds*
--wait *seconds*

Wait the specified number of seconds before exiting. The default if 5 seconds if option `--led` is specified and 0 otherwise.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N*]

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose

Produce verbose messages.

--version

Display the version number.

Normalized output format

In normalized output, each line describes one *object* (driver, device, channel, etc). The format of each line is:

```
type:name[=value]:...
```

The *type* identifies the kind of object which is described by the line. The *name* identifies a characteristics for the object with an optional *value*. There is no space characters. All integer values are in decimal format.

The normalized syntax can be used to search for specific objects with specific characteristics. See also the description of the command *tsanalyze* for another example of normalized output.

Normalized object types

The list of *type*, at beginning of lines, is the following:

`dtapi:`	Description of the Dektec runtime library ("DTAPI"). There is always one single `dtapi` line.
`driver:`	Description of one type of Dektec device driver.
`device:`	Description of one Dektec device.
`channel:`	Description of one channel inside a Dektec device.

Normalized DTAPI characteristics

The characteristics in `dtapi:` lines are:

`:version=`*`string`*`:`	Version of the DTAPI.

Normalized driver characteristics

The characteristics in `driver:` lines are:

`:pci:`	This is a PCI driver (Dta1xx)
`:usb:`	This is a USB driver (Dtu2xx)
`:version=`*`string`*`:`	Version of the driver.

Normalized device characteristics

The characteristics in `device:` lines are:

`:address=`*`int`*`:`	USB address.
`:bus=`*`int`*`:`	PCI bus number.
`:device=`*`int`*`:`	Device index.
`:device-id=`*`int`*`:`	Device id
`:fw-variant=`*`int`*`:`	Firmware variant.
`:fw-version=`*`int`*`:`	Firmware version.
`:model=`*`string`*`:`	Device model name.
`:nb-input=`*`int`*`:`	Count of input ports.
`:nb-output=`*`int`*`:`	Count of output ports.
`:nb-port=`*`int`*`:`	Count of all ports.
`:pci:`	This is a PCI device.
`:serial=`*`int`*`:`	Serial number.
`:slot=`*`int`*`:`	PCI slot number in the PCI bus.

`:subsys-id=`*int*`:`	Subsystem id
`:subsys-vendor-id=`*int*`:`	Subsystem vendor id
`:usb:`	This is a USB device.
`:vendor-id=`*int*`:`	Vendor id
`:vpd-bo=`*string*`:`	Bitrate offset (from Vital Product Data area)
`:vpd-cl=`*string*`:`	Customer id (from Vital Product Data area)
`:vpd-ec=`*string*`:`	Engineering change level (from Vital Product Data area)
`:vpd-id=`*string*`:`	Device description (from Vital Product Data area)
`:vpd-mn=`*string*`:`	Manufacture id (from Vital Product Data area)
`:vpd-pd=`*string*`:`	Production date (from Vital Product Data area)
`:vpd-pn=`*string*`:`	Part number (from Vital Product Data area)
`:vpd-sn=`*string*`:`	Serial number (from Vital Product Data area)
`:vpd-xt=`*string*`:`	Crystal stability (from Vital Product Data area)

Normalized channel characteristics

The characteristics in `channel:` lines are:

`:access-downconverted:`	Access to downconverted signal.
`:adjust-level:`	Adjustable level
`:asi:`	This is a DVB/ASI port.
`:asi-raw-10bit:`	Raw 10-bit ASI mode available.
`:atsc:`	ATSC modulator.
`:bidir:`	This is a bidirectional port.
`:channel=`*int*`:`	Channel index inside device.
`:channel-modelling:`	Channel modelling available.
`:cmmb:`	CMMB modulator.
`:dedicated-clock-input:`	Dedicated clock input available.
`:dedicated-clock-input-ratio:`	Dedicated clock input available, can be divided by providing a ratio.
`:device=`*int*`:`	Device index of the device containing the channel.
`:diversity:`	Diversity mode available.
`:double-buffer:`	This is a double-buffered device.
`:dtmb:`	DTMB modulator.
`:dvb-c:`	DVB-C modulator.
`:dvb-c2:`	DVB-C2 modulator.
`:dvb-raw-10bit:`	DVB 10-bit raw mode available.
`:dvb-s:`	DVB-S modulator.
`:dvb-s2:`	DVB-S2 modulator.
`:dvb-t:`	DVB-T modulator.
`:dvb-t2:`	DVB-T2 modulator.
`:dvb-t2-mi:`	DVB-T2-MI modulator.
`:failsafe:`	Failsafe
`:if-output:`	IF output
`:ip=`*string*`:`	IP address
`:io-clock-select:`	I/O clock selection available.

`:io-config:`	I/O standard and mode configuration available.
`:io-rate-select:`	TS rate clock selection available.
`:iq-output:`	Digital IQ output.
`:iq-samples:`	Direct I/Q samples available.
`:isdb-s:`	ISDB-S modulator.
`:isdb-t:`	ISDB-T modulator.
`:lband:`	L-Band
`:lock-io-rate:`	Lock output to input TS rate available.
`:loop-through:`	Loop-through available.
`:lvds1:`	SPI LVDS1 available.
`:lvds2:`	SPI LVDS2 available.
`:lvttl:`	SPI LVTTL available.
`:mac=`*`string`*`:`	MAC address
`:modulator:`	This is a modulator port.
`:port=`*`int`*`:`	Port number.
`:qam:`	QAM modulator.
`:qam-a:`	QAM-A (DVB-C) modulator.
`:qam-b:`	QAM-B (USA) modulator.
`:qam-c:`	QAM-C (Japan) modulator.
`:shared-input:`	Shared antenna input available.
`:sdi:`	This is an SDI port.
`:sdi-time-stamp:`	SDI frames time-stamping available.
`:sdi-time-stamp-64:`	SDI frames 64-bit time-stamping available.
`:snr-setting:`	SNR setting available.
`:spi:`	This is an SPI port.
`:spi-external-clock:`	SPI external clock available.
`:spi-fixed-clock:`	SPI fixed clock available.
`:spi-serial-8-bit:`	SPI serial 8-bit available.
`:spi-serial-10-bit:`	SPI serial 10-bit available.
`:transmit-on-time-stamp:`	Transmission on time-stamp available.
`:transparent:`	Transparent mode available.
`:ts-over-ip:`	This an IP port, for TS over IP.
`:uhf:`	UHF modulator.
`:vhf:`	VHF modulator.
`:virtual-stream:`	Virtual stream channel.

tsdump

Dump TS packets

This utility dumps the contents of MPEG transport stream packets.

Usage

```
tsdump [options] [input-file ...]
```

Input files

MPEG transport streams, either capture files or a pipe from a live stream. Must be a binary stream of 188-byte packets. If omitted, the standard input is used.

Note that if the option `--raw` is used, the input files can be any type of file, not necessarily TS files.

Options

-a
--ascii

Include ASCII dump in addition to hexadecimal.

-b
--binary

Include binary dump in addition to hexadecimal.

-c
--c-style

Same as `--raw-dump` (no interpretation of packets) but dump the bytes in C-language style, eg. "`0x01, 0x02, `" instead of "`01 02`". Useful to include *tsdump* output as data in a C source file.

-h
--headers-only

Dump packet headers only, not payload.

-l
--log

Display a short one-line log of each packet instead of full dump.

--log-size *value*

With option `--log`, specify how many bytes are displayed in each packet.

The default is 188 bytes (complete packet).

-m *value*
--max-packets *value*

Maximum number of packets to dump per file.

-n
--nibble

Same as `--binary` but add separator between 4-bit nibbles.

--no-headers

Do not display header information.

--no-pager

Do not send output through a pager process. By default, if the output device is a terminal, the output is paged. See 3.1.4 for more details.

-o
--offset

Display offset from start of packet with hexadecimal dump.

--payload

Hexadecimal dump of TS payload only, skip TS header.

-p *pid1*[*-pid2*]
--pid *pid1*[*-pid2*]

Dump only packets with these PID values. Several `--pid` options may be specified.

By default, all packets are displayed.

-r
--raw-file

Raw dump of file, do not interpret as TS packets. With this option, tsdump simply acts as an hexa / ASCII file dumper.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N*]

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose

Produce verbose messages.

--version

Display the version number.

tsecmg

Minimal generic DVB SimulCrypt-compliant ECMG

This utility behaves as a DVB SimulCrypt compliant ECMG. It can be used to debug system integration, replacing any standard ECM Generator. Most DVB SimulCrypt parameters can be adjusted from the command line to test the behaviour of an SCS.

This fake ECMG can be used with the *tsp* plugin named *scrambler* to build an end-to-end demo of a DVB SimulCrypt system.

This fake ECMG accepts all *Super_CAS_Id* values. All ECM requests are instantaneously responded. The returned ECM is a fake one. The fake ECM's are TLV messages containing the access criteria and the control words as sent by the SCS in clear format.

Warning: It is obvious that this ECMG shall never be used on a production system since it returns ECM's with clear control words.

Usage

```
tsecmg [options]
```

Options

--ac-delay-start *value*

This option sets the DVB SimulCrypt option *AC_delay_start*, in milliseconds. By default, use the same value as `--delay-start`.

--ac-delay-stop *value*

This option sets the DVB SimulCrypt option *AC_delay_stop*, in milliseconds. By default, use the same value as `--delay-stop`.

--comp-time *value*

This option specifies the computation time of an ECM. The clear ECM's which are generated by this ECMG take no time to generate. But, in order to emulate the behaviour of a real ECMG, this parameter forces a delay of the specified duration before returning an ECM.

-c *value*
--cw-per-ecm *value*

Specify the required number of control words per ECM. This option sets the DVB SimulCrypt option *CW_per_msg*. It also set *lead_CW* to *CW_per_msg* - 1. By default, use 2 control words per ECM, the current one and next one.

--delay-start *value*

This option sets the DVB SimulCrypt option *delay_start*, in milliseconds. Default: 200 ms.

--delay-stop *value*

This option sets the DVB SimulCrypt option *delay_stop*, in milliseconds. Default: 200 ms.

--ecmg-scs-version *value*

Specifies the version of the ECMG ⇔ SCS DVB SimulCrypt protocol. Valid values are 2 and 3. The default is 2.

--log-data[=*level*]

Same as `--log-protocol` but applies to *CW_provision* and *ECM_response* messages only.

To debug the session management without being flooded by data messages, use `--log-protocol=info --log-data=debug`.

--log-protocol[=*level*]

Log all ECMG ⇔ SCS protocol messages using the specified level. If the option is not present, the messages are logged at debug level only. If the option is present without value, the messages are logged at info level. A level can be a numerical debug level or any of the following: `fatal`, `severe`, `error`, `warning`, `info`, `verbose`, `debug`.

--max-comp-time *value*

Specify the maximum ECM computation time in milliseconds. This option sets the DVB SimulCrypt option *max_comp_time*. By default, use the value of `--comp-time` (which is itself zero by default) plus 100 ms.

--no-reuse-port

Disable the reuse port socket option. Do not use unless completely necessary.

-o
--once

Accept only one client and exit at the end of the session.

-p *value*
--port *value*

TCP port number of the ECMG server. Default: 2222.

-r *value*
--repetition *value*

This option sets the DVB SimulCrypt option *ECM_rep_period*, the requested repetition period of ECM's, in milliseconds. Default: 100 ms.

-s
--section-mode

Return ECM's in section format. This option sets the DVB SimulCrypt parameter *section_TSpkt_flag* to zero. By default, ECM's are returned in TS packet format.

--transition-delay-start *value*

This option sets the DVB SimulCrypt option *transition_delay_start*, in milliseconds. Default: -500 ms.

--transition-delay-stop *value*

This option sets the DVB SimulCrypt option *transition_delay_stop*, in milliseconds. Default: 0 ms.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N*]

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose

Produce verbose messages.

--version

Display the version number.

tsemmg

Minimal generic DVB SimulCrypt-compliant EMMG

This utility behaves as a DVB SimulCrypt compliant EMMG. It can be used to debug system integration, replacing any standard EMM Generator. Most DVB SimulCrypt parameters can be adjusted from the command line to test the behaviour of a MUX.

This fake EMMG can be used with the *tsp* plugin named *datainject* to build an end-to-end demo of a DVB SimulCrypt system.

Usage

```
tsemmg [options] [section-file ...]
```

Parameters

The parameters are files containing sections in binary or XML format. Several files can be specified. All sections are loaded and injected in the MUX using the EMMG/PDG ⇔ MUX protocol. The list of all sections from all files is cycled as long as *tsemmg* is running. The sections can be of any type, not only EMM's.

By default, when no input file is specified, this EMMG generates fake EMM sections of a fixed size and all payload bytes contain the same value. The value of the fake EMM *table_id* and the value of the payload bytes are incremented in each new section. See options `--emm-size`, `--emm-min-table-id` and `--emm-max-table-id`.

Options

-b *value*
--bandwidth *value*

Specify the bandwidth of the data which are sent to the MUX in kilobits per second.

Default: 100 kb/s.

--bytes-per-send *value*

Specify the average size in bytes of each data provision. The exact value depends on sections and packets sizes. Default: 500 bytes.

--channel-id *value*

This option sets the DVB SimulCrypt parameter *data_channel_id*. Default: 1.

-c *value*
--client-id *value*

This option sets the DVB SimulCrypt parameter *client_id*. Default: 0.

For EMM injection, the most signification 16 bits shall be the *CA_system_id* of the corresponding CAS.

--cycles *value*

Inject the sections from the input files the specified number of times. By default, inject sections indefinitely.

-d *value*
--data-id *value*

This option sets the DVB SimulCrypt parameter *data_id*. Default: 0.

--emm-max-table-id *value*

Specify the maximum *table_id* of the automatically generated fake EMM's. The default is `0x8F`.

When generating fake EMM's, the table ids are cycled from the minimum to the maximum value.

--emm-min-table-id *value*

Specify the minimum table id of the automatically generated fake EMM's. The default is `0x82`.

--emm-size *value*

Specify the size in bytes of the automatically generated fake EMM's. The default is 100 bytes.

--emmg-mux-version *value*

Specify the version of the EMMG/PDG ⇔ MUX DVB SimulCrypt protocol.

Valid values are 1 to 5. The default is 2.

-i
--ignore-allocated

Ignore the allocated bandwidth as returned by the MUX. Continue to send data at the planned bandwidth, even if it is higher than the allocated bandwidth.

--log-data[=*level*]

Same as `--log-protocol` but applies to *data_provision* messages only.

To debug the session management without being flooded by data messages, use `--log-protocol=info --log-data=debug`.

--log-protocol[=*level*]

Log all EMMG/PDG ⇔ MUX protocol messages using the specified level. If the option is not present, the messages are logged at debug level only. If the option is present without value, the messages are logged at info level. A level can be a numerical debug level or any of the following: `fatal`, `severe`, `error`, `warning`, `info`, `verbose`, `debug`.

--max-bytes *value*

Stop after sending the specified number of bytes. By default, send data indefinitely.

-m *address:port*
--mux *address:port*

Specify the IP address (or host name) and TCP port of the MUX.

This is a required parameter, there is no default.

--requested-bandwidth *value*

This option sets the DVB SimulCrypt parameter *bandwidth* in the *stream_BW_request* message. The value is in kilobits per second. The default is the value of the `--bandwidth` option.

Specifying distinct values for `--bandwidth` and `--requested-bandwidth` can be used for testing the behaviour of a MUX.

-s
--section-mode

Send EMM's or data in section format. This option sets the DVB SimulCrypt parameter *section_TSpkt_flag* to zero. By default, EMM's and data are sent in TS packet format.

--stream-id *value*

This option sets the DVB SimulCrypt parameter *data_stream_id*. Default: 1.

-t *value*
--type *value*

This option sets the DVB SimulCrypt parameter *data_type*. Default: 0 (EMM). In addition to integer values, the following names can be used: `emm` (0), `private-data` (1) and `ecm` (2).

-u *[address:]port*
--udp *[address:]port*

Specify that the *data_provision* messages shall be sent using UDP.

By default, the *data_provision* messages are sent over TCP using the same TCP connection as the management commands.

If the IP address (or host name) is not specified, use the same IP address as the `--mux` option. The port number is required, even if it is the same as the TCP port.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N*]

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose

Produce verbose messages.

--version

Display the version number.

tsfixcc

Fix Continuity Counters

This utility fixes errors in the continuity counters (CC) in a transport stream file. If packets are missing (non continuous CC), the CC in all subsequent packets in the affected PID's are modified to remove the discontinuity.

If the file needs to be repeatedly played, tsfixcc can also add empty packets at the end of the file to fill the discontinuities between the end and the beginning of the file when the playback wraps to the beginning.

Usage

```
tsfixcc [options] file
```

File

MPEG transport stream. Must be a binary stream of 188-byte packets. This file must be a regular file (cannot be a pipe). It is open in read/write mode and is directly updated.

Options

-c
--circular

Enforce continuity when the file is played repeatedly. Add empty packets, if necessary, on each PID so that the continuity is preserved between end and beginning of file.

Note, however, that this method is not compliant with the MPEG-2 Transport Stream standard as defined in [1]. The standard specifies that the continuity counter shall not be incremented on packets without payload.

-n
--noaction

Display what should be performed but do not modify the file.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N*]

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose

Produce verbose messages.

--version

Display the version number.

tsftrunc

Transport Stream File Truncation

This utility truncates a captured transport stream file to remove trailing incomplete packets.

See also the utility *tsresync* for a more powerful way to recover corrupted transport stream files.

Usage

```
tsftrunc [options] file ...
```

Files

MPEG transport stream files. They must be binary streams of 188-byte packets. The files must be regular files (cannot be pipes). They are open in read/write mode and are directly updated.

Options

-b *value*
--byte *value*

Truncate the file at the next packet boundary after the specified size in bytes. Mutually exclusive with `--packet`.

-n
--noaction

Do not perform truncation, check mode only.

-p *value*
--packet *value*

Index of first packet to truncate. If unspecified, all complete packets are kept in the file. Extraneous bytes at end of file (after last multiple of 188 bytes) are truncated.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N***]**

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose

Produce verbose messages.

--version

Display the version number.

tsgenecm

Generate one ECM using any DVB SimulCrypt compliant ECMG

This command connects to a DVB SimulCrypt compliant ECMG and requests the generation of one ECM.

Restriction: The target ECMG shall support current or current/next control words in ECM, meaning `CW_per_msg` = `1` or `2` and `lead_CW` = `0` or `1`.

Usage

```
tsgenecm [options] output-file
```

Output file

Name of the binary output section file which receives the generated ECM.

Options

-a *value*
--access-criteria *value*

Specifies the access criteria for the service as sent to the ECMG. The value must be a suite of hexadecimal digits.

--channel-id *value*

Specifies the DVB SimulCrypt *ECM_channel_id* for the ECMG (default: 1).

-d *seconds*
--cp-duration *seconds*

Specifies the crypto-period duration in seconds (default: 10 seconds).

--cp-number *value*

Crypto-period number (default: 0).

-c *value*
--cw-current *value*

Current control word (required). The value must be a suite of hexadecimal digits.

-n *value*
--cw-next *value*

Next control word (optional). The value must be a suite of hexadecimal digits.

-i *value*
--ecm-id *value*

Specifies the DVB SimulCrypt *ECM_id* for the ECMG (default: 1).

-e *host:port*
--ecmg *host:port*

Specify an ECM Generator host name (or IP address) and TCP port.

-v *value*
--ecmg-scs-version *value*

Specifies the version of the ECMG <=> SCS DVB SimulCrypt protocol. Valid values are 2 and 3. The default is 2.

--log-data[=*level*]

Same as `--log-protocol` but applies to *CW_provision* and *ECM_response* messages only.

To debug the session management without being flooded by data messages, use `--log-protocol=info --log-data=debug`.

--log-protocol[=*level*]

Log all ECMG ⇔ SCS protocol messages using the specified level. If the option is not present, the messages are logged at debug level only. If the option is present without value, the messages are

logged at info level. A level can be a numerical debug level or any of the following: `fatal`, `severe`, `error`, `warning`, `info`, `verbose`, `debug`.

`--stream-id` *`value`*

Specifies the DVB SimulCrypt *ECM_stream_id* for the ECMG (default: 1).

`-s` *`value`*
`--super-cas-id` *`value`*

Specify the DVB SimulCrypt *Super_CAS_Id*. This is required when `--ecmg` is specified.

Generic common command options

The following options are implicitly defined in all commands.

`--debug[=`*`N`***`]`**

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

`--help`

Display command help text.

`-v`
`--verbose`

Produce verbose messages.

`--version`

Display the version number.

tshides

List HiDes modulator devices

This utility lists HiDes modulator devices and their characteristics.

Usage

```
tshides [options]
```

Options

-a *value*
--adapter *value*

Specify the HiDes adapter number to list. By default, list all HiDes devices.

Use `--adapter` or `--device` but not both.

-b *value*
--bandwidth *value*

Bandwidth in MHz with `--gain-range`.

The value must be one of 5, 6, 7, 8 (MHz). The default is 8 MHz.

-c
--count

Only display the number of devices, not their names or characteristics.

-d "*name*"
--device "*name*"

Specify the HiDes device name to list. By default, list all HiDes devices.

Use `--adapter` or `--device` but not both.

-f *value*
--frequency *value*

Frequency, in Hz, of the output carrier with `--gain-range`.

The default is the first UHF channel.

-g
--gain-range

Display the allowed range of output gain for the specified device.

Usually, the allowed range of gain depends on the frequency and the bandwidth. This is why the gain range is not displayed with the other characteristics. Use the options `--frequency` and `--bandwidth` to display the corresponding gain range.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N***]**

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose

Produce verbose messages.

--version

Display the version number.

tslsdvb

List DVB Receiver Devices

This utility lists the DVB receiver devices (DVB-S, DVB-C, DVB-T) in the system with their characteristics.

Usage

```
tslsdvb [options]
```

Options

-a *N*
--adapter *N*

Specify the *N*th DVB adapter in the system, the first index being zero. This option can be used instead of device name.

On Linux systems, this means `/dev/dvb/adapterN`.

-d "*name*"
--device-name "*name*"

Specify the name of the DVB receiver device to use. The syntax of the device name depends on the operating system. See section 6.1.3, page 285, for more details on DVB receiver devices naming.

By default, when no device name or adapter is specified, list all available receiver devices.

Windows-specific options:

-t *name*
--test *name*

Run a specific DirectShow test. Produce a very verbose output, for debug only. The names of the available tests are listed below.

`none`	Do not run any test. This is the default.
`enumerate-devices`	Enumerate all DirectShow devices which are used with DVB tuners. This test is useful to detect all devices which may not be recognized as valid tuners by TSDuck.
`tuning-spaces`	List all DirectShow tuning spaces which are installed in the system and their compatibility with the various network providers.
`bda-tuners`	List all BDA tuners and their compatibility with the various predefined "network provider" filters.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N***]**

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose

Produce verbose messages.

--version

Display the version number.

tsp

Transport Stream Processor

The transport stream processor is a general-purpose packet processing framework.

It receives an MPEG Transport Stream from a user-specified input plugin, applies MPEG packet processing through several user-specified packet processor plugins and sends the processed stream to a user-specified output plugin.

All input, processors and output plugins are shared libraries (`.so` files on Linux, `.dll` files on Windows).

The following figure illustrates the structure of a tsp process using three packet processing plugins.

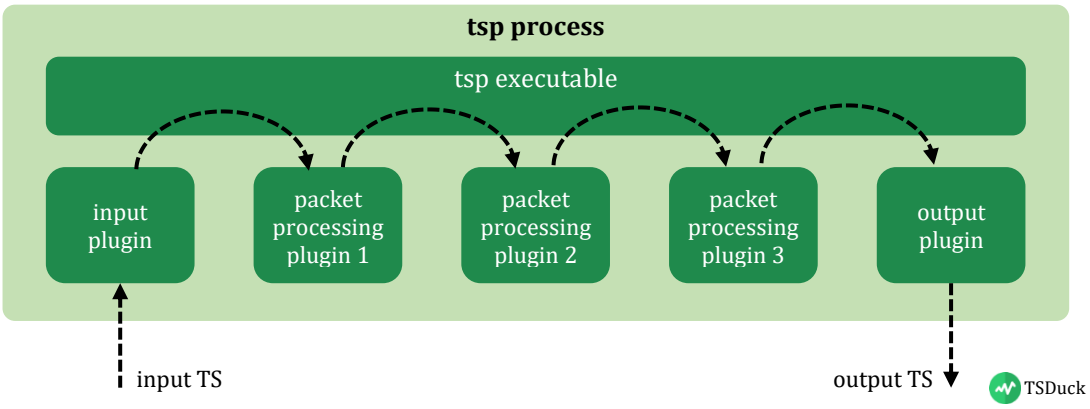

Figure 1: Transport stream processor diagram

This section describes the general syntax and usage of the tsp command. All plugins are documented in details, in alphabetical order, in chapter 4, page 92. The section 5.2 gives a few examples of tsp commands, both simple and complex examples.

Usage

The general syntax of the *tsp* command is the following:

```
tsp [tsp-options] \
    [-I input-name [input-options]] \
    [-P processor-name [processor-options]] ... \
    [-O output-name [output-options]]
```

All *tsp-options* must be placed on the command line before the input, packet processing and output plugin specifications. There must be at most one input and one output plugin. There may be any number of packet processing plugins. On the command line, the order of the packet processing plugins is significant: the TS packets are passed from one processor to the other in this order.

Offline and real-time defaults

There are two main classes of usage for *tsp*, offline and real-time processing. Offline processing works on static data such as transport stream files without specific timing constraints. Real-time processing applies to streaming devices such as tuners, Dektec devices or IP streams.

In the *tsp* command and in many plugins, some command line options affect tuning and performances. Roughly, we have to find a balance between throughput and latency.

- To get a higher throughput, we must minimize the data copy and thread context switching operations. This is achieved using larger buffer sizes and letting plugins work on larger amounts of TS packets. This requires less CPU and provides better overall performances. But this also has the side effect of increasing the latency.
- To get a lower latency, we must basically do the opposite: work on smaller data chunks, pass data faster (more frequently) from plugin to plugin. The drawback is an increase of CPU requirement.

There is no unique choice. When working on offline files, increasing the throughput and reducing the CPU load is the right choice. But for streaming and real-time processing, reducing the latency is the priority.

To optimize the offline or real-time processing, many tuning options can be adjusted. While fine tuning is sometimes useful, the user mainly needs two sets of default options: offline or real-time.

By default, *tsp* and all plugins use the offline defaults, the tuning options which give good performances at the expense of a higher latency.

The real-time defaults are used without having to specify all individual options in two cases:

- The option `--realtime` is specified in the *tsp* command line.
- At least one plugin in the chain is designed to work in real-time.

In these two cases, *tsp* and all plugins use their real-time defaults (unless, of course, options are individually set).

The second condition is an intrinsic property of a plugin. Examples of "real-time" plugins include *dvb*, *dektec*, *ip*, *play* or *regulate*. These plugins are somehow designed to work on real-time streams. Their simple presence in the *tsp* command is sufficient to trigger the use of real-time defaults for all plugins. It is still possible to force the use of offline defaults using the *tsp* option `--realtime=off`, even if a real-time plugin is present.

Rendering speed and transmission speed

With *tsp*, a stream has a *rendering* speed (the speed of the audio / video) and a *transmission* speed (the speed at which packets go through *tsp*).

As a general rule, the work *bitrate* refers to the rendering speed. So, when a plugin inserts data with a *bitrate of 100 kb/s* for instance, this means that the data will be received at this bitrate when the transport stream is played in real time (independently of the file processing speed, if the data insertion was previously performed on an offline file).

It is important to understand the differences between the two. Real-time streams, from broadcast or multicast, have identical transmission and rendering speeds because they are transmitted to watch TV. Files, on the other hand, have a very high transmission speed, typically the I/O speed of the disk, maybe 1 Gb/s or more on SSD.

Some plugins explicitly manipulate the rendering or transmission speed. The plugin *pcrbitrate*, for instance, is designed to evaluate the rendering speed based on embedded time stamps in the stream. The plugin *regulate*, on the other hand, is designed to alter the transmission speed.

Let's review some examples where these plugins should be used.

Consider that you have recorded a 6 Mb/s single program transport stream and you want to send it through UDP/IP to a remote media player. Using "*tsp -I file*", you read it and send it to *"-O ip*". The effective reading speed of the file will be 500 M/b for instance. So, on a gigabit network, you send a 6 Mb/s video stream at 1 Gb/s, 166 times faster as it should be. Thus, a 15 minutes video is received in 5 seconds and the player displays almost nothing. In this case, you must use the plugin *regulate* between "*-I file*" and "*-O ip*". The plugin acts as a bottleneck and lets packets flow out at 6 Mb/s only.

But, when the source has the same transmission and rendering speeds (DVB tuner, IP source), the plugin *regulate* is useless. At best, it does nothing. At worst, it introduces undesirable artifacts.

There are also cases where the transmission speed regulation is done automatically. If the media player is a local application and is started using "*-O play*", *tsp* communicates with the player through a pipe. A pipe is a self-regulated communication mechanism. So, even if the input is a disk file with a high reading speed, using *regulate* is not necessary because the same role is played here by the pipe. The difference with the previous example is that UDP/IP is not a regulated communication channel, unlike pipes and TCP/IP.

Bitrate propagation

At any point in the chain, all plugin have some knowledge of the transport stream bitrate, or *rendering* speed. Some plugins use that bitrate information, some others don't. The plugin *regulate* is a typical example. It uses the *rendering* speed as an information to lower the *transmission* speed.

As a general rule, *tsp* collects the input bitrate, either from the input plugin itself which extracts the bitrate from a hardware input device (this is the case for ASI cards for instance) or, if the input plugin is not able to report a bitrate, *tsp* automatically analyzes PCR's at the output of the input plugin and computes the corresponding bitrate.

Then, the bitrate is transmitted from plugin to plugin.

Some plugins may inadvertently propagate incorrect bitrates while some plugins may force a (correct) recomputation of the bitrate. To illustrate the first case, consider "*-I file ... -P zap ...*" using sample bitrate values. You read a complete 36 Mb/s input and *tsp* evaluates this bitrate. Then, "*-P zap*" extracts a 4 Mb/s service and removes everything else. But it does not recompute the transport stream bitrate. So, the propagated bitrate information is still 36 Mb/s. If this information is not used downstream in other plugins, we don't care. But if we use the bitrate information in "*-P regulate -O ip ...*", we will regulate at 36 Mb/s a stream which should be played at 4 Mb/s. This is why, in specific situations like this, we need to recompute the bitrate using "*-P pcrbitrate*" before "*-P regulate*".

Modifying, inserting and deleting packets

In the complete chain of processing, between the input and the output plugin, each TS packet goes through[1] all packet processing plugins, one after the other, in the order of the command line.

A packet processing plugin may read, modify or delete existing packets. **But it cannot add new packets**.

Roughly, each packet processing plugin has one of the following functions (or sometimes a combination of them):

- Analysis (read packets).
- Modification (modify existing packets).
- Removal (delete packets from the stream).
- Data injection (add new packets).

The last case cannot be directly implemented. To achieve data injection, a plugin usually "steals stuffing". Each time a new TS packet needs to be injected, a plugin waits for the next *null packet* (i.e. a packet in PID `0x1FFF`) and replaces this null packet with the new packet to insert.

Consequently, the original amount of stuffing and its distribution in a stream directly influences the insertion profile of new packets. Specifically, it is not possible to add more data than the stuffing bitrate. Moreover, precise timing cannot be always achieved. When data need to be inserted at a given bitrate, the plugin tries to reach this average bitrate (provided that there is enough stuffing) but cannot guarantee a precise constant inter-packet distance.

In broadcast streams, where the modulation parameters impose a fixed bitrate, there is always some stuffing. With variable bitrate, simple-program transport streams for IP, there can be no stuffing at all.

What are the options when the original amount of stuffing is not sufficient to insert the required data? It depends on the requirements on the stream.

If the stream is targeted for broadcast, with a given target bitrate which cannot be changed, there is no other solution than removing existing data to make room for the new data. Some plugins such as *filter* or *svremove* delete individual PID's or complete services. By default, the deleted packets are simply removed from the stream. But these plugins also have a `--stuffing` option which replaces deleted packets by stuffing instead of removing them. Thus, you can increase the stuffing bitrate without altering the global transport stream bitrate.

If there is no requirement on the global bitrate, it is possible to insert artificial stuffing at input level using the global *tsp* option `--add-input-stuffing`. The option adds a given number of null packets after a given number of input packets (for instance, add 1 null packet every 15 input packets). The parameters influence the amount and distribution of the artificial stuffing. Do not be afraid of inserting too much stuffing. It is always possible to remove the stuffing in excess using "*-P filter -n -p 0x1FFF*" at the end of the chain, after all injection plugins.

---

[1] In fact, a TS packet never moves. It is loaded in a large circular buffer and stays there. Each plugin uses a sliding window over the circular buffer and inspects or modifies packets without moving them.

Merging and forking

As indicated above, *tsp* processes one single transport stream. However, specific plugins such as *merge* and *fork* respectively combine and duplicate transport streams. They are designed to route transport streams from and to other applications. When the "other" application is another instance of *tsp*, we can create complex processing graphs.

This is illustrated in the diagram below.

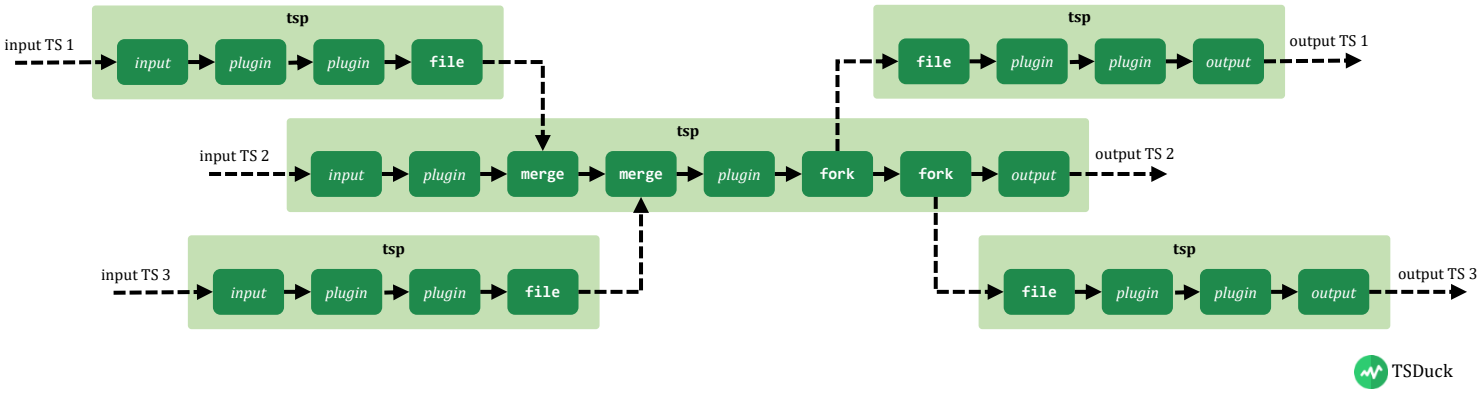

Figure 2: Merging and forking transport streams

Packet labelling

Transport streams packets may receive one or more *label* from any packet processing plugin. A label is an integer value from 0 to 31, inclusive. A label remains attached to the packet all along the chain, from plugin to plugin. Later, it is possible to select packets with a label value or invoke a specific plugin only on packets having a given label.

The plugin *filter* has an option named `--set-label` to assign a label to the selected packets. Note that, with this option, the plugin *filter* does not drop unselected packets; it keeps all packets but assigns the specified label to the seleted packets.

All packet processing plugins accept the option `--only-label` which selects only the packets with a given label. Thus, only the packets with that label pass through the plugin. All other packets, without that label, are directly passed to the next plugin in the chain.

The following example illustrates the usage of labels. The first three plugins select different kinds of packets and assign a label value depending on the kind of packet. These *filter* plugins do not drop any packet, they just assign labels to some of them. Later, three other plugins are applied only to one of these labels. In this example, we consequently count packets with unit start indicator and scrambling control value 2 and 3, respectively.

```
tsp -I ... \
    -P filter --unit-start --set-label 2 \
    -P filter --scrambling 2 --set-label 10 \
    -P filter --scrambling 3 --set-label 11 \
    -P count --only-label 2 --total --tag unit \
    -P count --only-label 10 --total --tag scr2 \
    -P count --only-label 11 --total --tag scr3 \
    -O ...

* count: unit: total: counted 5,311 packets out of 5,311
* count: scr2: total: counted 8,378 packets out of 8,378
* count: scr3: total: counted 7,439 packets out of 7,439
```

Global tsp options

-a *nullpkt/inpkt*
--add-input-stuffing *nullpkt/inpkt*

Specify that *nullpkt* null TS packets must be automatically inserted after every *inpkt* input TS packets. Both *nullpkt* and *inpkt* must be non-zero integer values. This option is useful to artificially increase the input bitrate by adding stuffing.

Example: the option "`-a 14/24`" adds 14 null packets every 24 input packets, effectively turning a 24 Mb/s input stream (terrestrial) into a 38 Mb/s stream (satellite).

`--add-start-stuffing` *count*

Specify that *count* null TS packets must be automatically inserted at the start of the processing, before the first packet coming from the input plugin.

`--add-stop-stuffing` *count*

Specify that *count* null TS packets must be automatically inserted at the end of the processing, after the last packet coming from the input plugin.

`-b` *value*
`--bitrate` *value*

Specify the input bitrate, in bits/seconds. By default, the input bitrate is provided by the input plugin or by analysis of the PCR's at the beginning of the input stream. If no or not enough PCR are found, the DTS from video PID's are used.

Use option `--bitrate` when you know precisely the input bitrate and you do not trust the input device, the PCR's or the DTS.

See also the plugin *pcrbitrate* for permanent recomputation of the bitrate based on PCR's or DTS.

`--bitrate-adjust-interval` *value*

Specify the interval in seconds between bitrate adjustments, ie. when the output bitrate is adjusted to the input one. The default is 5 seconds. Some output processors ignore this setting. Typically, ASI or modulator devices use it, while file devices ignore it. This option is ignored if `--bitrate` is specified.

`--buffer-size-mb` *value*

Specify the buffer size in mega-bytes. This is the size of the buffer between the input and output devices. The default is 16 MB. Increasing the buffer size may improve the performance at the expense of increasing the overall latency (implicit time-shifting).

See also the options `--max-input-packets` and `--max-flushed-packets` to adjust the latency without modifying the global buffer size.

`-d[`*N***`]`**
`--debug[=`*N***`]`**

Produce debug output. Specify an optional debug level *N*. Do not use in normal operation.

Without this option, no debug output is produced. When the option is specified but not the level *N*, the default debug level is 1, that is to say a reasonable amount of information. The higher the debug level is, the more output is produced.

The debug setting is automatically transmitted to all plugins.

`-i`
`--ignore-joint-termination`

Ignore all `--joint-termination` options in plugins.

Some plugins have termination conditions. For instance, the plugin `until` passes packets until some specified condition, the plugins `mux` and `inject` may terminate `tsp` after completing the data insertion, etc.

A plugin can decide to terminate `tsp` on its own. The termination is unconditional, regardless of the state of the other plugins. Thus, if several plugins have termination conditions, `tsp` stops when the first plugin decides to terminate. In other words, there is an "*or*" operator between the various termination conditions.

The idea behind *joint termination* is to terminate `tsp` when several plugins have jointly terminated their processing. If several plugins have a "*joint termination*" condition (usually using the option `--joint-termination`), `tsp` stops when the last plugin triggers the joint termination condition. In other words, there is an "*and*" operator between the various joint termination conditions.

The tsp-option `--ignore-joint-termination` disables the termination of `tsp` when all plugins have reached their joint termination condition. The plugins continue to pass packets as if some additional joint termination condition was still pending.

-l
--list-processors

List all available processors.

--log-message-count *value*

Specify the maximum number of buffered log messages. Log messages are displayed asynchronously in a low priority thread. This value specifies the maximum number of buffered log messages in memory, before being displayed. When too many messages are logged in a short period of time, while plugins use all CPU power, the low-priority log thread has no resource. It cannot display yet the buffered messages and extra messages are dropped. Increase this value if you think that too many messages are dropped. The default is 512 messages.

See also the option `--synchronous-log`.

--max-flushed-packets *value*

Specify the maximum number of packets to be processed before flushing them to the next processor or the output. When the processing time is high and some packets are lost, try decreasing this value.

The offline default is 10 000 packets. The real-time default is 1000 packets.

--max-input-packets *value*

Specify the maximum number of packets to be received at a time from the input plugin.

By default, in offline mode, tsp reads as many packets as it can, depending on the free space in the buffer. The real-time default is 1000 packets.

-m
--monitor

Continuously monitor the system resources which are used by tsp. This includes CPU load, virtual memory usage. Useful to verify the stability of the application or benchmarking the packet processing performance.

-r[*keyword*]
--realtime[*keyword*]

Specifies if *tsp* and all plugins should use default values for real-time or offline processing.

By default, if any plugin prefers real-time, the real-time defaults are used. If no plugin prefers real-time, the offline default are used.

If `-r` or `--realtime` is used alone, the real-time defaults are enforced. The explicit values '`no`', '`false`', '`off`' are used to enforce the offline defaults and the explicit values '`yes`', '`true`', '`on`' are used to enforce the real-time defaults.

-s
--synchronous-log

With this option, each logged message is guaranteed to be displayed, synchronously, without any loss of message. The downside is that a plugin thread may be blocked for a short while when too many messages are logged. This option shall be used when all log messages are needed and the source and destination are not live streams (files for instance). This option is not recommended for live streams, when the responsiveness of the application is more important than the logged messages.

-t
--timed-log

Each logged message contains a time stamp.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N*]

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose

Produce verbose messages.

--version

Display the version number.

Plugin activation options

-I *name*
--input *name*

Designate the shared library plugin for packet input. By default, read packets from standard input.

-O *name*
--output *name*

Designate the shared library plugin for packet output. By default, write packets to standard output.

-P *name*
--processor *name*

Designate a shared library plugin for packet processing. Several packet processors are allowed. Each packet is successively processed by each processor, in the order of the command line. By default, there is no processor and the packets are directly passed from the input to the output.

The specified plugin *name* is used to locate a shared library for the plugin (`.so` file on Linux, `.dll` file on Windows). Usually, all plugins files are in the same directory as the `tsp` executable. But, more generally, a plugin can be designated in a number of ways, in the following order. When a method fails, the next one is attempted.

- If the plugin name is a complete path name, with a directory, this path name is used.
- Without directory in the plugin name, a list of directories is searched:
  - If the environment `TSPLUGINS_PATH` is defined, a list of directories is parsed. Directories are separated by a semicolon '`;`' on Windows and a colon '`:`' on UNIX systems.
  - The same directory as the `tsp` executable file is used as last choice.
  - In each of these directories, the file named `tsplugin_`*name*`.so` or `.dll` is searched.
  - If not found, the file *name* and then *name*`.so` or `.dll` is searched.
- If still not found, the standard algorithm of the operating system is applied to locate the shared library file, using the specified name (on Linux, see the man page of `dlopen(3)` for more details).

The *input-options*, *processor-options* and *output-options*, as specified in the general syntax of the tsp command, are specific to their corresponding plugin. All available plugins are documented in chapter 4, page 92.

All plugins accept the following common options:

--help

The plugin displays its syntax and exits.

This means that the following type of command can be used to display the help text for a specific plugin:

```
tsp {-I|-O|-P} name --help
```

tspacketize

Packetize PSI/SI Tables in a Transport Stream PID

This utility packetizes PSI/SI tables in a transport stream PID.

Usage

```
tspacketize [options] [input-file[=rate] ...]
```

Parameters

input-file[=*rate*]

Binary or XML files containing one or more sections or tables. By default, files with a name ending in `.xml` are XML and files with a name ending in `.bin` are binary. For other file names, explicitly specify `--binary` or `--xml`.

If the file name is omitted, the standard input is used (binary by default, specify `--xml` otherwise).

If different repetition rates are required for different files, a parameter can be "*filename*=*value*" where *value* is the repetition rate in milliseconds for all sections in that file. For repetition rates to be effective, the bitrate of the target PID must be specified, see option `-b` or `--bitrate`.

Options

--binary

Specify that all input files are binary, regardless of their file name.

-b *value*
--bitrate *value*

Specifies the bitrate (in bits/second) of the target PID. This information is used to schedule sections in the output list of packets when specific bitrates are specified for sections. When no specific bitrate is specified for any input file, this option is ignored.

-c
--continuous

Continuous packetization. By default, generate one cycle of sections.

-f
--force-crc

Force recomputation of CRC32 in long sections. Ignore the CRC32 values in the input files.

By default, the CRC32 of every section is verified and sections with wrong CRC32 are rejected.

-o *file-name*
--output *file-name*

Output file name for TS packets. By default, use standard output.

-p *value*
--pid *value*

PID of the output TS packets. This is a required parameter, there is no default value.

-s
--stuffing

Insert stuffing at end of each section, up to the next TS packet boundary. By default, sections are packed and start in the middle of a TS packet, after the previous section. Note, however, that section headers are never scattered over a packet boundary.

--xml

Specify that all input files are XML, regardless of their file name.

Generic common command options

The following options are implicitly defined in all commands.

`--debug[=`*`N`*`]`

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

`--help`

Display command help text.

`-v`
`--verbose`

Produce verbose messages.

`--version`

Display the version number.

tspsi

Dump All PSI Tables

This utility extracts all PSI tables (PAT, CAT, PMT, NIT, BAT, SDT[2]) from a transport stream. The output is rather primitive but it exactly exhibits the structure of tables, sections and descriptors.

Usage

```
tspsi [options] [input-file]
```

Input file

MPEG transport stream, either a capture file or a pipe from a live stream. Must be a binary stream of 188-byte packets. If omited, standard input is used.

Options

-a
--all-versions

Display all versions of PSI tables (need to read the complete transport stream). By default, display only the first version of each PSI table and stop when all expected PSI are extracted.

--cat-only

Display only the CAT, ignore other PSI tables.

-c
--clear

Indicate that this is a clear transport stream, without conditional access information. Useful to avoid further reading the transport stream, waiting for a non-existent CAT.

-d
--dump

Dump all PSI sections.

--exclude-current

Exclude PSI tables with "current" indicator. This is rarely necessary.

See also `--include-next`.

--include-next

Include PSI tables with "next" indicator. By default, they are excluded.

--no-pager

Do not send output through a pager process. By default, if the output device is a terminal, the output is paged. See 3.1.4 for more details.

-o *file-name*
--output-file *file-name*

File name for text output.

Tables and sections formatting options

--atsc

Assume that the transport stream is an ATSC one.

ATSC streams are normally automatically detected from their signalization. This option is only useful when ATSC-related stuff are found in the TS before the first ATSC-specific table. For instance, when a PMT with ATSC-specific descriptors is found before the first ATSC MGT or VCT.

[2] I know, BAT and SDT are SI, not PSI ☺

-c
--c-style

Same as `--raw-dump` (no interpretation of section) but dump the bytes in C-language style, e.g. "`0x01, 0x02,`" instead of "`01 02`". Useful to include this output as data in a C source file.

--default-charset *name*

Default character set to use when interpreting DVB strings without explicit character table code. According to DVB standard ETSI EN 300 468, the default DVB character set is ISO-6937. However, some bogus signalization may assume that the default character set is different, typically the usual local character table for the region. This option forces a non-standard character table. The available table names are: ISO-6937, ISO-8859-1, ISO-8859-10, ISO-8859-11, ISO-8859-13, ISO-8859-14, ISO-8859-15, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, UNICODE, UTF-8.

--default-pds *value*

Default private data specifier (PDS). This option is meaningful only when the signalization is incorrect, when private descriptors appear in tables without a preceding *private_data_specifier_descriptor*. The specified value is used as private data specifier to interpret private descriptors. The PDS value can be an integer or one of (not case-sensitive): "Nagra", "TPS", "EACEM"," EICTA", "Logiways", "CanalPlus", "Eutelsat".

--europe

A synonym for '`--default-charset ISO-8859-15`'. This is a handy shortcut for commonly incorrect signalization on some European satellites. In that signalization, the character encoding is ISO-8859-15, the most common encoding for Latin & Western Europe languages. However, this is not the default DVB character set and it should be properly specified in all strings, which is not the case with some operators. Using this option, all DVB strings without explicit table code are assumed to use ISO-8859-15 instead of the standard ISO-6937 encoding.

--nested-tlv[=*min-size*]

With option `--tlv`, try to interpret the value field of each TLV record as another TLV area. If the *min-size* value is specified, the nested TLV interpretation is performed only on value fields larger than this size. The syntax of the nested TLV is the same as the enclosing TLV.

-r
--raw-dump

Raw dump of section, no interpretation.

--tlv *syntax*

For sections of unknown types, this option specifies how to interpret some parts of the section payload as TLV records. Several `--tlv` options are allowed, each one describes a part of the section payload.

Each syntax string has the form "*start,size,tagSize,lengthSize,order*". The *start* and *size* fields define the offset and size of the TLV area in the section payload. If the *size* field is "`auto`", the TLV extends up to the end of the section. If the start field is "`auto`", the longest TLV area in the section payload will be used. The fields *tagSize* and *lengthSize* indicate the size in bytes of the Tag and Length fields in the TLV structure. The field *order* must be either "`msb`" or "`lsb`" and indicates the byte order of the Tag and Length fields.

All fields are optional. The default values are "`auto,auto,1,1,msb`".

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N*]

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose
: Produce verbose messages.

--version
: Display the version number.

tsresync

Resynchronize Corrupted Transport Stream Files

This utility resynchronizes a corrupted transport stream file.

Usage

```
tsresync [options] [input-file]
```

Input file

MPEG transport stream, either a capture file or a pipe from a live stream. Must be a binary stream of transport stream packets, with various encapsulation or possible corruptions.

If omited, the standard input is used.

Options

-c
--continue

Continue re-resynchronizing after loss of synchronization. By default, stop after first packet not starting with `0x47`.

-h *value*
--header-size *value*

When used with `--packet-size`, specifies the size of extra data preceeding each packet in the input file. The default is zero.

-k
--keep

Keep TS packet size from input to output file. By default, strip extra data and reduce packets to 188 bytes. See option `--packet-size` for a description of supported input packet sizes.

-m *value*
--min-contiguous *value*

Minimum size containing contiguous valid packets to consider a slice of input file as containing actual packets (default: 512 kB).

-o *file-name*
--output *file-name*

Output file name (standard output by default).

-p *value*
--packet-size *value*

Expected TS packet size in bytes. By default, try:

- 188-byte (standard)
- 204-byte (trailing 16-byte Reed-Solomon outer FEC)
- 192-byte (leading 4-byte timestamp in M2TS/Blu-ray disc files).

If the input file contains any other type of packet encapsulation, use options `--packet-size` and `--header-size`.

-s *value*
--sync-size *value*

Number of initial bytes to analyze to find start of packet synchronization (default: 1 MB).

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N***]**

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose

Produce verbose messages.

--version

Display the version number.

tsscan

DVB Network Scanning

This utility scans frequencies, transport streams and services in a DVB network.

Usage

```
tsscan [options]
```

Tuner device options and tuning parameters

All options from the *dvb* input plugin are also available to *tsscan*. See page 137 for the list of options.

If no tuner device is specified, the first DVB receiver is used.

If tuning parameters are present (frequency or channel reference), the NIT is read on the specified frequency and a full scan of the corresponding network is performed.

By default, without specific frequency, an UHF-band scanning is performed (see option `--uhf-band`).

Scanning options

`--best-quality`

With UHF/VHF-band scanning, for each channel, use the offset with the best signal quality. By default, use the average of lowest and highest offsets with required minimum quality and strength.

`--best-strength`

With UHF/VHF -band scanning, for each channel, use the offset with the best signal strength. By default, use the average of lowest and highest offsets with required minimum quality and strength.

`---first-channel` *value*

For UHF/VHF-band scanning, specify the first channel to scan (default: lowest channel in band).

`-first-offset` *value*

For UHF/VHF-band scanning, specify the first offset to scan on each channel. Note that `tsscan` may scan lower offsets. As long as some signal is found at a specified offset, `tsscan` continues to check up to 3 lower offsets below the "*first*" one. This means that if a signal is found at offset -2, offset -3 will be checked anyway, etc. up to offset -5.

`-g`
`--global-service-list`

Same as `--service-list` but display a global list of services at the end of scanning instead of per transport stream.

`--last-channel` *value*

For UHF/VHF -band scanning, specify the last channel to scan (default: highest channel in band).

`--last-offset` *value*

For UHF/VHF-band scanning, specify the last offset to scan on each channel. Note that `tsscan` may scan higher offsets. As long as some signal is found at a specified offset, `tsscan` continues to check up to 3 higher offsets above the "*last*" one. This means that if a signal is found at offset +2, offset +3 will be checked anyway, etc. up to offset +5.

`--min-quality` *value*

Minimum signal quality percentage. Frequencies with lower signal quality are ignored (default: 10%).

`--min-strength` *value*

Minimum signal strength percentage. Frequencies with lower signal strength are ignored (default: 10%).

-n
--no-offset

For UHF/VHF-band scanning, scan only the central frequency of each channel. This is now the default. Specify option `--use-offsets` to scan all offsets.

--psi-timeout *milliseconds*

Specifies the timeout, in milli-seconds, for PSI/SI table collection. Useful with `--service-list` or NIT-based scan. The default is 10 000 milli-seconds.

--save-channels *filename*

Save the description of all channels in the specified XML file. See Appendix B, page 293, for more details on channels configuration files.

If the file name is "-", use the default tuning configuration file.

See also option `--update-channels`.

-l
--service-list

Read SDT of each channel and display the list of services.

--show-modulation

Display modulation parameters.

Windows-specific note: With UHF band scanning, the actual modulation parameters of a transponder may not be available. This depends on the driver of the tuner. Most drivers do not report the correct values.

-u
--uhf-band

Perform DVB-T or ATSC UHF-band scanning. This is the default scanning method when no tuning parameter is given to read a NIT.

--update-channels *filename*

Update the description of all channels in the specified XML file. The content of each scanned transport stream is replaced in the file. If the file does not exist, it is created. See Appendix B, page 293, for more details on channels configuration files.

If the file name is "-", use the default tuning configuration file.

See also option `--save-channels`.

--use-offsets

For UHF/VHF-band scanning, do not scan only the central frequency of each channel. Also scan frequencies with offsets.

As an example, if a signal is transmitted at offset +1, the reception may be successful at offsets -1 to +3 (but not -2 and +4). With this option, `tsscan` checks all offsets and reports that the signal is at offset +1 (central point between offsets -1 and +3).

By default, `tsscan` reports that the signal is found at the central frequency of the channel (offset zero). This significantly speeds up the scanning process but does not provide any offset information.

--vhf-band

Perform DVB-T or ATSC VHF-band scanning.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N*]

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose

Produce verbose messages.

--version

Display the version number.

tssmartcard

Smart-Card Utility

This utility lists or resets the smart-card readers in the system.

Usage

```
tssmartcard [options] [reader-name]
```

Reader name

The optional `reader-name` parameter indicates the smart-card reader device name to list or reset.

By default, without any option or parameter, the command lists all smart-card reader devices in the system.

Options

-c
--cold-reset
: Perform a cold reset on the smart-card.

-e
--eject
: Eject the smart-card (if supported by the reader device).

-t *value*
--timeout *value*
: Timeout in milliseconds. The default is 1000 ms (1 second).

-w
--warm-reset
: Perform a warm reset on the smart-card.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N*]
: Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help
: Display command help text.

-v
--verbose
: Produce verbose messages.

--version
: Display the version number.

tsstuff

Add stuffing to a TS file to reach a target bitrate

This utility adds stuffing packets to a TS file to reach a target bitrate. Time stamps (PCR or DTS) are extracted from one *reference PID* in the input file and stuffing packets are added so that the time stamps are approximately synchronized with the TS target bitrate.

Usage

```
tsstuff [options] [input-file]
```

Input file

The input file is a TS file, typically with variable bitrate content. By default, the standard input is used.

Options

-b *value*
--bitrate *value*

Target constant bitrate of the output file. This is mandatory parameter, there is no default.

--buffer-size *value*

Input buffer size, in bytes. Must be large enough to always contain two time stamps in the reference PID. Default: 4,194,304 bytes (4 MB).

-d
--dts-based

Use Decoding Time Stamps (DTS) in the reference PID to evaluate the amount of stuffing to insert. The default is to use Program Clock References (PCR) instead of DTS.

-f *value*
--final-inter-packet *value*

Number of stuffing packets to add between input packets after the last time stamp (PCR or DTS). By default, use the same number as in the previous segment, between the last two time stamps.

-i *value*
--initial-inter-packet *value*

Number of stuffing packets to add between input packets before the first time stamp (PCR or DTS). By default, use the same number as in the first segment, between the first two time stamps.

-l *value*
--leading-packets *value*

Number of consecutive stuffing packets to add at the beginning of the output file, before the first input packet. The default is zero.

-m *value*
--min-interval *value*

Minimum interval, in milli-seconds, between two recomputations of the amount of stuffing to insert. This duration is based on time-stamps, not real time. The default is 100 ms.

-o *filename*
--output *filename*

Output file name (standard output by default). The output file is a TS file with the same packets as the input file with interspersed stuffing packets and a constant bitrate.

-r *value*
--reference-pid *value*

PID in which to collect time stamps (PCR or DTS) to use as reference for the insertion of stuffing packets. By default, use the first PID containing the specified type of time stamps (PCR or DTS).

-t *value*
--trailing-packets *value*

Number of consecutive stuffing packets to add at the end of the output file, after the last input packet. The default is zero.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N***]**

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose

Produce verbose messages.

--version

Display the version number.

tsswitch

Transport stream input source switch using remote control

This utility uses several transport stream inputs and one single output. One input is selected and passed to the output. Using either predefined policies or remote control, it is possible to switch back and forth between inputs.

All inputs and output are performed using external plugins. These plugins are the same as the plugins which are used by *tsp*.

Using the input plugins *file* or *fork*, it is possible to connect applications to some *tsswitch* input. One of these applications can be *tsp*, in which case it is possible to insert specific processing between the input plugin and the switch.

See a sample usage with a system diagram in section 5.1.7.

Cycling through input plugins

The list of input plugins is ordered by index on the command line, from 0 to n-1. By default, the input plugin 0 is started when the command starts. When a plugin terminates (end of input or error), the next one is started. When the last plugin terminates, the *tsswitch* command terminates.

Running all input plugins in sequence, from 0 to n-1, is called a *cycle*. By default, only one cycle is executed before *tsswitch* terminates. Using the option `--cycle`, it is possible to execute a given number of cycles. With the option `--infinite`, *tsswitch* runs endlessly.

With the option `--terminate`, *tsswitch* terminates when the current plugin terminates. In this case, without remote control, *tsswitch* only executes the first plugin. If the remote control was used to switch to another input, *tsswitch* terminates when the current plugin terminates, whichever it is.

Input switching modes

There are three different modes when switching from an input plugin to another one.

By default, only one input plugin is active at a time. When *tsswitch* starts, the first plugin is started. When an input switch is requested, the current plugin is first stopped. When the stop operation is complete, the next plugin is started. This mode is required when two plugins use the same input device such as a tuner. Since the device cannot be shared, it must be completely stopped and closed before being reused by the next plugin. This is the safest mode. The downside is that there could be a transmission hole in the output during the switch.

With option `--delayed-switch`, the switching operation is slightly different. The next plugin is started first. In the meantime, output packets continue to be fetched from the previous input plugin. When the next plugin starts to receive packets, the switch is performed: output packets are now read from the next plugin. Finally, the previous input plugin is stopped. This mode guarantees a smooth transition. However, the actual output switch is delayed until the next plugin is fully operational.

With option `--fast-switch`, all input plugins are started in parallel from the beginning and are never stopped. All input plugins continuously read packets and fill their buffer. The current plugin performs normal flow control with the output plugin, without packet loss. All other input plugins continuously overwrite their circular input buffer. When an input switch is requested, the output plugin immediately jumps into the next plugin buffer where the latest packets are already available. This mode guarantees a smooth and immediate switch. It is appropriate for live streams only.

Remote control

Using the option `--remote`, *tsswitch* listens to UDP datagrams on a given port. Each datagram contains one switch command. A command is an ASCII string. Any trailing control characters such as CR or LF is ignored.

The command string can be one of:

- An input index (e.g. “`0`”, “`1`”, “`2`”, etc.) Upon reception, *tsswitch* immediately switches to the selected input plugin.
- Strings “`next`” and “`previous`” (or “`prev`”) to switch to the next and previous input, respectively.

- Strings "`exit`" or "`quit`" to properly terminate *tsswitch*.
- Strings "`halt`" or "`abort`" to immediately abort the *tsswitch* process.

Note that the *bash*[3] shell provides an easy way to redirect output to an UDP message. The following sample commands send UDP messages on port 4444 to system 127.0.0.1 (the local host). This is the easiest way to use the *tsswitch* remote control.

```
echo >/dev/udp/127.0.0.1/4444 2
echo >/dev/udp/127.0.0.1/4444 next
echo >/dev/udp/127.0.0.1/4444 prev
echo >/dev/udp/127.0.0.1/4444 exit
```

Usage

The general syntax of the *tsswitch* command is the following:

```
tsswitch [tsswitch-options] \
         -I input-name [input-options] ... \
         [-O output-name [output-options]]
```

All *tsswitch-options* must be placed on the command line before the input and output plugin specifications. There must be at least one input plugin and at most one output plugin. The default output plugin is `file`, sending all packets to the standard output.

On the command line, the order of the input plugins is significant. They are indexed from 0 to n-1. This index value is used in the remote control protocol to select an input stream.

Plugin activation options

`-I` *name*
`--input` *name*

Designate the shared library plugin for packet input. There is no default. At least one input plugin shall be specified.

`-O` *name*
`--output` *name*

Designate the shared library plugin for packet output. By default, write packets to standard output.

All input and output plugins which are available for *tsp* can be used by *tsswitch*. See the description of the command *tsp* for the method to locate the plugin files.

General options

`-b` *value*
`--buffer-packets` *value*

Specify the size in TS packets of each input plugin buffer. The default is 512 packets.

`--max-input-packets` *value*

Specify the maximum number of TS packets to read at a time. This value may impact the switch response time. The default is 128 packets. The actual value is never more than half the `--buffer-packets` value.

`--max-output-packets` *value*

Specify the maximum number of TS packets to write at a time. The default is 128 packets.

Input cycles options

`-c` *value*
`--cycle` *value*

Specify how many times to repeat the cycle through all input plugins in sequence. By default, all input plugins are executed in sequence only once (`--cycle 1`). The options `--cycle`, `--infinite` and `--terminate` are mutually exclusive.

---

[3] This is a feature of *bash*, not a Linux feature. It is available on all platforms, including macOS or Cygwin.

--first-input *value*

Specify the index of the first input plugin to start. By default, the first plugin (index 0) is used.

-i
--infinite

Infinitely repeat the cycle through all input plugins in sequence.

-t
--terminate

Terminate execution when the current input plugin terminates.

Input modes options

-d
--delayed-switch

Perform delayed input switching. When switching from one input plugin to another one, the second plugin is started first. Packets from the first plugin continue to be output while the second plugin is starting. Then, after the second plugin starts to receive packets, the switch occurs: packets are now fetched from the second plugin. Finally, after the switch, the first plugin is stopped.

By default, the current input is first stopped and then the next one is started. Options `--delayed-switch` and `--fast-switch` are mutually exclusive.

-f
--fast-switch

Perform fast input switching. All input plugins are started at once and they continuously receive packets in parallel. Packets are dropped, except for the current input plugin. This option is typically used when all inputs are live streams on distinct devices (not the same DVB tuner for instance).

By default, only one input plugin is started at a time. When switching, the current input is first stopped and then the next one is started. Options `--delayed-switch` and `--fast-switch` are mutually exclusive.

-p *value*
--primary-input *value*

Specify the index of the input plugin which is considered as primary or preferred.

This input plugin is always started, never stopped, even without `--fast-switch`. When no packet is received on this plugin, the normal switching rules apply. However, as soon as packets are back on the primary input, the reception is immediately switched back to it.

By default, there is no primary input, all input plugins are equal.

--receive-timeout *value*

Specify a receive timeout in milliseconds (independently of any equivalent feature the input plugins).

When the current input plugin has received no packet within this timeout, automatically switch to the next plugin.

By default, without `--primary-input`, there is no automatic switch when the current input plugin is waiting for packets. With `--primary-input`, the default is 2,000 ms.

Remote control options

-a *address*
--allow *address*

Specify an IP address or host name which is allowed to send remote commands. Several `--allow` options can be used to specify several allowed remote control systems.

By default, all received commands are accepted. If at least one `--allow` option is specified, any remote command which is not sent by an allowed host is rejected.

This is a security feature, but not a perfect one since IP address spoofing is trivial with UDP.

--no-reuse-port

Disable the reuse port socket option. Do not use unless completely necessary.

-r [*address*:]*port*
--remote [*address*:]*port*

Specify the local UDP port which is used to receive remote commands. If an optional address is specified, it must be a local IP address of the system. By default, there is no remote control.

--udp-buffer-size *value*

Specifies the UDP socket receive buffer size (socket option).

Logging options

--log-message-count *value*

Specify the maximum number of buffered log messages. Log messages are displayed asynchronously in a low priority thread. This value specifies the maximum number of buffered log messages in memory, before being displayed. When too many messages are logged in a short period of time, while plugins use all CPU power, extra messages are dropped. Increase this value if you think that too many messages are dropped. The default is 512 messages.

-m
--monitor

Continuously monitor the system resources which are used by *tsswitch*. This includes CPU load, virtual memory usage. Useful to verify the stability of the application.

-s
--synchronous-log

Each logged message is guaranteed to be displayed, synchronously, without any loss of message. The downside is that a plugin thread may be blocked for a short while when too many messages are logged. This option shall be used when all log messages are needed and the source and destination are not live streams (files for instance). This option is not recommended for live streams, when the responsiveness of the application is more important than the logged messages.

--timed-log

Each logged message contains a time stamp.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N*]

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose

Produce verbose messages.

--version

Display the version number.

tstabcomp

Compile or decompile MPEG tables from XML files

This utility is an MPEG table compiler which takes MPEG tables in *source form* as XML files and produces binary section files.

The *tstabcomp* utility is also an MPEG table decompiler. From a binary file containing sections, it recreates an XML file. This XML file can be edited by hand and recompiled for instance.

See section 2.2 for a description of the format of PSI/SI files which can be manipulated by TSDuck and more specifically section 2.2.2 for a complete description of XML files.

Usage

```
tstabcomp [options] input-file ...
```

Input files

XML source files to compile or binary table files to decompile. By default, files ending in *.xml* are compiled and files ending in *.bin* are decompiled. For other files, explicitly specify `--compile` or `--decompile`.

Options

`--atsc`

Assume that the tables are used in an ATSC context.

This option is only useful when decompiling ATSC-related stuff which are found in the binary section file before the first ATSC-specific table. For instance, when a PMT with ATSC-specific descriptors is found before the first ATSC MGT or VCT. This is how *tstabcomp* can interpret these descriptors as ATSC descriptors and not as private DVB descriptors.

`-c`
`--compile`

Compile all files as XML source files into binary files. This is the default for *.xml* files.

`-d`
`--decompile`

Decompile all files as binary files into XML files. This is the default for *.bin* files.

`--default-charset` *name*

Default DVB character set to use. The available table names are: ISO-6937, ISO-8859-1, ISO-8859-10, ISO-8859-11, ISO-8859-13, ISO-8859-14, ISO-8859-15, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, UNICODE, UTF-8.

With `--compile`, this character set is used to encode strings. If a given string cannot be encoded with this character set or if this option is not specified, an appropriate character set is automatically selected.

With `--decompile`, this character set is used to interpret DVB strings without explicit character table code. According to DVB standard ETSI EN 300 468, the default DVB character set is ISO-6937. However, some bogus signalization may assume that the default character set is different, typically the usual local character table for the region. This option forces a non-standard character table.

`--europe`

A synonym for `'--default-charset ISO-8859-15'`.

`-o` *file-name*
`--output` *file-name*

Specify the output file name. By default, the output file has the same name as the input and extension *.bin* (compile) or *.xml* (decompile).

If the specified path is a directory, the output file is built from this directory and default file name. If more than one input file is specified, the output path, if present, must be a directory name.

--pack-and-flush

When loading a binary file for decompilation, pack incomplete tables, ignoring missing sections, and flush them.

Use with care because this may create inconsistent tables.

--strict-xml

Save XML documents in strictly conformant XML format. By default, do not escape characters when this is not syntactically necessary to make the XML text more human-readable.

-x
--xml-model

Display the XML model of the table files. This model is not a full XML-Schema, this is an informal template file which describes the expected syntax of TSDuck XML files. If `--output` is specified, the model is saved here. Do not specify input files.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N*]

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose

Produce verbose messages.

--version

Display the version number.

tstabdump

Dump MPEG Tables

This utility dumps in human readable format MPEG tables, as saved in binary files by the *tstables* utility.

Usage

```
tstabdump [options] [input-file ...]
```

Input files

Binary section file. Several files can be specified. By default, without file and without `--ip-udp`, the binary tables are read from the standard input.

With `--ip-udp`, no file shall be specified. Binary sections and tables are received over UDP/IP as sent by the utility *tstables* or the plugin *tables*.

Options

`-x` *value*
`--max-tables` *value*

Maximum number of tables or sections to dump. Stop logging tables when this limit is reached. This option is useful with `--ip-udp` which never ends otherwise.

`--no-pager`

Do not send output through a pager process. By default, if the output device is a terminal, the output is paged. See 3.1.4 for more details.

Tables and sections formatting options

`--atsc`

Assume that the transport stream is an ATSC one.

ATSC streams are normally automatically detected from their signalization. This option is only useful when ATSC-related stuff are found in the TS before the first ATSC-specific table. For instance, when a PMT with ATSC-specific descriptors is found before the first ATSC MGT or VCT.

`-c`
`--c-style`

Same as `--raw-dump` (no interpretation of section) but dump the bytes in C-language style, eg. "`0x01, 0x02,`" instead of "`01 02`". Useful to include this output as data in a C source file.

`--default-charset` *name*

Default character set to use when interpreting DVB strings without explicit character table code. According to DVB standard ETSI EN 300 468, the default DVB character set is ISO-6937. However, some bogus signalization may assume that the default character set is different, typically the usual local character table for the region. This option forces a non-standard character table. The available table names are: ISO-6937, ISO-8859-1, ISO-8859-10, ISO-8859-11, ISO-8859-13, ISO-8859-14, ISO-8859-15, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, UNICODE, UTF-8.

`--default-pds` *value*

Default private data specifier (PDS). This option is meaningful only when the signalization is incorrect, when private descriptors appear in tables without a preceding *private_data_specifier_descriptor*. The specified value is used as private data specifier to interpret private descriptors. The PDS value can be an integer or one of (not case-sensitive): "Nagra", "TPS", "EACEM"," EICTA", "Logiways", "CanalPlus", "Eutelsat".

`--europe`

A synonym for '`--default-charset ISO-8859-15`'. This is a handy shortcut for commonly incorrect signalization on some European satellites. In that signalization, the character encoding is ISO-8859-15, the most common encoding for Latin & Western Europe languages. However, this is not the default DVB character set and it should be properly specified in all strings, which is not

the case with some operators. Using this option, all DVB strings without explicit table code are assumed to use ISO-8859-15 instead of the standard ISO-6937 encoding.

--nested-tlv[=*min-size*]

With option `--tlv`, try to interpret the value field of each TLV record as another TLV area. If the *min-size* value is specified, the nested TLV interpretation is performed only on value fields larger than this size. The syntax of the nested TLV is the same as the enclosing TLV.

-r
--raw-dump

Raw dump of section, no interpretation.

--tlv *syntax*

For sections of unknown types, this option specifies how to interpret some parts of the section payload as TLV records. Several `--tlv` options are allowed, each one describes a part of the section payload.

Each syntax string has the form "*start,size,tagSize,lengthSize,order*". The *start* and *size* fields define the offset and size of the TLV area in the section payload. If the *size* field is "`auto`", the TLV extends up to the end of the section. If the start field is "`auto`", the longest TLV area in the section payload will be used. The fields *tagSize* and *lengthSize* indicate the size in bytes of the Tag and Length fields in the TLV structure. The field *order* must be either "`msb`" or "`lsb`" and indicates the byte order of the Tag and Length fields.

All fields are optional. The default values are "`auto,auto,1,1,msb`".

UDP reception options

These options apply only when `--ip-udp` is used. In this case, the binary sections are received using UDP/IP. No input file is used.

--buffer-size *value*

Specify the UDP socket receive buffer size (socket option).

--default-interface

Let the system find the appropriate local interface on which to listen. By default, listen on all local interfaces.

--first-source

Filter UDP packets based on the source address. Use the sender address of the first received packet as only allowed source.

This option is useful when several sources send packets to the same destination address and port. Accepting all packets could result in a corrupted stream and only one sender shall be accepted.

To allow a more precise selection of the sender, use option `--source`. Options `--first-source` and `--source` are mutually exclusive.

-ip-udp [[*source@*]*address*:]*port*

Specify that the sections and tables are received from UDP/IP, as sent by *tstables* or the plugin *tables*.

The *port* part is mandatory and specifies the UDP port to listen on. The *address* part is optional. It specifies an IP multicast address to listen on. It can be also a host name that translates to a multicast address. If the address is not specified, the plugin simply listens on the specified local port and receives the packets which are sent to one of the local (unicast) IP addresses of the system.

An optional source address can be specified as *source@address:port* in the case of source-specific multicast (SSM).

--local-address *address*

Specify the IP address of the local interface on which to listen. It can be also a host name that translates to a local address. By default, listen on all local interfaces.

--no-encapsulation

With `--ip-udp`, receive the tables as raw binary messages in UDP packets. By default, the tables are formatted into TLV messages.

--no-reuse-port

Disable the reuse port socket option. Do not use unless completely necessary.

--receive-timeout *value*

Specify the UDP reception timeout in milliseconds. This timeout applies to each receive operation, individually. By default, receive operations wait for data, possibly forever.

--reuse-port

Set the reuse port socket option. This is now enabled by default, the option is present for legacy only.

--source *address[:port]*

Filter UDP packets based on the specified source address.

This option is useful when several sources send packets to the same destination address and port. Accepting all packets could result in a corrupted stream and only one sender shall be accepted.

Options `--first-source` and `--source` are mutually exclusive.

--ssm

This option forces the usage of source-specific multicast (SSM) using the source address which is specified by the option `--source`. Without `--ssm`, standard ("any-source') multicast is used and the option `--source` is used to filter incoming packets.

The `--ssm` option is implicit when the classical SSM syntax *source@address:port* is used.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N***]**

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose

Produce verbose messages.

--version

Display the version number.

tstables

Collect MPEG Tables

This utility collects MPEG tables from a transport stream. The tables can be saved in a human readable format, in binary or XML files or sent over UDP/IP to some collecting server. It is possible to save the tables in several formats at the same time. By default, the tables are displayed in human-readable format on the standard output.

Usage

```
tstables [options] [input-file]
```

Input file

MPEG transport stream, either a capture file or a pipe from a live stream. The input must be a binary stream of 188-byte packets. If the input file is omitted, the standard input is used.

Tables and sections selection options

--all-once

Same as `--all-sections` but collect each section only once per combination of PID, table id, table id extension, section number and version.

-a
--all-sections

Display/save all sections, as they appear in the stream. By default, collect complete tables, with all sections of the tables grouped and ordered and collect each version of a table only once.

Note that this mode is incompatible with `--xml-output` since valid XML structures may contain complete tables only.

-d
--diversified-payload

Select only sections with *diversified* payload. This means that section payloads containing the same byte value (all `0x00` or all `0xFF` for instance) are ignored. Typically, such sections are stuffing and can be ignored that way.

--exclude-current

Exclude short sections and long sections with "current" indicator. This is rarely necessary.

See also `--include-next`.

--include-next

Include long sections with "next" indicator. By default, they are excluded.

-x *value*
--max-tables *value*

Maximum number of tables to dump. Stop execution when this limit is reached.

--negate-pid

Negate the PID filter: specified PID's are excluded.

Warning: this can be a dangerous option on complete transport streams since PID's not containing sections can be accidentally selected.

-n
--negate-tid

Negate the TID filter: specified TID's are excluded.

--negate-tid-ext

Negate the TID extension filter: specified TID extensions are excluded.

--no-duplicate

Do not report consecutive identical tables with a short section in the same PID. This can be useful for ECM's. This is the way to display new ECM's only. By default, tables with long sections are reported only when a new version is detected but tables with a short section are all reported.

--no-pager

Do not send output through a pager process. By default, if the output device is a terminal, the output is paged. See 3.1.4 for more details.

-p *pid1*[*-pid2*]
--pid *pid1*[*-pid2*]

PID filter: select packets with these PID values. Several `-p` or `--pid` options may be specified.

By default, without `-p` or `--pid` option, all PID's are used.

PID's containing PES data are automatically ignored.

--psi-si

Add all PID's containing PSI/SI tables, ie. PAT, CAT, PMT, NIT, SDT and BAT. The PMT PID's are dynamically collected each time a new PAT is encountered.

Note that EIT, TDT and TOT are not included. Use `--pid 18` to get EIT and `--pid 20` to get TDT and TOT.

-t *id1*[*-id2*]
--tid *id1*[*-id2*]

TID filter: select sections with these TID (table id) values. Several `-t` or `--tid` options may be specified. Without `-t` or `--tid` option, all tables are saved.

-e *id1*[*-id2*]
--tid-ext *id1*[*-id2*]

TID extension filter: select sections with these table id extension values (apply to long sections only). Several `-e` or `--tid-ext` options may be specified. Without `-e` or `--tid-ext` option, all tables are saved.

Output options

-b *filename*
--binary-output *filename*

Save the sections in raw binary format in the specified output file name. See also option `-m`, `--multiple-files`.

-f
--flush

Flush standard output after each display. Useful to monitor the content if the output has been redirected to a disk file.

-i *address:port*
--ip-udp *address:port*

Send binary tables over UDP/IP to the specified destination. The *address* specifies an IP address which can be either unicast or multicast. It can be also a host name that translates to an IP address. The *port* specifies the destination UDP port.

--local-udp address

With `--ip-udp`, when the destination is a multicast address, specify the IP address of the outgoing local interface. It can be also a host name that translates to a local address.

--log

Display a short one-line log of each table instead of full table display.

--log-size *value*

With option `--log`, specify how many bytes are displayed at the beginning of the table payload (the header is not displayed). The default is 8 bytes.

-m
--multiple-files

Create multiple binary output files, one per section. A binary output file name must be specified (option `-b` or `--binary-output`). Assuming that the specified file name has the form '`base.ext`', each file is created with the name '`base_p`*XXXX*`_t`*XX*`.ext`' for short sections and '`base_p`*XXXX*`_t`*XX*`_e`*XXXX*`_v`*XX*`_s`*XX*`.ext`' for long sections, where the *XX* respectively specify the hexadecimal values of the PID, TID (table id), TIDext (table id extension), version and section index.

--no-encapsulation

With `--ip-udp`, send the tables as raw binary messages in UDP packets. By default, the tables are formatted into TLV messages.

-o *filename*
--output-file *filename*
--text-output *filename*

Save the tables or sections in human-readable text format in the specified file name. By default, when no output option is specified, text is produced on the standard output.

If you need text formatting on the standard output in addition to other output like binary files (`--binary-output`) or UPD/IP (`--ip-udp`), explicitly specify this option with "`-`" as output file name.

--packet-index

Display the index of the first and last TS packet of each displayed section or table.

--rewrite-binary

With `--binary-output`, rewrite the same file with each table. The specified file always contains one single table, the latest one.

--rewrite-xml

With `--xml-output`, rewrite the same file with each table. The specified file always contains one single table, the latest one.

--strict-xml

Save XML documents in strictly conformant XML format. By default, do not escape characters when this is not syntactically necessary to make the XML text more human-readable.

--time-stamp

Display a time stamp (current local time) with each table.

--ttl *value*

With `--ip-udp`, specifies the TTL (Time-To-Live) socket option. The actual option is either "Unicast TTL" or "Multicast TTL", depending on the destination address. Remember that the default Multicast TTL is 1 on most systems.

--xml-output *filename*

Save the tables in XML format in the specified file. To output the XML text on the standard output, explicitly specify this option with "`-`" as output file name.

Tables and sections manipulation options

--fill-eit

Before exiting, add missing empty sections in EIT's and flush them. This can be useful with segmented EIT schedule where empty sections at end of segments are usually not transmitted.

--pack-all-sections

Same as `--all-sections` but also modify each long section so that it becomes a valid complete table. Its *section_number* and *last_section_number* are forced to zero. Use with care because this may create inconsistent tables. This option can be useful with tables with sparse sections such as EIT's to save them in XML format (as an alternative, see also `--fill-eit`).

--pack-and-flush

Before exiting, pack incomplete tables, ignoring missing sections, and flush them. Use with care because this may create inconsistent tables. Unlike option `--pack-all-sections`, `--pack-and-flush` does not force `--all-sections` because it only applies to the last incomplete tables before exiting.

Tables and sections formatting options

--atsc

Assume that the transport stream is an ATSC one.

ATSC streams are normally automatically detected from their signalization. This option is only useful when ATSC-related stuff are found in the TS before the first ATSC-specific table. For instance, when a PMT with ATSC-specific descriptors is found before the first ATSC MGT or VCT.

-c
--c-style

Same as `--raw-dump` (no interpretation of section) but dump the bytes in C-language style, eg. "`0x01, 0x02,`" instead of "`01 02`". Useful to include this output as data in a C source file.

--default-charset *name*

Default character set to use when interpreting DVB strings without explicit character table code. According to DVB standard ETSI EN 300 468, the default DVB character set is ISO-6937. However, some bogus signalization may assume that the default character set is different, typically the usual local character table for the region. This option forces a non-standard character table. The available table names are: ISO-6937, ISO-8859-1, ISO-8859-10, ISO-8859-11, ISO-8859-13, ISO-8859-14, ISO-8859-15, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9, UNICODE, UTF-8.

--default-pds *value*

Default private data specifier (PDS). This option is meaningful only when the signalization is incorrect, when private descriptors appear in tables without a preceding *private_data_specifier_descriptor*. The specified value is used as private data specifier to interpret private descriptors. The PDS value can be an integer or one of (not case-sensitive): "Nagra", "TPS", "EACEM"," EICTA", "Logiways", "CanalPlus", "Eutelsat".

--europe

A synonym for '`--default-charset ISO-8859-15`'. This is a handy shortcut for commonly incorrect signalization on some European satellites. In that signalization, the character encoding is ISO-8859-15, the most common encoding for Latin & Western Europe languages. However, this is not the default DVB character set and it should be properly specified in all strings, which is not the case with some operators. Using this option, all DVB strings without explicit table code are assumed to use ISO-8859-15 instead of the standard ISO-6937 encoding.

--nested-tlv[=*min-size*]

With option `--tlv`, try to interpret the value field of each TLV record as another TLV area. If the *min-size* value is specified, the nested TLV interpretation is performed only on value fields larger than this size. The syntax of the nested TLV is the same as the enclosing TLV.

-r
--raw-dump

Raw dump of section, no interpretation.

--tlv *syntax*

For sections of unknown types, this option specifies how to interpret some parts of the section payload as TLV records. Several `--tlv` options are allowed, each one describes a part of the section payload.

Each syntax string has the form "*start,size,tagSize,lengthSize,order*". The *start* and *size* fields define the offset and size of the TLV area in the section payload. If the *size* field is "`auto`", the TLV extends up to the end of the section. If the start field is "`auto`", the longest TLV area in the section payload will be used. The fields *tagSize* and *lengthSize* indicate the size in bytes of the Tag and

Length fields in the TLV structure. The field *order* must be either "`msb`" or "`lsb`" and indicates the byte order of the Tag and Length fields.

All fields are optional. The default values are "`auto,auto,1,1,msb`".

Generic common command options

The following options are implicitly defined in all commands.

`--debug[=`*`N`*`]`

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

`--help`

Display command help text.

`-v`
`--verbose`

Produce verbose messages.

`--version`

Display the version number.

tsterinfo

DVB-Terrestrial Information

This utility performs various operations and conversions on DVB-T transmission and modulation parameters:

- Compute the carrier frequency from a UHF or VHF channel number and optional offset count.

  Triggered when option `--uhf-channel`, `--vhf-channel` and optionally `--offset-count`, are specified.

- Retrieve the UHF or VHF channel number and offset count from a carrier frequency.

  Triggered when option `--frequency` is specified.

- Compute the nominal transport stream bitrate from OFDM modulation parameters (bandwidth, high-priority stream error correction rate, constellation and guard interval). Supported for non-hierarchical transmission only.

  Triggered when options `--guard-interval` and `--high-priority-fec`, and optionally `--bandwidth` and `--constellation`, are specified.

- Given a transport stream bitrate, retrieve the OFDM modulation parameters (bandwidth, high-priority stream error correction rate, constellation and guard interval). Sometimes, several combinations of parameters are possible ; they are all reported (see also option `--max-guess`). This could be useful on Windows systems where the tuners are not able to report their current parameters. In that case, you can use `tsanalyze`, `tsbitrate` or `tsp -v` to evaluate the transport stream bitrate based on PCR analysis. Then, `tsterinfo` will retrieve the most probable modulation parameters. Note that only the four mentioned parameters can be retrieved. All other DVB-T transmission parameters are independent from the transport stream bitrate.

  Triggered when option `--bitrate` is specified.

See some examples in section 5.1.5.

Usage

```
tsterinfo [options]
```

Options

-w *value*
--bandwidth *value*

Specify the OFMD bandwith, used to compute the resulting bitrate. Must be one of "8-MHz", "7-MHz", "6-MHz", "5-MHz" (default: "8-MHz").

-b *value*
--bitrate *value*

Transport stream bitrate in bits/second, based on 188-byte packets. Given this bitrate, `tsterinfo` will try to guess the OFDM modulation parameters: bandwidth, high-priority stream error correction rate, constellation and guard interval.

-c *value*
--constellation *value*

Specify the OFMD constellation, used to compute the resulting bitrate. Must be one of "QPSK", "16-QAM", "64-QAM" (default: "64-QAM").

-d
--default-region

Display the default region for UHF/VHF band frequency layout.

See also option `--hf-band-region`.

-f *value*
--frequency *value*

Carrier frequency in Hz. UHF or VHF channel and offset will be displayed.

-g *value*
--guard-interval *value*

Specify the OFMD guard interval, used to compute the resulting bitrate. Must be one of "1/32", "1/16", "1/8", "1/4" (no default).

-r *name*
--hf-band-region *name*

Specify the region for UHF/VHF band frequency layout.

The default region is “europe". Another default region may be specified per user in the TSDuck configuration file (see Appendix A).

-h *value*
--high-priority-fec *value*

Specify the OFMD error correction for high priority streams, used to compute the resulting bitrate. Must be one of "1/2", "2/3", "3/4", "5/6", "7/8" (no default).

-m *value*
--max-guess *value*

When used with --bitrate, specify the maximum number of sets of modulation parameters to display. By default, display only one set of parameters, the one giving the closest bitrate. When the given bitrate is not exact and the transmission parameters are uncertain, it may be useful to display more than one possible set of values. The difference between the specified bitrate and nominal bitrate is displayed for each set of parameters. The various sets of parameters are displayed in increasing order of bitrate difference (ie. most probable parameters first).

When more than one set of parameters give the same bitrate, they are all displayed, regardless of --max-guess.

-o *value*
--offset-count *value*

Specify the number of offsets from the UHF or VHF channel. The default is zero. See options --uhf-channel and --vhf-channel.

-s
--simple

Produce simple output: only numbers, no comment, no formatting. Typically useful to write scripts and reuse tsterinfo output.

-u *value*
--uhf-channel *value*

Specify the UHF channel number of the carrier. Can be combined with an --offset-count option. The resulting frequency will be displayed.

-v *value*
--vhf-channel *value*

Specify the VHF channel number of the carrier. Can be combined with an --offset-count option. The resulting frequency will be displayed.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N*]

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

--verbose

Produce verbose messages.

--version

Display the version number.

tsversion

Check version, download and upgrade TSDuck

By default, this utility simply displays the TSDuck version. It can also connect to GitHub to list all available releases of TSDuck, check for a new version, download it or upgrade TSDuck to the latest version.

The following command checks for a new version online and, if one is available, downloads it and upgrades TSDuck:

```
tsversion --upgrade
```

Detecting the availability of a new release always works. However, to perform an upgrade, the binary packages for the current operating system and architecture must be available online. Not all combinations of binary packages are available. It is only guaranteed that TSDuck can be upgraded by `tsversion` for Windows 32 and 64 bits, Fedora 64 bits, Ubuntu 64 bits and MacOS (through Homebrew). For other platforms, you have to recompile TSDuck from sources.

Listing versions and information about versions access the GitHub site. This is the only TSDuck command which performs Internet access.

Remote information is requested from the GitHub API. GitHub limits the anonymous access to its API to a certain number of requests per hour per source IP address. If you get an error such as "*API rate limit exceeded*", you may have to wait for the next hour and retry. Alternatively, if you are a registered GitHub user and you have a registered authentication token, this rate limit is removed. Set the value of your authentication token into the environment variable `TSDUCK_GITHUB_API_TOKEN` before using *tsversion*. For macOS users, if the environment variable `HOMEBREW_GITHUB_API_TOKEN` is already defined, it will be used.

Usage

```
tsversion [options]
```

Options

-a
--all

List all available versions of TSDuck from GitHub.

-b
--binary

With `--download`, fetch the binary installers of the latest version. This is the default. When `--source` is specified, you have to explicitly specify `--binary` if you also need the binary installers.

-c
--check

Check if a new version of TSDuck is available from GitHub.

-d
--download

Download the latest version (or the version specified by `--name`) from GitHub. By default, download the binary installers for the current operating system and architecture. Specify `--source` to download the source code.

If a local file with the same name and size already exists, the local file is reused and the download operation is skipped.

-f
--force

Force downloads even if a file with same name and size already exists.

-i
--integer

Display the current version of TSDuck in integer format, suitable for comparison in a script.

Example: 31000669 for 3.10-669 (5 digits are used for the last commit number).

-l
--latest

Display the latest version of TSDuck from GitHub.

-n *version-name*
--name *version-name*

Get information for or download from GitHub the specified version, not the latest one.

-o *dir-name*
--output-directory *dir-name*

Specify the output directory for downloaded files (current directory by default).

--proxy-host *name*

Optional proxy host name for Internet access.

--proxy-password *string*

Optional proxy password for Internet access (for use with `--proxy-user`).

--proxy-port *value*

Optional proxy port for Internet access (for use with `--proxy-host`).

--proxy-user *name*

Optional proxy user name for Internet access.

-s
--source

With `--download`, download the source code archive instead of the binary installers.

-t
--this

Display the current version of TSDuck (this executable).

-u
--upgrade

Upgrade TSDuck to the latest version.

Generic common command options

The following options are implicitly defined in all commands.

--debug[=*N***]**

Produce verbose debug output. Specify an optional debug level *N* (1 by default).

--help

Display command help text.

-v
--verbose

Produce verbose messages.

--version

Display the version number.

4 TSP Plugins

This chapter contains the reference documentation of all plugins for *tsp*, the *transport stream processor*. The input and output plugins can also be used by the command *tsswitch*.

The Table 2 lists all available plugins.

Table 2: tsp plugins

Plugin	Type	Description
aes	packet	Experimental AES Scrambling
analyze	packet	Analyze the structure of the transport stream
bat	packet	Perform various transformations on the BAT
bitrate_monitor	packet	Monitor the bitrate of the TS or a given PID
boostpid	packet	Boost the bitrate of a PID, stealing stuffing packets
cat	packet	Perform various transformations on the CAT
clear	packet	Extract clear (non scrambled) sequences
continuity	packet	Check TS continuity counters
count	packet	Count TS packets per PID
craft	input, packet	Build or modify specifically crafted packets
cutoff	packet	Set labels on TS packets upon reception of UDP messages
datainject	packet	DVB SimulCrypt-compliant EMM and private data injector
decap	packet	Decapsulate TS packets from a PID produced by *encap* plugin
dektec	input, output	Dektec DTA-1xx DVB-ASI and modulator devices I/O
descrambler	packet	Generic DVB descrambler
drop	output	Drop output packets
duplicate	packet	Duplicate PID's, reusing null packets
dvb	input	DVB receiver devices (DVB-S, DVB-C, DVB-T) input
eit	packet	Analyze EIT sections
encap	packet	Encapsulate packets from several PID's into one single PID
file	input, output, packet	Transport stream files input / output. As packet processor plugin, save packets to a file and pass to next plugin
filter	packet	Filter TS packets according to various criteria
fork	input, output, packet	Exchange packets with a created process, either input or output
hides	output	Send the transport stream to a HiDes modulator device
history	packet	Report a history of major events on the transport stream
hls	input, output	Receive or generate HTTP Live Streaming (HLS) media
http	input	Read a transport stream from an HTTP server
inject	packet	Inject a table into a transport stream
ip	input, output	UDP/IP sockets I/O, including multicast IP
limit	packet	Limit the global bitrate by dropping packets
merge	packet	Merge TS packets coming from the output of a created process
mpe	packet	Extract MPE (Multi-Protocol Encapsulation) datagrams
mpeinject	packet	Encapsulate and inject an incoming UDP stream into MPE

Plugin	Type	Description
mux	packet	Inject TS packets from a file into the transport
nit	packet	Perform various transformations on the NIT
nitscan	packet	Scan the NIT for tuning information
null	input	Null packets generator
pat	packet	Perform various transformations on the PAT
pattern	packet	Replace packet payload with a binary pattern
pcradjust	packet	Adjust PCR's according to a constant bitrate
pcrbitrate	packet	Permanently recompute bitrate based on PCR's
pcrextract	packet	Extract PCR's from TS packets
pcrverify	packet	Verify PCR values
pes	packet	Analyze PES packets
play	output	Play output TS on a media player
pmt	packet	Perform various transformations on the PMT
psi	packet	Extract all PSI tables (PAT, CAT, PMT, NIT, BAT, SDT)
psimerge	packet	Merge PSI/SI from mixed streams
reduce	packet	Reduce the bitrate by removing stuffing packets
regulate	packet	Regulate TS packets flow according to a bitrate or PCR
remap	packet	Generic PID remapper
rmorphan	packet	Remove unreferenced (“*orphan*”) PID’s
rmsplice	packet	Remove ads insertions using SCTE 35 splicing information
scrambler	packet	DVB scrambler
sdt	packet	Perform various transformations on the SDT
sections	packet	Remove or merge sections from various PID's
sifilter	packet	Extract PSI/SI PID’s
skip	packet	Skip leading packets in a TS
slice	packet	Pass or drop packets based on packet numbers or relative TS time
spliceinject	packet	Inject SCTE 35 splice commands in a transport stream
stuffanalyze	packet	Analyze the level of stuffing in sections
svremove	packet	Remove a service
svrename	packet	Rename a service (modify service id, name, type, etc.)
t2mi	packet	Extract T2-MI (DVB-T2 Modulator Interface) packets
tables	packet	Collect MPEG tables
teletext	packet	Extract Teletext subtitles in SRT format
time	packet	Schedule packets pass or drop
timeref	packet	Update TDT and TOT with a new time reference
trigger	packet	Trigger actions on selected labeled TS packets
tsrename	packet	Rename a transport stream (modify ts id, etc.)
until	packet	Pass TS packets until specified conditions
zap	packet	Zap on one service, create an SPTS

Some plugins are related to the scrambling of TS packets and Conditional Access Systems. Please note the following:

- The DVB-CSA scrambling algorithm is inherently and purposely very slow with a software implementation. A 3.4 MHz Pentium 4 CPU, for instance, cannot (de)scramble more than 20 Mb/s. Be cautious not to ask for impossible tasks, like real time (de)scrambling of a complete TS on a regular PC.
- These *tsp* plugins are implemented for testing Conditional Access Systems, either on the head-end or set-top box side. TSDuck does not provide any support to hack or circumvent Conditional Access Systems and will never do so. The CAS-related plugins require and use external CAS-provided systems (ECMG, EMMG and smartcards). All secrecy and proprietary CAS information remain isolated inside these external systems and TSDuck does not attempt to access this type of secret and private information. TSDuck only interacts with these systems using their external communication protocols.

aes

Experimental AES Scrambling

This plugin scrambles or descrambles the payload of packets from a specified service using AES and a fixed key. Various chaining modes are allowed. All video, audio and subtitles components of the service are scrambled.

By default, the plugin scrambles the packets. Use option `--descramble` to descramble the packets.

Usage

```
tsp -P aes [options] [service]
```

Parameter

Specifies the service to scramble or descramble. If the argument is an integer value (either decimal or hexadecimal), it is interpreted as a service id. Otherwise, it is interpreted as a service name, as specified in the SDT. The name is not case sensitive and blanks are ignored.

If the service is unspecified, individual PID's are scrambled (see option `--pid`).

Options

`--cbc`

Use Cipher Block Chaining (CBC) mode without padding. The residue (last part of the packet payload, shorter than 16 bytes) is left clear.

`--cts1`

Use Cipher Text Stealing (CTS) mode. TS packets with a payload shorter than 17 bytes are left clear.

Several incompatible designs of CTS exist. This one implements the description in:

1) Bruce Schneier, Applied Cryptography (2nd, Ed.), pp 191, 195

2) RFC 2040, The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms

3) "CBC ciphertext stealing" in http://en.wikipedia.org/wiki/Ciphertext_stealing

`--cts2`

Use Cipher Text Stealing (CTS) mode. TS packets with a payload shorter than 16 bytes are left clear.

Several incompatible designs of CTS exist. This one implements the description in http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/ciphertext%20stealing%20proposal.pdf

`--cts3`

Use ECB Cipher Text Stealing (CTS) mode. TS packets with a payload shorter than 17 bytes are left clear.

Several incompatible designs of CTS exist. This one implements the description of “ECB ciphertext stealing" in http://en.wikipedia.org/wiki/Ciphertext_stealing

`--cts4`

Use ECB Cipher Text Stealing (CTS) mode. TS packets with a payload shorter than 17 bytes are left clear.

Several incompatible designs of CTS exist. This one implements the ECB ciphertext stealing which is used in ST 71xx chips.

`-d`
`--descramble`

Descramble instead of scramble.

`--dvs042`

Use DVS 042 (now ANSI/SCTE 52 2003) cipher block chaining mode.

TS packets with a payload shorter than 16 bytes are left clear. Note that the DVS 042 standard allows the scrambling of short messages (shorter than the cipher block size, ie. 16 bytes with AES) but the two versions of the standard (ANSI/SCTE 52 2003 and ANSI/SCTE 52 2008) have incompatible descriptions of the processing of short messages. To avoid conflicts, this plugin does not scramble these short messages.

--ecb

Use Electronic Code Book (ECB) mode without padding. The residue (last part of the packet payload, shorter than 16 bytes) is left clear. This is the default mode.

-i *value*
--iv *value*

Specifies the initialization vector. Must be a string of 32 hexadecimal digits. Must not be used in ECB mode and the various ECB-CTS modes. The default IV is all zeroes.

-k *value*
--key *value*

Specifies a fixed and constant AES key for all TS packets. The value must be a string of 32 or 64 hexadecimal digits. This is a mandatory parameter.

-p *pid1*[*-pid2*]
--pid *pid1*[*-pid2*]

Specifies PID's to scramble. Can be used instead of specifying a service.

Several `-p` or `--pid` options may be specified.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[*-label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

analyze

Global Transport Stream Analysis

This plugin performs various types of global analysis on the transport stream. It is equivalent to the *tsanalyze* utility. Actually, the following two commands produce the same result:

```
tsanalyze options filename
tsp -I file filename -P analyze options -O drop
```

Usage

```
tsp -P analyze [options]
```

General purpose options

-i *seconds*
--interval *seconds*

Produce a new output file at regular intervals. After outputing a file, the analysis context is reset, ie. each output file contains a fully independent analysis.

-m
--multiple-files

When used with `--interval` and `--output-file`, create a new file for each analysis instead of rewriting the previous file. Assuming that the specified output file name has the form `base.ext`, each file is created with a time stamp in its name as `base_YYYYMMDD_hhmmss.ext`.

-o *filename*
--output-file *filename*

Specify the output text file for the analysis result. By default, use the standard output.

Warning: if you do not specify this option, be sure to redirect the output plugin to something different from the default. Otherwise, the text output of the analysis will be mixed with the binary output of the TS packets !

Analysis and output control options

The options for controlling the analysis and the output are the same as for the *tsanalyze* utility.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

bat

Perform Various Transformations on a BAT

This plugin performs various transformations on the BAT, either all BAT's of the transport stream or one specific BAT for one specific bouquet.

Usage

```
tsp -P bat [options]
```

Options

--bitrate *value*

Specifies the bitrate in bits / second of the PID containing the BAT if a new one is created.

The default is 3,000 b/s.

-b *value*
--bouquet-id *value*

Specify the bouquet id of the BAT to modify and leave other BAT's unmodified. By default, all BAT's are modified.

--cleanup-private-descriptors

Remove all private descriptors without preceding *private_data_specifier_descriptor*.

-c
--create

Create a new empty BAT if none was received after one second.

This is equivalent to `--create-after 1000`.

--create-after *milliseconds*

Create a new empty BAT if none was received after the specified number of milliseconds. If an actual BAT is received later, it will be used as the base for transformations instead of the empty one.

-i
--increment-version

Increment the version number of the BAT.

--inter-packet *value*

When a new BAT is created and `--bitrate` is not present, this option specifies the packet interval for the BAT PID, that is to say the number of TS packets in the transport between two packets of the PID.

Use instead of `--bitrate` if the global bitrate of the TS cannot be determined.

-v *value*
--new-version *value*

Specify a new value for the version of the BAT.

--pds *value*

With option `--remove-descriptor`, specify the private data specifier which applies to the descriptor tag values above `0x80`.

--remove-descriptor *value*

Remove from the BAT all descriptors with the specified tag. Several `--remove-descriptor` options may be specified to remove several types of descriptors. See also option `--pds`.

-r *value*
--remove-service *value*

Remove the specified service_id from the following descriptors: *service_list_descriptor*, *logical_channel_number_ descriptor*. Several `--remove-service` options may be specified to remove several services.

--remove-ts *value*

Remove from the BAT all references to the transport stream with the specified *ts_id* value. Several `--remove-ts` options may be specified to remove several TS.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

bitrate_monitor

Monitor the bitrate of the TS or a given PID

This plugin is used to monitor the bitrate of the complete transport stream or a given PID. Note that the bitrate is the instantaneous bitrate, meaning that it is computed from the packets received during the last *n* seconds (*n* is a plugin parameter, default value = 5).

If the bitrate value is outside of the specified range, an alarm is reported.

An alarm command can be specified to report anomalies in a custom way. If such a command is present, it will be called with the problem description as parameter.

Usage

```
tsp -P bitrate_monitor [options]
```

Options

-a *"command"*
--alarm-command *"command"*

Command to run when the bitrate goes either out of range or back to normal.

The command receives an additional string parameter containing an informational message.

--min *value*

Set minimum allowed value for bitrate in bits/s. Default value = 10 bits/s.

--max *value*

Set maximum allowed value for bitrate bits/s. Default value = 2^{32} bits/s.

Note that default values for min and max bitrate are only useful to detect if packets for the given PID are broadcast or not.

-p *value*
--periodic-bitrate *value*

Always report bitrate at the specific interval in seconds, even if the bitrate is in range.

--pid *value*

Specifies the PID to monitor.

By default, when no `--pid` is specified, monitor the bitrate of the full TS.

Compatibility : Previously, the PID to monitor could be specified as a command line parameter, without explicit `--pid` option. This is still accepted for compatibility for old scripts.

--set-label-above *label1*[*-label2*]

Set the specified labels on all packets while the bitrate is above normal.

Several `--set-label-above` options may be specified.

--set-label-below *label1*[*-label2*]

Set the specified labels on all packets while the bitrate is below normal.

Several `--set-label-below` options may be specified.

--set-label-go-above *label1*[*-label2*]

Set the specified labels on one packet when the bitrate goes above normal.

Several `--set-label-go-above` options may be specified.

--set-label-go-below *label1*[*-label2*]

Set the specified labels on one packet when the bitrate goes below normal.

Several `--set-label-go-below` options may be specified.

--set-label-go-normal *label1*[*-label2*]

Set the specified labels on one packet when the bitrate goes back to normal (within range).

Several `--set-label-go-normal` options may be specified.

`--set-label-normal` *`label1`*`[-`*`label2`*`]`

Set the specified labels on all packets while the bitrate is normal (within range).

Several `--set-label-normal` options may be specified.

`--tag` `'`*`string`*`'`

Message tag to be displayed in alarms. Useful when the plugin is used several times in the same process.

`-t` *`value`*
`--time-interval` *`value`*

Time interval in seconds used to compute the bitrate. The default is 5 seconds.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

`--help`

Display this help text.

`--only-label` *`label1`*`[-`*`label2`*`]`

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

boostpid

Boost the Bitrate of a PID

This plugin artificially increases the bitrate of a selected PID by adding empty packets (ie. without payload). The plugin does not really insert new packets in the TS, it "steals" stuffing packets.

Usage

```
tsp -P boostpid [options] pid addpkt inpkt
```

Parameters

pid

The first parameter specifies the PID to boost.

addpkt inpkt

The second and third parameters specify that *addpkt* TS packets must be automatically added after every *inpkt* input TS packets in the PID. Both *addpkt* and *inpkt* must be non-zero integer values.

As an example, the parameters 3 1 indicate to add 3 new empty packets in the PID for every existing packet. The resulting bitrate of the PID is multiplied by 4.

Take care to limit the added packet ratio to something realistic. The value 1000/1, for instance, is unrealistic since it is impossible in most cases to find 1000 stuffing packets to replace between all existing packets of the PID.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

cat

Perform Various Transformations on the CAT

This plugin performs various transformations on the CAT.

Usage

```
tsp -P cat [options]
```

Options

-a casid/pid[/private-data]
--add-ca-descriptor *casid/pid[/private-data]*

Add a *CA_descriptor* in the CAT with the specified CA System Id and EMM PID. The optional private data must be a suite of hexadecimal digits. Several **–add-ca-descriptor** options may be specified to add several descriptors.

-b *value*
--bitrate *value*

Specifies the bitrate in bits / second of the PID containing the CAT if a new one is created.

The default is 3,000 b/s.

--cleanup-private-descriptors

Remove all private descriptors without preceding *private_data_specifier_descriptor*.

-c
--create

Create a new empty CAT if none was received after one second.

This is equivalent to `--create-after 1000`.

--create-after *milliseconds*

Create a new empty CAT if none was received after the specified number of milliseconds. If an actual CAT is received later, it will be used as the base for transformations instead of the empty one.

This can be useful to force the creation of a CAT in a TS which has none (the CAT is an optional table).

-i
--increment-version

Increment the version number of the CAT.

--inter-packet *value*

When a new CAT is created and `--bitrate` is not present, this option specifies the packet interval for the CAT PID, that is to say the number of TS packets in the transport between two packets of the PID.

Use instead of `--bitrate` if the global bitrate of the TS cannot be determined.

-v *value*
--new-version *value*

Specify a new value for the version of the CAT.

-r *id1*[-*id2*]
--remove-casid *id1*[-*id2*]

Remove all *CA_descriptors* with any of the specified CA System Ids.

Several `--remove-casid` options may be specified.

--remove-pid *pid1*[-*pid2*]

Remove all *CA_descriptors* with the specified EMM PID values.

Several `--remove-pid` options may be specified.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

`--help`

Display this help text.

`--only-label` *`label1`*`[-`*`label2`*`]`

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

clear

Extract Clear (Non Scrambled) Sequences

This plugin extracts clear (non scrambled) sequences of a transport stream.

The extraction is based on one "*reference*" service (see option `-s`). When a clear packet is found on any audio or video stream of the reference service, all subsequent packets in the TS are transmitted. When no clear packet has been found in the last second, all subsequent packets in the TS are dropped.

This plugin is typically used after the plugin `zap`. It let the service pass when it is clear and drops it when it is scrambled.

Usage

```
tsp -P clear [options]
```

Options

-a
--audio

Check only audio PIDs for clear packets. By default, audio and video PIDs are checked.

-d *value*
--drop-after-packets *value*

Specifies the number of packets after the last clear packet to wait before stopping the packet transmission. By default, stop 1 second after the last clear packet (based on current bitrate).

-s *name-or-id*
--service *name-or-id*

Specify the reference service. If the argument is an integer value (either decimal or hexadecimal), it is interpreted as a service id. Otherwise, it is interpreted as a service name, as specified in the SDT. The name is not case sensitive and blanks are ignored. If this option is not specified, the first service in the PAT is used.

--stuffing

Replace excluded packets with stuffing (null packets) instead of removing them. Useful to preserve bitrate.

-v
--video

Check only video PIDs for clear packets. By default, audio and video PIDs are checked.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

continuity

Check Continuity Counters

This plugin checks the continuity counters on TS packets, PID per PID.

Usage

```
tsp -P continuity [options]
```

Options

-f
--fix

Fix incorrect continuity counters. By default, only display discontinuities.

-p *pid1*[*-pid2*]
--pid *pid1*[*-pid2*]

Check or fix continuity counters only in packets with these PID values. Several `-p` or `--pid` options may be specified. By default, all PID's are checked or fixed.

-t “*string*”
--tag “*string*”

Message tag to be displayed when packets are missing. Useful when the plugin is used several times in the same command line.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[*-label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

count

Count TS packets per PID

This plugin counts packets per PID and provides either a summary of packet counts or a detailed list of packet per PID.

Usage

```
tsp -P count [options]
```

Options

-a
--all

Report packet index and PID for all packets from the selected PID's. By default, only a final summary is reported.

-b
--brief

Brief display. Report only the numerical values, not comment on their usage. This option is useful for automatic processing of the resulting output.

-i *value*
--interval *value*

Report a time-stamp and global packet counts at regular intervals. The specified value is a number of packets.

-n
--negate

Negate the filter: specified PID's are excluded.

-o *filename*
--output-file *filename*

Specify the output file for reporting packet counters. By default, report on standard error using the tsp logging mechanism.

-p *pid1*[*-pid2*]
--pid *pid1*[*-pid2*]

PID filter: select packets with these PID values. Several `-p` or `--pid` options may be specified. By default, if `--pid` is not specified, all PID's are selected.

-s
--summary

Display a final summary of packet counts per PID. This is the default, unless `--all` or `--total` is specified, in which case the final summary is reported only if `--summary` is specified.

--tag “*string*”

Message tag to be displayed with count report lines. Useful when the plugin is used several times in the same command line.

-t
--total

Display the total packet counts in all PID's.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

craft (input)

Build specifically crafted input packets

This plugin generates fake transport stream packets from scratch. The various fields in the packets are specified using command line options.

Usage

```
tsp -I craft [options]
```

Options

--cc *value*

Specify the initial value of the *continuity_counter* field (0 by default).

--constant-cc

Do not increment the continuity counter.

By default, the continuity counter in incremented when the packet has a payload.

-c *value*
--count *value*

Specify the number of crafted packets to generate. After the last packet, an end-of-file condition is generated.

By default, if `--count` is not specified, crafted packets are generated endlessly.

--discontinuity

Set the *discontinuity_indicator* in the packets.

An adaptation field is created.

--error

Set the *transport_error_indicator* in the packets.

--es-priority

Set the *elementary_stream_priority_indicator* in the packets.

An adaptation field is created.

-j
--joint-termination

When `--count` is specified, perform a *joint termination* when completed instead of unconditional termination. See the description of the `tsp` command for more details on *joint termination*.

--no-payload

Do not use a payload.

--opcr *value*

Set this OPCR value in the packets.

An adaptation field is created.

--payload-pattern *value*

Specify the binary pattern to apply on packets payload.

The value must be a string of hexadecimal digits specifying any number of bytes. The pattern is repeated to fill the payload. The last repetition of the pattern is truncated if necessary.

The default is `FF`.

--payload-size *value*

Specify the size of the packet payload in bytes. When necessary, an adaptation field is created.

Note that `--payload-size 0` specifies that a payload exists with a zero size. This is different from `--no-payload` which also specifies that the payload does not exist.

By default, the payload uses all free space in the packet.

--pcr *value*

Set this PCR value in the packets.

An adaptation field is created.

-p *value*
--pid *value*

Specify the PID for the packets (0 by default).

--priority

Set the *transport_priority* flag in the packets.

--private-data *value*

Specify the complete binary content of the *transport_private_data* in the adaptation field. The value must be a string of hexadecimal digits specifying any number of bytes.

--pusi

Set the *payload_unit_start_indicator* in the packets.

--random-access

Set the *random_access_indicator* in the packets.

An adaptation field is created.

--scrambling *value*

Specify the value of the *transport_scrambling_control* field (0 by default).

--splice-countdown *value*

Create a splicing point and set this splice countdown value in the packets.

An adaptation field is created.

Generic common input plugin options

The following options are implicitly defined in all input plugins.

--help

Display plugin help text.

craft (packet processing)

Craft specific low-level transformations on packets

This plugin modifies precise fields in all TS packets.

Some operations may need space in the adaptation field. By default, the payload is left unmodified and a transformation is rejected if it needs to enlarge the adaptation field since this would destroy part of the existing payload. Enlarging the adaptation field is possible only when `--payload-pattern` is specified, in which case the payload is overwritten anyway.

Usage

```
tsp -P craft [options]
```

Options

`--clear-discontinuity`
Clear the *discontinuity_indicator* in the packets.

`--clear-error`
Clear the *transport_error_indicator* in the packets.

`--clear-es-priority`
Clear the *elementary_stream_priority_indicator* in the packets.

`--clear-priority`
Clear the *transport_priority* flag in the packets.

`--clear-pusi`
Clear the *payload_unit_start_indicator* in the packets.

`--clear-random-access`
Clear the *random_access_indicator* in the packets.

`--continuity-counter` *value*
Specify the value of the *continuity_counter* field.

`--discontinuity`
Set the *discontinuity_indicator* in the packets.
Space is required in the adaptation field.

`--error`
Set the *transport_error_indicator* in the packets.

`--es-priority`
Set the *elementary_stream_priority_indicator* in the packets.
Space is required in the adaptation field.

`--no-opcr`
Remove the OPCR from the packets.

`--no-payload`
Remove the payload.

`--no-pcr`
Remove the PCR from the packets.

`--no-private-data`
Remove the private data from adaptation field.

`--no-splice-countdown`
Remove the splicing point from the packets.

--offset-pattern *value*

Specify starting offset in payload when using `--payload-pattern`. By default, the pattern replacement starts at the beginning of the packet payload.

--opcr *value*

Set this OPCR value in the packets.

Space is required in the adaptation field.

--pack-pes-header

When a TS packet contains the start of a PES packet and the header of this PES packet contains stuffing, shift the TS payload to remove all possible stuffing from the PES header. Create TS stuffing in the adaptation field to compensate.

With PES data streams such as subtitles, the PES header sometimes contains stuffing to make sure that the PES packet uses an integral number of full TS packets. This option is a way to create space in the adaptation field of TS packets without destroying data. Then, PCR or other data can be added in the adaptation fields.

--payload-pattern *value*

Overwrite the payload and specify the binary pattern to apply. The value must be a string of hexadecimal digits specifying any number of bytes. The pattern is repeated to fill the payload.

--payload-size *size*

Resize the packet payload to the specified value in bytes.

When necessary, an adaptation field is created or enlarged. Without `--payload-pattern`, the existing payload is either shrunk or enlarged.

When an existing payload is shrunk, the end of the payload is truncated.

When an existing payload is enlarged, its end is padded with `0xFF` bytes.

Note that `--payload-size 0` specifies that a payload exists with a zero size. This is different from `--no-payload` which also specifies that the payload does not exist.

--pcr *value*

Set this PCR value in the packets.

Space is required in the adaptation field.

--pes-payload

With this option, the modified payload is the PES payload, not the TS payload. When the TS packet does not contain the start of a PES packet, the TS payload is not modified.

With `--payload-size`, the TS payload is resized so that the part of the PES payload which is in the TS packet gets the specified size.

With `--payload-pattern` and `--offset-pattern`, the pattern is applied inside the PES payload at the specified offset.

-p *value*
--pid *value*

Modify the PID to the specified value.

--priority

Set the transport_priority flag in the packets.

--private-data *value*

Specify the binary content of the *transport_private_data* in the adaptation field. The value must be a string of hexadecimal digits specifying any number of bytes.

Space is required in the adaptation field.

--pusi

Set the *payload_unit_start_indicator* in the packets.

--random-access

Set the *random_access_indicator* in the packets.

Space is required in the adaptation field.

--scrambling *value*

Specify the value of the *transport_scrambling_control* field.

--splice-countdown *value*

Create a splicing point and set this splice countdown value in the packets.

Space is required in the adaptation field.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

cutoff

Set labels on TS packets upon reception of UDP messages

This plugin set or clear labels on the TS packets upon reception of text commands from UDP.

This plugin is typically used as a remote command reception. Depending on the remote commands, packets are marked and can be processed differently in subsequent plugins in the chain.

Remote commands

The plugin *cutoff* listens to UDP datagrams on a given port. Each datagram contains exactly one command. A command is an ASCII string. Any trailing control characters such as CR or LF is ignored.

The command string can be one of:

`pulse-label` *n*	Set the label *n* on the next TS packet (only once).
`start-label` *n*	Set the label *n* on all TS packets (until the next `stop-label` command).
`stop-label` *n*	Stop setting the label *n* on all TS packets.
`exit`	Exit the *tsp* execution, simulate an end of stream at the next TS packet.

Note that the *bash*[4] shell provides an easy way to redirect output to an UDP message. The following sample commands send UDP messages on port 4444 to system 127.0.0.1 (the local host). This is the easiest way to control the plugin *cutoff*.

```
echo >/dev/udp/127.0.0.1/4444 pulse-label 1
echo >/dev/udp/127.0.0.1/4444 start-label 2
echo >/dev/udp/127.0.0.1/4444 stop-label 2
echo >/dev/udp/127.0.0.1/4444 exit
```

Usage

```
tsp -P cutoff [options] [[source@]address:]port
```

Parameter

The parameter [*address*:]*port* describes the destination of incoming UDP datagrams. All datagrams which are received on this stream are text commands.

The *port* part is mandatory and specifies the UDP port to listen on. The *address* part is optional. It specifies an IP multicast address to listen on. It can be also a host name that translates to a multicast address.

An optional source address can be specified as *source@address:port* in the case of source-specific multicast (SSM).

If the address is not specified, the plugin simply listens on the specified local port and receives the packets which are sent to one of the local (unicast) IP addresses of the system.

Options

`-a` *address*
`--allow` *address*

Specify an IP address or host name which is allowed to send remote commands. Several `--allow` options can be used to specify several allowed remote control systems.

By default, all received commands are accepted. If at least one `--allow` option is specified, any remote command which is not sent by an allowed host is rejected.

This is a security feature, but not a perfect one since IP address spoofing is trivial with UDP.

[4] This is a feature of *bash*, not a Linux feature. It is available on all platforms, including macOS or Cygwin.

-b *value*
--buffer-size *value*

Specify the UDP socket receive buffer size (socket option).

--default-interface

Let the system find the appropriate local interface on which to listen. By default, listen on all local interfaces.

-f
--first-source

Filter UDP packets based on the source address. Use the sender address of the first received packet as only allowed source.

This option is useful when several sources send packets to the same destination address and port. Accepting all commands could result in inconsistent processing and only one sender shall be accepted.

To allow a more precise selection of the sender, use option `--source`. Options `--first-source` and `--source` are mutually exclusive.

-l *address*
--local-address *address*

Specify the IP address of the local interface on which to listen. It can be also a host name that translates to a local address.

By default, listen on all local interfaces.

--max-queue *value*

Specify the maximum number of queued UDP commands before their execution into the stream.

The default is 128.

--no-reuse-port

Disable the reuse port socket option. Do not use unless completely necessary.

--receive-timeout *value*

Specify the UDP reception timeout in milliseconds. This timeout applies to each receive operation, individually.

By default, receive operations wait for commands, possibly forever.

-r
--reuse-port

Set the reuse port socket option. This is now enabled by default, the option is present for legacy only.

-s *address[:port]*
--source *address[:port]*

Filter UDP packets based on the specified source address.

This option is useful when several sources send packets to the same destination address and port. Accepting all commands could result in inconsistent processing and only one sender shall be accepted.

Options `--first-source` and `--source` are mutually exclusive.

--ssm

This option forces the usage of source-specific multicast (SSM) using the source address which is specified by the option `--source`. Without `--ssm`, standard ("any-source') multicast is used and the option `--source` is used to filter incoming packets.

The `--ssm` option is implicit when the classical SSM syntax *source@address:port* is used.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

datainject

DVB SimulCrypt EMM and Private Data Injector

This plugin receives EMM's and/or private data using the DVB SimulCrypt EMMG/PDG ⇔ MUX protocol and injects them into the transport stream in a specific PID.

This plugin is a TCP server (MUX side of the protocol). It accepts only one EMMG/PDG connection at a time.

If the injected data are EMM's, make sure to update the CAT accordingly (see the plugin *cat*).

Usage

```
tsp -P datainject [options]
```

Options

-b *value*
--bitrate-max *value*

Specifies the maximum bitrate for the data PID in bits / second. By default, the data PID bitrate is limited by the stuffing bitrate (data insertion is performed by replacing stuffing packets).

--buffer-size *value*

Specify the TCP and UDP socket receive buffer size in bytes (socket option).

-v *value*
--emmg-mux-version *value*

Specifies the version of the EMMG/PDG <=> MUX DVB SimulCrypt protocol.

Valid values are 1 to 5. The default is 2.

--log-data[=*level*]

Same as `--log-protocol` but applies to *data_provision* messages only.

To debug the session management without being flooded by data messages, use `--log-protocol=info --log-data=debug`.

--log-protocol[=*level*]

Log all EMMG/PDG ⇔ MUX protocol messages using the specified level. If the option is not present, the messages are logged at debug level only. If the option is present without value, the messages are logged at info level. A level can be a numerical debug level or any of the following: `fatal`, `severe`, `error`, `warning`, `info`, `verbose`, `debug`.

--no-reuse-port

Disable the reuse port socket option. Do not use unless completely necessary.

-p *value*
--pid *value*

Specifies the PID for the data insertion. This option is mandatory.

-q *value*
--queue-size *value*

Specifies the maximum number of data sections or TS packets in the internal queue, i.e. messages which are received from the EMMG/PDG client but not yet inserted into the transport stream.

The default is 1000.

-r
--reuse-port

Set the reuse port socket option. This is now enabled by default, the option is present for legacy only.

-s *[address:]port*
--server *[address:]port*

Specifies the local TCP port on which the plugin listens for an incoming EMMG/PDG connection. This option is mandatory.

When present, the optional address shall specify a local IP address or host name (by default, the plugin accepts connections on any local IP interface). This plugin behaves as a MUX, ie. a TCP server, and accepts only one EMMG/PDG connection at a time.

-u *[address:]port*
--udp *[address:]port*

Specifies the local UDP port on which the plugin listens for data provision messages (these messages can be sent using TCP or UDP).

By default, the UDP reception uses the same port and optional local address as specified for TCP using option `--server`.

--unregulated

Insert data packets immediately. Do not regulate the insertion of data packets, do not limit the data bitrate.

This is useful to test invalid EMMG's which do not comply with the allocated bitrate policy.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

decap

Decapsulate TS packets from a PID produced by the *encap* plugin

This plugin is the counterpart of the *encap* plugin. It decapsulates the original TS packets from a "tunnel" PID which was created by *encap*. See the documentation of the *encap* plugin for more details.

The decapsulated packets replace the tunnel PID. Because of the encapsulation overhead, the total volume of decapsulated packets is slightly smaller (approximately 2%) than the encapsulation PID. The packets in excess are replaced by null packets after decapsulation.

Usage

```
tsp -P decap [options]
```

Options

-i
--ignore-errors

Ignore errors such malformed encapsulated stream.

-p *value*
--pid *value*

Specify the input PID containing all encapsulated PID's. This is a mandatory parameter, there is no default.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

dektec (input)

Dektec DTA-1xx and DTU-2xx ASI and demodulator devices

This input plugin receives packets from a Dektec DTA-1xx or DTU-2xx DVB-ASI or demodulator device.

Using this plugin forces *tsp* and all plugins to use their real-time defaults (see the reference documentation for *tsp).*

Usage

```
tsp -I dektec [options]
```

Options

--atsc3-bandwidth *value*

ATSC demodulators: indicate the ATSC 3.0 bandwidth.

Must be one of "6-MHz", "7-MHz", "8-MHz". The default is 8-MHz.

--c2-bandwidth *value*

DVB-C2 demodulators: indicate the DVB-C2 bandwidth.

Must be one of "6-MHz", "8-MHz". The default is 8-MHz.

-c *value*
--channel *value*

Channel index on the input Dektec device. By default, use the first input channel on the device.

--code-rate *value*

For demodulators devices only: specify the code rate. The specified value depends on the modulation type.

DVB-S: "1/2", "2/3", "3/4", "4/5", "5/6", "6/7", "7/8".

DVB-S2: "1/2", "1/3", "1/4", "2/3", "2/5", "3/4", "3/5", "4/5", "5/6", "6/7", "7/8", "8/9", "9/10".

DVB-T: "1/2", "2/3", "3/4", "5/6", "7/8".

The value "auto" can be used to automatically detect the code rate. This is the default.

--constellation *value*

DVB-T demodulators: indicate the constellation type.

Must be one of "16-QAM", "64-QAM", "QPSK", "auto".

The value "auto" can be used to automatically detect the constellation. This is the default.

-d *value*
--device *value*

Device index, from 0 to N-1 (with N being the number of Dektec devices in the system). Use the command "tsdektec -a [-v]" to have a complete list of devices in the system. By default, use the first input Dektec device.

--dvbt-bandwidth *value*

DVB-T/T2 demodulators: indicate the bandwidth in MHz. The default is 8 MHz.

Must be one of "1.7", "10", "5", "6", "7", "8". The bandwidth values 1.7, 5 and 10 MHz are valid for DVB-T2 only.

-f *value*
--frequency *value*

For demodulator devices only: specify the frequency, in Hz, of the input carrier. There is no default.

For DVB-S/S2 receivers, the specified frequency is the *intermediate* frequency. For convenience, the option --satellite-frequency can be used instead of --frequency when the intermediate frequency is unknown. When --frequency is used with DVB-S/S2, the original

satellite frequency is unknown, it is impossible to determine if a high band is used and no “high band 22 kHz tone” is send to the LNB.

For DTA-2137 receivers, the valid range is 950 MHz to 2150 MHz (L Band).

--guard-interval *value*

DVB-T demodulators: indicate the guard interval.

Must be one of "1/16", "1/32", "1/4", "1/8", "auto". The default is “auto”.

--isdbt-bandwidth *value*

ISDB-T demodulators: indicate the bandwidth in MHz.

Must be one of "5", "6", "7", "8". The default is 8 MHz.

--isdbt-segments *value*

ISDB-T demodulators: indicate the number of segments.

Must be one of "1", "3" or "13". The default is 1.

--isdbt-subchannel *value*

ISDB-T demodulators: indicate the sub-channel number (0..41) of the centre segment of the spectrum. The default is 22.

--j83 *value*

QAM demodulators: indicate the ITU-T J.83 annex to use. Must be one of "A", "B", "C".

A is DVB-C, B is “American QAM”, C is “Japanese QAM”. The default is A.

--lnb *string*

DVB-S/S2 receivers: description of the LNB which is used to convert the `--satellite-frequency` into an *intermediate* frequency. This option is useless when `--satellite-frequency` is not specified.

The format of the string is "low_freq[,high_freq,switch_freq]" where all frequences are in MHz.

The characterisctics of the default universal LNB are low_freq = 9750 MHz, high_freq = 10600 MHz, switch_freq = 11700 MHz.

-m *value*
--modulation *value*

For demodulators, indicate the modulation type. The supported modulation types depend on the device model. The default modulation type is DVB-S.

Must be one of "ATSC-VSB", "ATSC-3.0", "DAB", "DVB-C2", "DVB-S", "DVB-S-QPSK" (same as "DVB-S"), "DVB-S2", "DVB-S2-QPSK" (same as "DVB-S2"), "DVB-S2-8PSK", "DVB-S2-16APSK", "DVB-S2-32APSK", "DVB-T", "DVB-T2", "ISDB-T", "QAM" (auto-detection of QAM type), "128-QAM", "16-QAM", "256-QAM", "32-QAM", "64-QAM".

--polarity *value*

DVB-S/S2 receivers: indicate the polarity.

Must be one of "horizontal", "vertical". The default is "vertical".

--qam-b *value*

QAM demodulators: with `--j83 B`, indicate the QAM-B interleaver mode.

Must be one of "I8-J16", "I16-J8", "I32-J4", "I64-J2", "I128-J1", "I128-J1D", "I128-J2", "I128-J3", "I128-J4", "I128-J5", "I128-J6", "I128-J7", "I128-J8", "auto". The default is “auto”.

-t *value*
--receive-timeout *value*

Specify the data reception timeout in milliseconds. This timeout applies to each receive operation, individually. By default, receive operations wait for data, possibly forever.

--satellite-frequency *value*

DVB-S/S2 receivers: indicate the target satellite frequency, in Hz, of the input carrier. The actual frequency at the input of the receiver is the *intermediate* frequency which is computed based on the characteristics of the LNB (see option `--lnb`). This option is useful when the satellite frequency is better known than the intermediate frequency.

The options `--frequency` and `--satellite-frequency` are mutually exclusive.

--satellite-number *value*

DVB-S/S2 receivers: indicate the satellite/dish number.

Must be 0 to 3 with DiSEqC switches and 0 to 1 for non-DiSEqC switches. The default is 0.

--symbol-rate *value*

DVB-C/S/S2 demodulators: Specify the symbol rate in symbols/second. By default, automatically detect the symbol rate.

--t2-profile *value*

DVB-T2 demodulators: indicate the DVB-T2 profile.

Must be one of "`base`", "`lite`". The default is “`base`”.

--transmission-mode *value*

DVB-T demodulators: indicate the transmission mode.

Must be one of "`2K`", "`8K`", "`auto`". The default is “`auto`”.

--vsb *value*

ATSC demodulators: indicate the VSB constellation.

Must be one of "`8`", "`16`". The default is 8.

Generic common input plugin options

The following options are implicitly defined in all input plugins.

--help

Display plugin help text.

dektec (output)

Dektec DTA-1xx and DTU-2xx ASI and modulator devices

This output plugin sends packets to a DVB-ASI Dektec DTA-1xx or DTU-2xx device or a Dektec DTA-1xx modulator.

Using this plugin forces *tsp* and all plugins to use their real-time defaults (see the reference documentation for *tsp).*

Usage

```
tsp -O dektec [options]
```

Overview of options

For multi-standard modulators such as the DTA-115, the type of required modulation must be specified if it is different from the default modulation. See Table 3 for the default modulation type by device model.

Table 3: Dektec modulators default modulation types

Device model	Default modulation
DTA-107	DVB-S (QPSK)
DTA-107.S2	DVB-S2 (QPSK)
DTA-110	DVB-C (64-QAM)
DTA-110T	DVB-T
DTA-115	DVB-T

Depending on the type of output, the combination of required and optional options is different. See Table 4 for the applicability of options by modulation type. The modulation type is specified using option `--modulation`. Mandatory options are marked using (*).

Table 4: Command line options for Dektec modulators

Modulation	Applicable options
All (common options)	`--bitrate --channel --device --stuffing --fifo-size`
DVB-ASI	`--204`
All except DVB-ASI	`--frequency --instant-detach --inversion --level --modulation --offset-count --uhf-channel --vhf-channel`
x-QAM	`--j83 --qam-b`
ADBT-T, DMB-T/H	`--bandwidth --dmb-constellation --dmb-fec --dmb-frame-numbering --dmb-header --dmb-interleaver --pilots`
ATSC	`--vsb --vsb-taps`
CMMB	`--cmmb-area-id --cmmb-bandwidth --cmmb-pid(*) --cmmb-transmitter-id`
DVB-S	`--convolutional-rate --lnb --satellite-frequency --symbol-rate`
DVB-S2	`--convolutional-rate --lnb --pilots --s2-gold-code --s2-short-fec-frame --satellite-frequency --symbol-rate`
DVB-T	`--bandwidth --cell-id --constellation --convolutional-rate --guard-interval --indepth-interleave --mpe-fec --time-slice --transmission-mode`

Modulation	Applicable options
DVB-T2	`--bandwidth --bandwidth-extension --cell-id –fef --fef-interval --fef-length --fef-s1 --fef-s2 --fef-signal --fef-type --fft-mode --miso –papr --pilot-pattern --plp0-code-rate --plp0-fec-type --plp0-group-id --plp0-high-efficiency --plp0-id --plp0-il-length --plp0-il-type --plp0-in-band --plp0-issy --plp0-modulation --plp0-null-packet-deletion --plp0-rotation --plp0-type --t2-fpsf --t2-guard-interval --t2-l1-modulation --t2-network-id --t2-system-id`
ISDB-T	*not supported yet*

Detailed options

`--204`

For DVB-ASI devices only: Send 204-byte packets (188 meaningful bytes plus 16 stuffing bytes for Reed-Solomon coding). By default, send 188-byte packets.

`--bandwidth` value

DVB-T/H, DVB-T2, ADTB-T and DMB-T/H modulators: indicate bandwidth in MHz. Must be one of "`1.7`", "`5`", "`6`", "`7`", "`8`" or "`10`". The default is 8 MHz. The bandwidth values 1.7 and 10 MHz are valid for DVB-T2 only.

`--bandwidth-extension`

DVB-T2 modulators: indicate that the extended carrier mode is used. By default, use normal carrier mode.

`-b` *value*
`--bitrate` *value*

Specify output bitrate in bits/second. By default, use the input device bitrate or, if the input device cannot report bitrate, analyze some PCR's at the beginning of the input stream to evaluate the original bitrate of the transport stream.

`--cell-id` *value*

DVB-T and DVB-T2 modulators: indicate the cell identifier to set in the transmission parameters signaling (TPS). Disabled by default with DVB-T. Default value is `0` with DVB-T2.

`-c` *value*
`--channel` *value*

Channel index on the output Dektec device. By default, use the first output channel on the device.

`--cmmb-area-id` *value*

CMMB modulators: indicate the area id. The valid range is 0 to 127. The default is zero.

`--cmmb-bandwidth` *value*

CMMB modulators: indicate bandwidth in MHz. Must be one of "`2`" or "`8`". The default is 8 MHz.

`--cmmb-pid` *value*

CMMB modulators: indicate the PID of the CMMB stream in the transport stream. This is a required parameter for CMMB modulation.

`--cmmb-transmitter-id` *value*

CMMB modulators: indicate the transmitter id. The valid range is 0 to 127. The default is zero.

`--constellation` *value*

DVB-T modulators: indicate the constellation type. Must be one of "`QPSK`", "`16-QAM`", "`64-QAM`". The default is 64-QAM.

-r *rate*
--convolutional-rate *rate*

For modulators devices only: specify the convolutional rate. The specified value depends on the modulation type. The default is "3/4".

DVB-S: "1/2", "2/3", "3/4", "4/5", "5/6", "6/7", "7/8".

DVB-S2: "1/2", "1/3", "1/4", "2/3", "2/5", "3/4", "3/5", "4/5", "5/6", "6/7", "7/8", "8/9", "9/10".

DVB-T: "1/2", "2/3", "3/4", "5/6", "7/8".

-d *value*
--device *value*

Device index, from 0 to N-1 (with N being the number of Dektec devices in the system). Use the command "`tsdektec -a [-v]`" to have a complete list of devices in the system. By default, use the first output Dektec device.

--dmb-constellation *value*

DMB-T/H, ADTB-T modulators: indicate the constellation type. Must be one of: "`4-QAM-NR`", "`4-QAM`", "`16-QAM`", "`32-QAM`", "`64-QAM`". The default is 64-QAM. 4-QAM-NR and 32-QAM can be used only with `--dmb-fec 0.8`.

--dmb-fec *value*

DMB-T/H, ADTB-T modulators: indicate the FEC code rate. Must be one of "`0.4`", "`0.6`", "`0.8`". The default is 0.8.

--dmb-frame-numbering

DMB-T/H, ADTB-T modulators: indicate to use frame numbering. The default is to use no frame numbering.

--dmb-header *value*

DMB-T/H, ADTB-T modulators: indicate the FEC frame header mode. Must be one of "`PN420`", "`PN595`" (ADTB-T only) or "`PN945`". The default is PN945.

--dmb-interleaver *value*

DMB-T/H, ADTB-T modulators: indicate the interleaver mode. Must be one "`1`" (B=54, M=240) or "`2`" (B=54, M=720). The default is 1.

--fef

DVB-T2 modulators: enable insertion of FEF's (Future Extension Frames). Not enabled by default.

--fef-interval *value*

DVB-T2 modulators: indicate the number of T2 frames between two FEF parts. The valid range is 1 to 255 and `--t2-fpsf` shall be divisible by `--fef-interval`. The default is 1.

--fef-length *value*

DVB-T2 modulators: indicate the length of a FEF-part in number of T-units (= samples). The valid range is 0 to `0x3FFFFF`. The default is 1.

--fef-s1 *value*

DVB-T2 modulators: indicate the S1-field value in the P1 signalling data. Valid values: 2, 3, 4, 5, 6 and 7. The default is 2.

--fef-s2 *value*

DVB-T2 modulators: indicate the S2-field value in the P1 signalling data. Valid values: 1, 3, 5, 7, 9, 11, 13 and 15. The default is 1.

--fef-signal *value*

DVB-T2 modulators: indicate the type of signal generated during the FEF period. Must be one of "`0`" (zero I/Q samples during FEF), "`1K`" (1K OFDM symbols with 852 active carriers containing BPSK symbols, same PRBS as the T2 dummy cells, not reset between symbols) or "`1K-384`" (1K OFDM symbols with 384 active carriers containing BPSK symbols). The default is `0`.

--fef-type *value*

DVB-T2 modulators: indicate the FEF type. The valid range is 0 ... 15. The default is 0.

--fft-mode *value*

DVB-T2 modulators: indicate the FFT mode. Must be one of "1K", "2K", "4K", "8K", "16K" or "32K". The default is 32K.

--fifo-size *value*

Set the FIFO size in bytes of the output channel in the Dektec device. The default value depends on the device type.

-f *value*
--frequency *value*

For modulator devices only: specify the frequency, in Hz, of the output carrier. There is no default.

For OFDM modulators, the options `--uhf-channel` or `--vhf-channel` and `--offset-count` (optional) may be used instead.

For DVB-S/S2 modulators, the specified frequency is the *intermediate* frequency. For convenience, the option `--satellite-frequency` can be used instead of `--frequency` when the intermediate frequency is unknown.

For DTA-107 (DVB-S) modulators, the valid range is 950 MHz to 2150 MHz.

For DTA-110 (DVB-C) and 110T (DVB-T/H) modulators, the valid range is 400 MHz to 862 MHz.

For DTA-115 (DVB-C/T/H) modulators, the valid range is 47 MHz to 862 MHz.

-g *value*
--guard-interval *value*

DVB-T modulators: indicate the guard interval. Must be one of: "1/32", "1/16", "1/8", "1/4". The default is 1/32.

--hf-band-region *name*

Specify the region for UHF/VHF band frequency layout.

The default region is "europe". Another default region may be specified per user in the TSDuck configuration file (see Appendix A).

--indepth-interleave

DVB-T modulators: use in-depth interleave. The default is native interleave.

-i
--input-modulation

All modulators devices: try to guess default modulation parameters from input stream. All explicitely specified parameters override these defaults.

If the input plugin is dvb, use the modulation parameters of the input signal as default values for their counterparts in the Dektec modulator. On Linux systems, the actual modulation parameters of the input signal are used. On Windows systems, the DirectShow/BDA drivers cannot return the actual modulation parameters and only the user-specified parameters in the input plugin are used (they can be different from the actual parameters of the input signal).

With other input plugins, if the specified output modulation is DVB-T, try to guess the following modulation parameters from the input bitrate: `--bandwidth --constellation --convolutional-rate --guard-interval`. When a specific bitrate can be produced by distinct combinations of modulation parameters, a deterministic order is applied to select the prefered combination.

--instant-detach

At end of stream, perform an "*instant detach*" of the output channel. The default is to wait until all bytes are sent. The default is fine for ASI devices. With modulators, the "*wait until sent*" mode may hang at end of stream and `--instant-detach` avoids this.

--inversion

For modulators devices only: enable spectral inversion.

--j83 *value*

QAM modulators: indicate the ITU-T J.83 annex to use. Must be one of "A" (DVB-C), "B" (American QAM) or "C" (Japanese QAM). The default is A.

-l *value*
--level *value*

Modulators: indicate the output level in units of 0.1 dBm (e.g. `--level -30` means -3 dBm). Not supported by all devices.

For DTA-107 modulators, the valid range is -47.0 to -27.0 dBm.

For DTA-115, QAM, the valid range is -35.0 to 0.0 dBm.

For DTA-115, OFDM, ISDB-T, the valid range is -38.0 to -3.0 dBm.

--lnb *string*

DVB-S/S2 modulators: description of the LNB which is used to convert the `--satellite-frequency` into an *intermediate* frequency. This option is useless when `--satellite-frequency` is not specified.

The format of the string is "`low_freq[,high_freq,switch_freq]`" where all frequences are in MHz.

The characterisctics of the default universal LNB are `low_freq` = 9750 MHz, `high_freq` = 10600 MHz, `switch_freq` = 11700 MHz.

--maintain-preload

If the FIFO were preloaded (see option `--preload-fifo`), roughly maintain the FIFO buffer size in order to maintain the delay from real-time. If the FIFO size drops to zero bytes, pause transmission till it gets back to the preload FIFO size.

--miso *value*

DVB-T2 modulators: indicate the MISO mode. Must be one of "`OFF`", "`1`", "`2`" or "`BOTH`". The default si OFF. This mode can be used to simulate antenna 1, antenna 2 or the average of antenna 1 and antenna 2 to simulate reception halfway between the antennas.

-m *value*
--modulation *value*

For modulators, indicate the modulation type. Must be one of: "`4-QAM`", "`16-QAM`", "`32-QAM`", "`64-QAM`", "`128-QAM`", "`256-QAM`", "`ADTB-T`", "`ATSC-VSB`", "`CMMB`", "`DMB-T`", "`DVB-S`", "`DVB-S-QPSK`" (same as DVB-S), "`DVB-S-BPSK`", "`DVB-S2`", "`DVB-S2-QPSK`" (same as DVB-S2), "`DVB-S2-8PSK`", "`DVB-S2-16APSK`", "`DVB-S2-32APSK`", "`DVB-T`", "`DVB-T2`", "`ISDB-T`". For DVB-H, specify DVB-T. For DMB-H, specify DMB-T.

The supported modulation types depend on the device model. See Table 3 above for the default modulation type by device model.

--mpe-fec

DVB-T/H modulators: indicate that at least one elementary stream uses MPE-FEC (DVB-H signalling).

-o *value*
--offset-count *value*

UHF and VHF modulators: specify the number of offsets from the UHF or VHF channel. Can be positive or negative. The default is zero. See options `--uhf-channel` and `--vhf-channel`.

--papr *value*

DVB-T2 modulators: indicate the Peak to Average Power Reduction method. Must be one of "`NONE`", "`ACE`" (Active Constellation Extension), "`TR`" (power reduction with reserved carriers) or "`BOTH`" (both ACE and TS). The default is NONE.

--pilots

DVB-S2 and ADTB-T modulators: enable pilots (default: no pilot).

-p *value*
--pilot-pattern *value*

DVB-T2 modulators: indicate the pilot pattern to use, a value in the range 1 to 8. The default is 7.

--plp0-code-rate *value*

DVB-T2 modulators: indicate the convolutional coding rate used by the PLP #0. Must be one of "1/2", "3/5", "2/3", "3/4", "4/5", "5/6". The default is 2/3.

--plp0-fec-type *value*

DVB-T2 modulators: indicate the FEC type used by the PLP #0. Must be one of "16K", "64K". The default is 64K LPDC.

--plp0-group-id *value*

DVB-T2 modulators: indicate the PLP group with which the PLP #0 is associated. The valid range is 0 to 255. The default is 0.

--plp0-high-efficiency

DVB-T2 modulators: indicate that the PLP #0 uses High Efficiency Mode (HEM). Otherwise Normal Mode (NM) is used.

--plp0-id *value*

DVB-T2 modulators: indicate the unique identification of the PLP #0 within the T2 system. The valid range is 0 to 255. The default is 0.

--plp0-il-length *value*

DVB-T2 modulators: indicate the time interleaving length for PLP #0. The valid range is 0 to 255. The default is 3.

If `--plp0-il-type` is set to "ONE-TO-ONE" (the default), this parameter specifies the number of TI-blocks per interleaving frame.

If `--plp0-il-type` is set to "MULTI", this parameter specifies the number of T2 frames to which each interleaving frame is mapped.

--plp0-il-type *value*

DVB-T2 modulators: indicate the type of interleaving used by the PLP #0. Must be one of "ONE-TO-ONE" (one interleaving frame corresponds to one T2 frame) or "MULTI" (one interleaving frame is carried in multiple T2 frames). The default is ONE-TO-ONE.

--plp0-in-band

DVB-T2 modulators: indicate that the in-band flag is set and in-band signalling information is inserted in PLP #0.

--plp0-issy *value*

DVB-T2 modulators: type of ISSY field to compute and insert in PLP #0. Must be one of "NONE", "SHORT", "LONG". The default is NONE.

--plp0-modulation *value*

DVB-T2 modulators: indicate the modulation used by PLP #0. Must be one of "BPSK", "QPSK", "16-QAM", "64-QAM", "256-QAM". The default is 256-QAM.

--plp0-null-packet-deletion

DVB-T2 modulators: indicate that null-packet deletion is active in PLP #0. Otherwise it is not active.

--plp0-rotation

DVB-T2 modulators: indicate that constellation rotation is used for PLP #0. Otherwise not.

--plp0-type *value*

DVB-T2 modulators: indicate the PLP type for PLP #0. Must be one of "COMMON", "1", "2". The default is COMMON.

--preload-fifo

Preload FIFO (hardware buffer) before starting transmission.

Preloading the FIFO will introduce a variable delay to the start of transmission, *if* the delivery of packets to the plug-in is pre-regulated, based on the size of the FIFO, the TS bit rate, and the size of the FIFO to preload, as controlled by the `--preload-fifo-percentage` or `--preload-fifo-delay` options.

If the delivery of packets to the plug-in isn't self-regulated (i.e. they are delivered faster than real-time, as might occur when loading from file), there is no benefit to preloading the FIFO, because in that case, the FIFO will fill up quickly anyway.

This option is implicitly set when using a modulator for output.

--preload-fifo-delay *value*

The use of this option indicates that the size of the FIFO to preload prior to starting transmission should be calculated based on the specified delay, in milliseconds, and the configured bit rate. That is, transmission will start after the specified delay worth of media has been preloaded.

This option takes precedence over the `--preload-fifo-percentage` option.

There is no default value, and the valid range is 100-100000.

--preload-fifo-percentage *value*

Percentage of size of FIFO to preload prior to starting transmission (default: 80%).

-q *value*
--qam-b *value*

QAM modulators: with `--j83 B`, indicate the QAM-B interleaver mode. Must be one of: "`I128-J1D`", "`I64-J2`", "`I32-J4`", "`I16-J8`", "`I8-J16`", "`I128-J1`", "`I128-J2`", "`I128-J3`", "`I128-J4`", "`I128-J5`", "`I128-J6`", "`I128-J7`", "`I128-J8`". The default is I128-J1D.

--s2-gold-code *value*

DVB-S2 modulators: indicate the physical layer scrambling initialization sequence, aka "gold code".

--s2-short-fec-frame

DVB-S2 modulators: use short FEC frames, 12 000 bits (default: long FEC frames, 64 800 bits).

--satellite-frequency *value*

DVB-S/S2 modulators: indicate the target satellite frequency, in Hz, of the output carrier. The actual frequency at the output of the modulator is the *intermediate* frequency which is computed based on the characteristics of the LNB (see option `--lnb`). This option is useful when the satellite frequency is better known than the intermediate frequency.

The options `--frequency` and `--satellite-frequency` are mutually exclusive.

-s
--stuffing

Automatically generate stuffing packets if tsp fails to provide packets fast enough.

This option applies only to ASI, SDI and hardware-based modulators (DVB-C, DVB-S). This option is ineffective on modulators which are partially software-based (DVB-T on DTA-110T or DTA-115).

--symbol-rate *value*

DVB-C/S/S2 modulators: Specify the symbol rate in symbols/second.

By default, the symbol rate is implicitly computed from the convolutional rate, the modulation type and the bitrate. But when `--symbol-rate` is specified, the input bitrate is ignored and the output bitrate is forced to the value resulting from the combination of the specified symbol rate, convolutional rate and modulation type.

The options `--symbol-rate` and `--bitrate` are mutually exclusive.

--t2-fpsf *value*

DVB-T2 modulators: indicate the number of T2 frames per super-frame. Must be in the range 1 to 255. The default is 2.

--t2-guard-interval *value*

DVB-T2 modulators: indicates the guard interval. Must be one of: "1/128", "1/32", "1/16", "19/256", "1/8", "19/128", "1/4". The default is 1/128.

--t2-l1-modulation *value*

DVB-T2 modulators: indicate the modulation type used for the L1-post signalling block. Must be one of "BPSK", "QPSK", "16-QAM", "64-QAM". The default is 16-QAM.

--t2-network-id *value*

DVB-T2 modulators: indicate the DVB-T2 network identification. The default is 0.

--t2-system-id *value*

DVB-T2 modulators: indicate the DVB-T2 system identification. The default is 0.

--time-slice

DVB-T/H modulators: indicate that at least one elementary stream uses time slicing (DVB-H signalling).

-t *value*
--transmission-mode *value*

DVB-T modulators: indicates the transmission mode. Must be one of "2K", "4K" or "8K". The default is 8K.

-u *value*
--uhf-channel *value*

UHF modulators: specify the UHF channel number of the output carrier. Can be used in replacement to `--frequency`. Can be combined with an `--offset-count` option. The UHF frequency layout depends on the region, see `--hf-band-region` option.

-v *value*
--vhf-channel *value*

VHF modulators: specify the VHF channel number of the output carrier. Can be used in replacement to `--frequency`. Can be combined with an `--offset-count` option. The VHF frequency layout depends on the region, see `--hf-band-region` option.

--vsb *value*

ATSC modulators: indicate the VSB constellation. Must be one of "8" (19,392,658 Mb/s) or "16" (38,785,317 Mb/s). The default is 8.

--vsb-taps *value*

ATSC modulators: indicate the number of taps of each phase of the root-raised cosine filter that is used to shape the spectrum of the output signal. The number of taps can have any value between 2 and 256 (the implementation is optimized for powers of 2). Specifying more taps improves the spectrum, but increases processor overhead. The recommend (and default) number of taps is 64 taps. If insufficient CPU power is available, 32 taps produces acceptable results, too.

Generic common output plugin options

The following options are implicitly defined in all output plugins.

--help

Display plugin help text.

descrambler

Generic DVB Descrambler

This plugin descrambles fixed PID's with fixed control words.

As a demo, it can also descramble services for which clear ECM's were generated using the utility named *tsecmg*, a DVB SimulCrypt-compliant ECMG for test and demo.

Usage

```
tsp -P descrambler [options] [service]
```

Parameter

The optional parameter specifies the service to descramble. If no fixed control word is specified, ECM's from the service are used to extract control words.

In the absence of explicit option such as `--atis-idsa`, `--dvb-cissa`, `--aes-cbc` or `--dvb-csa2`, the descrambling type is based on the *scrambling_descriptor* in the PMT of the service (if there is one).

If the argument is an integer value (either decimal or hexadecimal), it is interpreted as a service id. Otherwise, it is interpreted as a service name, as specified in the SDT. The name is not case sensitive and blanks are ignored. If the input TS does not contain an SDT, use service ids only.

Options

`--cas-id` *value*

Specify the *CA_system_id* to filter when searching for ECM streams. Since this descrambler is a demo tool using clear ECM's, it is unlikely that other real ECM streams exist. So, by default, any ECM stream is used to get the clear ECM's.

`-p` *pid1*`[-`*pid2*`]`
`--pid` *pid1*`[-`*pid2*`]`

Descramble packets with these PID values. Several `-p` or `--pid` options may be specified.

By default, descramble the specified service.

`--synchronous`

Specify to synchronously decipher the ECM's.

In real-time mode, the processing of packets continues in parallel while ECM's are deciphered. Use this option to force the stream processing to wait for ECM's at the point where the each ECM is received.

In offline mode, this option is always on. This is usually the right thing to do. Otherwise, if an ECM takes too long to be deciphered, the stream processing may reach the next crypto-period before the control word is available.

Note: this plugin only processes clear ECM's as generated by *tsecmg*. These ECM's are not ciphered and their processing is immediate. So, this option is useless in practice. However, this plugin is based on a generic descrambler implementation. For other conditional access systems, processing an ECM may be delegated to a smartcard and take a relatively long time. So, this option can be useful in that case.

Transport stream scrambling options

`--aes-cbc`

Use AES-CBC scrambling instead of DVB-CSA2 (the default).

The control words are 16-byte long instead of 8-byte. The residue is left clear. Specify a fixed initialization vector using the `--iv` option.

Note that this is a non-standard TS scrambling mode. The only standard AES-based scrambling modes are ATIS-IDSA and DVB-CISSA (DVB-CISSA is the same as AES-CBC with a DVB-defined IV).

A *scrambling_descriptor* is automatically added to the PMT of the service to indicate the use of AES-CBC scrambling. Since there is no standard value for AES-CBC, the user-defined *scrambling_mode* value `0xF0` is used.

--aes-ctr

Use AES-CTR scrambling instead of DVB-CSA2 (the default).

The control words are 16-byte long instead of 8-byte. The residue is included in the scrambling. Specify a fixed initialization vector using the `--iv` option. See the option `--ctr-counter-bits` for the size of the counter part in the IV.

Note that this is a non-standard TS scrambling mode. The only standard AES-based scrambling modes are ATIS-IDSA and DVB-CISSA.

A *scrambling_descriptor* is automatically added to the PMT of the service to indicate the use of AES-CTR scrambling. Since there is no standard value for AES-CTR, the user-defined *scrambling_mode* value `0xF1` is used.

--atis-idsa

Use ATIS-IDSA descrambling (ATIS-0800006) instead of DVB-CSA2 (the default).

The control words are 16-byte long instead of 8-byte.

--ctr-counter-bits *value*

With `--aes-ctr`, specifies the size in bits of the counter part.

In the initialization vector, the fixed nounce part uses the first 128 - *N* bits and the counter part uses the last *N* bits.

By default, the counter part uses the second half of the IV (64 bits).

-c *value*
--cw *value*

Specifies a fixed and constant control word (no crypto-period scheduling, no ECM insertion). The value must be a string of 16 hexadecimal digits (32 digits with `--atis-idsa` or `--dvb-cissa`).

--dvb-cissa

Use DVB-CISSA descrambling (see [16]) instead of DVB-CSA2 (the default).

The control words are 16-byte long instead of 8-byte.

--dvb-csa2

Use DVB-CSA2 descrambling. This is the default.

-f *name*
--cw-file *name*

Specifies a text file containing the list of control words to apply. Each line of the file must contain exactly 16 hexadecimal digits (32 digits with `--atis-idsa` or `--dvb-cissa`).

The next control word is used each time a new *transport_scrambling_control* value is found in the header of a TS packet. At the end of the list of control words, restart with the first one.

--iv *value*

With `--aes-cbc` or `--aes-ctr`, specifies a fixed initialization vector for all TS packets.

The value must be a string of 32 hexadecimal digits. The default IV is all zeroes.

-n
--no-entropy-reduction

Do not perform DVB-CSA2 control word entropy reduction to 48 bits, keep full 64-bit control words. This option is ignored with `--atis-idsa` or `--dvb-cissa`.

--output-cw-file *name*

Specifies a text file to create with all control words. Each line of the file will contain a control word with 16 or 32 hexadecimal digits, depending on the scrambling algorithm. Each time a new control word is used to descramble packets, it is logged in the file.

This option is specifically useful when the control words are dynamically extracted from ECM's. The created file can be used later using `--cw-file` to perform a direct descrambling test.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

drop (output)

Drop Output Packets

This output plugin simply drops all packets. This plugin is useful when the interesting work is done by the various packet processing plugins and the actual output packets are useless.

Usage

```
tsp -O drop [options]
```

Generic common output plugin options

The following options are implicitly defined in all output plugins.

--help

Display plugin help text.

duplicate

Duplicate PID's, reusing null packets

This plugin duplicates the content of several PID's into new PID's. The duplicated packets are created by replacing existing null packets. The input stream shall consequently contain at least as many null packets as packets to duplicate.

Usage

```
tsp -P duplicate [options] [pid[-pid]=newpid ...]
```

Specifying PID duplication

Each duplication is specified as "*pid=newpid*" or "*pid1-pid2=newpid*". All PID's can be specified as decimal or hexadecimal values. More than one PID duplication can be specified.

In the first form, the PID *pid* is duplicated as *newpid*.

In the latter form, all PID's within the range *pid1* to *pid2* (inclusive) are respectively duplicated as *newpid*, *newpid*+1, etc. (this behaviour is changed using option `--single`).

The null PID `0x1FFF` cannot be duplicated.

Options

`-d`
`--drop-overflow`

Silently drop overflow packets. By default, overflow packets trigger warnings.

See also option `--max-buffered-packets`.

`-m` *value*
`--max-buffered-packets` *value*

Specify the maximum number of buffered packets. The input packets to duplicate are internally buffered until a null packet is found and replaced by the buffered packet. An overflow is usually caused by insufficient null packets in the input stream.

The default is 1,024 packets.

`--reset-label` *label1*`[-`*label2*`]`

Clear the specified labels on the duplicated packets.

Several `--reset-label` options may be specified.

`--set-label` *label1*`[-`*label2*`]`

Set the specified labels on the duplicated packets.

Several `--set-label` options may be specified.

`-s`
`--single`

When a duplication is in the form "*pid1-pid2=newpid*", duplicate all input PID's within the range *pid1* to *pid2* to the same *newpid* value, not *newpid*, *newpid*+1, etc.

This option forces `--unchecked` since distinct PID's are duplicated to the same one.

`-u`
`--unchecked`

Do not perform any consistency checking while duplicating PID's. Duplicating two PID's to the same PID or to a PID which is already present in the input is accepted.

Note that this option should be used with care since the resulting stream can be illegal or inconsistent.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

dvb (input)

DVB-S, DVB-S2, DVB-C, DVB-T Devices Input

This input plugin receives TS packets from a DVB receiver device. These devices include a wide range of DVB-S, DVB-S2, DVB-C and DVB-T adapters. Most of them are simple tuners. See section 6.1 for more details on DVB receiver devices.

Using this plugin forces *tsp* and all plugins to use their real-time defaults (see the reference documentation for *tsp).*

Usage

```
tsp -I dvb [options]
```

General options

-a *N*
--adapter *N*

Specify the *N*th DVB adapter in the system, the first index being zero. This option can be used instead of device name.

On Linux systems, this means `/dev/dvb/adapterN`.

-d *"name"*
--device-name *"name"*

Specify the name of the DVB receiver device to use. Use the `tslsdvb` utility to list all available devices. By default, the first DVB receiver device is used. The syntax of the device name depends on the operating system. See section 6.1.3, page 285, for more details on DVB receiver devices naming.

--lnb *string*

Used for DVB-S and DVB-S2 tuners only.

For satellite reception, specifies the description of the LNB ("low-noise block" in the dish). The default value describes a dual-band so-called "universal LNB".

The format of the string is "`low_freq[,high_freq,switch_freq]`" where all frequencies are in MHz. The last two values are used only with a dual-band LNB. See the details in the table below.

Table 5: LNB settings for various bands

Field	Description	Universal	C-band	Ku-band	DBS
`low_freq`	Local oscillator frequency for lower band (or unique band)	`9750`	`5150`	`10750`	`11250`
`high_freq`	Local oscillator frequency for higher band (dual-band only)	`10600`			
`switch_freq`	Limit between lower and higher band (dual-band only)	`11700`			

--receive-timeout *milliseconds*

Specify the timeout, in milliseconds, for each receive operation. To disable the timeout and wait indefinitely for packets, specify zero. This is the default.

--signal-timeout *seconds*

Specify the timeout, in seconds, for the DVB frontend signal locking. If no signal is detected within this timeout, the command aborts. To disable the timeout and wait indefinitely for the signal, specify zero. The default is 5 seconds.

Linux-specific options

--demux-buffer-size *value*

Default buffer size, in bytes, of the demux device. The default is 1 MB.

Windows-specific options

--demux-queue-size *value*

Specify the maximum number of media samples in the queue between the DirectShow capture thread and the input plugin thread. The default is 1000 media samples.

Tuning

By default, no tuning is performed on the DVB frontend. The transponder on which the frontend is currently tuned is used.

There are two ways to specify a new transponder:

- Specifying individual tuning options, one for each tuning parameters. Common values are provided as default.
- The name of a channel contained in the transponder, using a channels configuration file. See Appendix B, page 293, for more details on channels configuration files.

Tuning method 1: Individual tuning options

--bandwidth *value*

Used for DVB-T/T2 tuners only.

Must be one of "auto", "8-MHz", "7-MHz" or "6-MHz".

For DVB-T2, also accept "5-MHz", "10-MHz" or "1.712-MHz".

The default is "8-MHz".

--delivery-system *value*

Used for DVB-S and DVB-S2 tuners only.

Specify which delivery system to use. Must be one of "DVB-S", "DVB-S2". The default is "DVB-S".

--fec-inner *value*

Used for DVB-S, DVB-S2 and DVB-C tuners only.

Specify the Inner Forward Error Correction. Must be one of "none", "auto", "1/2", "1/3", "1/4", "2/3", "2/5", "3/4", "3/5", "4/5", "5/6", "5/11", "6/7", "7/8", "8/9", "9/10". The default is "auto".

-f value
--frequency *value*

Specify the carrier frequency in Hz (all tuners).

For DVB-T tuners, the options --uhf-channel or --vhf-channel (and associated optional --offset-count) can be used instead of --frequency.

--guard-interval *value*

Used for DVB-T/T2 tuners only.

Must be one of "auto", "1/32", "1/16", "1/8", "1/4".

For DVB-T2, also accept "1/128", "19/128", "19/256".

The default is "1/32".

--hf-band-region *name*

Specify the region for UHF/VHF band frequency layout.

The default region is “europe". Another default region may be specified per user in the TSDuck configuration file (see Appendix A).

--hierarchy *value*

Used for DVB-T tuners only.

Must be one of "auto", "none", "1", "2", "4". The default is "none".

--high-priority-fec *value*

Used for DVB-T tuners only.

Error correction for high priority streams. See option --fec-inner for the list of possible values. The default is "auto".

--isi *value*

Used for DVB-S2 tuners only.

Specify the Input Stream Id (ISI) number to select, from 0 to 255. Used with multistream, see also options --pls-code and --pls-mode.

The default is to keep the entire stream, without multistream selection.

Warning: this option is supported on Linux only. Currently, Windows provides no support for multistream.

--low-priority-fec *value*

Used for DVB-T tuners only.

Error correction for low priority streams. See option --fec-inner for the list of possible values. The default is "auto".

-m *value*
--modulation *value*

Used for DVB-C, DVB-T, DVB-S2 and ATSC tuners.

Modulation type (aka *constellation* for DVB-T). Must be one of "QPSK", "8-PSK", "16-APSK", "32-APSK", "QAM" (auto-detected QAM), "16-QAM", "32-QAM", "64-QAM", "128-QAM", "256-QAM", "8-VSB", "16-VSB".

The default is "64-QAM" for DVB-T and DVB-C, "QPSK" for DVB-S2, "8-VSB" for ATSC.

--offset-count *value*

Used for DVB-T or ATSC tuners only.

Specify the number of offsets from the UHF or VHF channel. The default is zero.

See options --uhf-channel and --vhf-channel.

--pilots *value*

Used for DVB-S2 tuners only.

Presence of pilots frames. Must be one of "auto", "on" or "off". The default is "off".

--plp *value*

Used for DVB-T2 tuners only.

Specify the Physical Layer Pipe (PLP) number to select, from 0 to 255. The default is to keep the entire stream, without PLP selection.

Warning: this option is supported on Linux only. Currently, Windows provides no support for multistream.

--pls-code *value*

Used for DVB-S2 tuners only.

Specifiy the Physical Layer Scrambling (PLS) code value, from 0 to 262143 (0x3FFFF). Used with multistream, see also option --isi.

Warning: this option is supported on Linux only. Currently, Windows provides no support for multistream.

--pls-mode *mode*

Used for DVB-S2 tuners only.

Specify the Physical Layer Scrambling (PLS) mode. Used with multistream, see also option --isi. Must be one of "COMBO", "GOLD", "ROOT". The default is ROOT.

Warning: this option is supported on Linux only. Currently, Windows provides no support for multistream.

--polarity *value*

Used for DVB-S and DVB-S2 tuners only.

Must be one of "horizontal" or "vertical" for linear polarization, "left" or "right" for circular polarization. The default is "vertical".

--roll-off *value*

Used for DVB-S2 tuners only.

Roll-off factor. Must be one of "auto", "0.35", "0.25", "0.20". The default is "0.35" (implied for DVB-S, default for DVB-S2).

--satellite-number *value*

Used for DVB-S and DVB-S2 tuners only.

Satellite/dish number. Must be 0 to 3 with DiSEqC switches and 0 to 1 for non-DiSEqC switches. The default is zero.

--spectral-inversion *value*

Spectral inversion. Must be one of "on", "off" or "auto". The default is "auto".

-s *value*
--symbol-rate *value*

Used for DVB-S, DVB-S2 and DVB-C tuners only.

Symbol rate in symbols/second. The default is 27.5 mega-sym/s for satellite and 6.9 mega-sym/s for cable.

--transmission-mode *value*

Used for DVB-T/T2 tuners only.

Must be one of "auto", "2K", "4K", "8K".

For DVB-T2, also accept "1K", "2K-interleaved", "4K-interleaved", "16K", "32K".

The default is "8K".

--uhf-channel *value*

Used for DVB-T or ATSC tuners only.

Specify the UHF channel number of the carrier. Can be used in replacement to --frequency. Can be combined with an --offset-count option.

The UHF frequency layout depends on the region, see --hf-band-region option.

--vhf-channel *value*

Used for DVB-T or ATSC tuners only.

Specify the VHF channel number of the carrier. Can be used in replacement to --frequency. Can be combined with an --offset-count option.

The VHF frequency layout depends on the region, see --hf-band-region option.

Tuning method 2: Locating the transponder by channel name

-c *name*
--channel-transponder *name*

Tune to the transponder containing the specified channel. The channel name is not case-sensitive and blanks are ignored.

For ATSC networks, the channel name can be replaced by the channel id using the format "*major-id.minor-id*" (e.g. "1.2" or "12.8").

The channel is searched in a *channels configuration file* and the corresponding tuning information in this file is used.

--tuning-file *path*

Specify the channels configuration file to use for option --channel-transponder.

Channel configuration files can be created manually or using the utility *tsscan* or the plugin *nitscan*. The location of the default configuration file depends on the system.

See Appendix B, page 293, for more details on channels configuration files.

Generic common input plugin options

The following options are implicitly defined in all input plugins.

`--help`

Display plugin help text.

eit

Analyze EIT Sections

This plugin analyzes EIT sections and produces a report of *EIT present/following* and *EIT schedule* by transport stream and by service. The EPG depth in days is also reported by service (number of days in advance an event is signaled by an EIT schedule). See 5.2.16 for an example of report.

Usage

```
tsp -P eit [options]
```

Options

-o *filename*
--output-file *filename*

Specify the output file for the report (default: standard output).

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

encap

Encapsulate packets from several PID's into one single PID

This plugin encapsulates all packets from several PID's into one single PID. This unique output PID replaces all input PID's in the transport stream. The output PID is called the "tunnel" or "outer" PID through which all original or "inner" PID's are conveyed.

The reverse operation, the decapsulation, is performed by the *decap* plugin. It replaces the tunnel PID by all original PID's.

The encapsulation format is proprietary and defined below. Since this is not a standard format, it is not interoperable with external systems. The *encap* and *decap* plugins are typically used to hide the structure of some part of the transport stream into a private PID to cross some equipment which does not support the structure of the original stream or could damage its original structure.

Because of the encapsulation overhead, the total volume of encapsulated packets is slightly greater (by approximately 2%) than the original PID's. The encapsulation operation consequently needs some null packets in the original transport stream in addition to the original packets. The output tunnel PID replaces all original packets from the encapsulated PID's plus some null packets. It the original input stream has no stuffing at all, then the *tsp* option "`--add-input-stuffing 1/50`" is sufficient to reserve the additional overhead.

Usage

```
tsp -P encap [options]
```

Options

`-i`
`--ignore-errors`

Ignore errors such as PID conflict or packet overflow.

By default, a PID conflict is reported when the output PID is already present on input but not encapsulated. A packet overflow is reported when the input stream does not contain enough null packets to absorb the encapsulation overhead.

`-m` *value*
`--max-buffered-packets` *value*

Specify the maximum number of buffered packets. The buffered packets are produced by the encapsulation overhead. An overflow is usually caused by insufficient null packets in the input stream. The default is 1,024 packets.

`-o` *value*
`--output-pid` *value*

Specify the output PID containing all encapsulated PID's. This is a mandatory parameter, there is no default. The null PID `0x1FFF` cannot be the output PID.

`--pack`[`=`*value*]

Emit outer packets when they are full only.

By default, emit outer packets as soon as possible, when null packets are available on input. With the default behavior, inner packets are decapsulated with a better time accuracy, at the expense of a higher bitrate of the outer PID when there are many null packets in input.

With the option `--pack`, the emission of an outer packet is delayed until it is full. The bitrate of the outer PID is usually smaller but inner packets may be decapsulated later.

When packing is on, it is possible to limit the distance between packed packets by specifying a positive value. When an outer packet is not yet full but no other input packet is found after the specified number of packets in the TS, then the outer packet is forced to be emitted. With a zero value the distance is disabled (ie. the distance between input packets is unlimited). The value 1 is equivalent to not using the pack mode since outer packets are emitted after one TS packet.

--pcr-pid *value*

Specify a reference PID containing PCR's. The output PID will contain PCR's, based on the same clock. By default, the output PID does not contain any PCR.

--pes-mode *mode*

Enable PES mode encapsulation.

Must be one of "`disabled`", "`fixed`", "`variable`".

--pes-offset *value*

Offset used in Synchronous PES mode encapsulation. The value (positive or negative) is added to the current PCR to generate the PTS timestamp inserted in the PES header.

The recommended values are between -90000 and +90000 (1 second). The value 0 is equivalent to use the Asynchronous PES encapsulation.

It requires to use the PCR option `--pcr-pid`.

-p *pid1*[*-pid2*]
--pid *pid1*[*-pid2*]

Specify an input PID or range of PID's to encapsulate.

Several `--pid` options can be specified. The null PID `0x1FFF` cannot be encapsulated.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[*-label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

Encapsulation format

This section describes the private encapsulation format. It is informative only.

Due to the encapsulation overhead, the number of output packets is slightly larger than the input packets. The input stream must contain a few null packets to absorb the extra output packets. For this reason, null packets (PID `0x1FFF`) are never encapsulated.

There are two encapsulation formats, the *plain* mode and the *PES* mode. The plain mode is more compact but its structure is completely specific. The PES mode uses more overhead but it encapsulates the TS packets into PES packets, which may be easier to process in some cases.

Plain encapsulation format

We define the output elementary stream (ES) as the concatenation of all payloads of all TS packets in the output tunnel PID. In this ES, all input TS packets are contiguous, without encapsulation. The initial `0x47` synchronization byte is removed from all input packets since it is redundant and contains no information. Only the remaining 187 bytes are copied in the output ES.

The Payload Unit Start Indicator (PUSI) bit is set in the header of outer TS packets containing the start of an encapsulated packet. When the PUSI bit is set, the first byte of the payload is a *pointer field* to the beginning of the first encapsulated packet. This packetization method is directly adapted from the standard packetization process for sections, with 187-byte packets instead of sections.

PES encapsulation format

The same plain elementary stream is used, but with a PES envelope. This reduces the payload size, but makes the outer encapsulation more transparent. The overhead is increased by approximately 14 to 20%.

The PES envelope uses a KLVA SMPTE-336M encapsulation[5] to insert the inner payload into one private (testing) key. Each TS packet contains only one key, with a size no larger than the payload of one TS packet. So each PES packet fits into a single TS packet.

The SMPTE-336M encapsulation can be either asynchronous (without timestamps) or synchronous (with PTS). The latter consumes more space (+10 bytes) and is only useful when it is needed to remux the encapsulated stream with an external tool that requires to use PTS marks. No other advantages are provided.

Two variant strategies are implemented. The *fixed* mode uses the short (7-bit) BER encoding. This limits the PES payload to a maximum of 127 bytes. And the adaptation field of the outer packet is enlarged with some stuff. However, the advantage is that the PES is sufficient small to include more data in the outer TS packet. This reduces the possibility than some external processing will split the outer packet in two to accommodate the entire PES data.

The *variable* mode does not impose this restriction and outer packets are filled to the maximum. The drawback is that sometimes the long form of BER encoding is used with two bytes and others the short form with one byte. Furthermore, this increases the chances that some external processing occupies two outer packets for the same inner PES packet. Still, support for those split PES packets is included. The only requirement is that the 26 or 27 PES+KLVA header is inserted in the first packet (with PUSI on). The remaining payload can be distributed in the following TS packets.

The PES envelope has an overhead of 26, 27, 36 or 37 bytes based on:

- 9 bytes for the PES header.
- 0 or 5 bytes for the PTS (synchronous mode).
- 0 or 5 bytes for the Metadata AU Header (synchronous mode)
- 16 bytes for the UL key.
- 1 or 2 bytes for the payload size (BER short or long format).

To enable the use of the Synchronous encapsulation, it is required to use PCR's and provide an offset. This value (positive or negative) will be added to the PCR to compute the PTS. Recommended values are between -90000 and +90000 (-1 and +1 second, respectively). If you use negative values, then you can restore in advance the encapsulated stream after remuxing. However, this will be valid only if you use an external tool to remux. If you're unsure, then don't enable it.

Warning about the Synchronous mode: At start, the PTS marks can't be synchronized with the target PCR PID. This is because the PCR value is not read at start. But the PTS is required to be in all PES packets of the encapsulation. So, it is recommended to discard the outcoming stream until valid PTS values appear in the encapsulated stream.

In order to correctly identify the encapsulated PES stream, it is recommended to declare its PID as a component of an existing or new service. This PID component shall be described as follow in the PMT of the service:

- Stream type :
  - Asynchronous mode : Private Type (`0x06`)
  - Synchronous mode : Metadata Type (`0x15`)
- Descriptors :
  - Add a registration descriptor for "KLVA" (`0x4B4C5641`)

Example

We encapsulate several PID's in outer PID 7777. We attach this outer PID to service id 100. We add a *registration_descriptor* in the description of the outer PID in the PMT.

Asynchronous PES mode encapsulation :

```
tsp ... \
    -P encap -o 7777 --pes-mode ... \
    -P pmt -s 100 -a 7777/0x06 --add-pid-registration 7777/0x4B4C5641 \
    ...
```

[5] See https://impleotv.com/2017/02/17/klv-encoded-metadata-in-stanag-4609-streams/

Synchronous PES mode encapsulation (with PCR) :

```
tsp ... \
    -P encap -o 7777 --pes-mode --pes-offset -50000 --pcr-pid 101 ... \
    -P pmt -s 100 -a 7777/0x15 --add-pid-registration 7777/0x4B4C5641 \
    ...
```

file (input)

Transport Stream Files Input

This input module reads transport stream packets from one or more files. The specified files do not need to be regular files, they can be named pipes or anything that can be named and read from.

The default file is the standard input, which can also be a pipe. Since the plugin *file* is the default input plugin (if no option `-I` is specified), this means that the default *tsp* input is the standard input.

The input files must contain a flow of contiguous 188-bytes TS packets. If this is not the case, consider using the *tsresync* utility.

Usage

```
tsp -I file [options] [file-name ...]
```

Parameter

Name of the input files. The files are read in sequence, unless `--interleave` is specified.

If no file is specified, the standard input is read by default. When several files are specified, use '-' as file name to specify the standard input.

Options

-b *value*
--byte-offset *value*

Start reading each file at the specified byte offset (default: `0`). This option is allowed only if the input file is a regular file.

-f
--first-terminate

With `--interleave`, terminate the processing when any file reaches the end of file.

By default, continue reading until the last file reaches the end of file (other files are replaced with null packets after their end of file).

-i
--infinite

Repeat the playout of the file infinitely (default: only once). This option is allowed only if the input file is a regular file and there is only one input file.

--interleave[=*value*]

Interleave files instead of reading them one by one. All files are simultaneously opened.

The optional value is a chunk size *N*, a packet count (default is 1). *N* packets are read from the first file, then *N* from the second file, etc. and then loop back to *N* packets again from the first file, etc.

-l *value*
--label-base *value*

Set a label on each input packet. Packets from the first file are tagged with the specified base label, packets from the second file with base label plus one, and so on.

For a given file, if the computed label is above the maximum (31), its packets are not labelled.

-p *value*
--packet-offset *value*

Start reading each file at the specified TS packet (default: `0`).

This option is allowed only if all input files are regular file.

-r *count*
--repeat *count*

Repeat the playout of each file the specified number of times (default: only once).

This option is allowed only if all input files are regular files.

If several input files are specified, the first file is repeated the specified number of times, then the second file is repeated the same number of times, and so on.

Generic common input plugin options

The following options are implicitly defined in all input plugins.

`--help`

Display plugin help text.

file (output)

Transport Stream Files Output

This output plugin writes the TS packets to a file. The output file receives a flow of contiguous 188-bytes TS packets.

The default file is the standard output, which can be a pipe. Since the plugin *file* is the default output plugin (if no option –O is specified), this means that the default tsp output is the standard output.

Usage

```
tsp -O file [options] [file-name]
```

Parameter

Name of the created output file. Use standard output by default.

Options

-a
--append

If the file already exists, append to the end of the file. By default, existing files are overwritten.

-k
--keep

Keep existing file (abort if the specified file already exists). By default, existing files are overwritten.

Generic common output plugin options

The following options are implicitly defined in all output plugins.

--help

Display plugin help text.

file (packet processing)

Save Packets to a File and Pass

This plugin writes the TS packets to a file and pass them to the next plugin in the chain. The output file receives a flow of contiguous 188-bytes TS packets.

Usage

```
tsp -P file [options] file-name
```

Parameter

Name of the created output file.

Options

-a
--append

If the file already exists, append to the end of the file. By default, existing files are overwritten.

-k
--keep

Keep existing file (abort if the specified file already exists). By default, existing files are overwritten.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

filter

General-Purpose Packet Filter

This plugin filters TS packets according to various conditions. When a packet meets at least one of the specified condition, it is passed to the next packet in the chain. Otherwise, it is dropped.

Note: To filter packets which meets several simultaneous conditions ("and" instead of "or"), simply chain several filter plugins on the command line.

Usage

```
tsp -P filter [options]
```

Options

--adaptation-field

Select packets with an adaptation field.

--after-packets *count*

Let the first *count* packets pass transparently without filtering. Start to apply the filtering criteria after that number of packets.

-c
--clear

Select clear (unscrambled) packets. Equivalent to "`--scrambling-control 0`".

--every *count*

Select one packet every that number of packets.

--input-stuffing

Select packets which were articially inserted as stuffing before the input plugin using *tsp* options `--add-start-stuffing`, `--add-input-stuffing` and `--add-stop-stuffing`.

Be aware that these packets may no longer be null packets if some previous plugin injected data, replacing stuffing.

-i *index1*[-[*index2*]]
--interval *index1*[-[*index2*]]

Select all packets in the specified interval from the start of the stream. The packets in the stream are indexed starting at zero.

- In the form *index1*, only one packet is selected, at the specified index.
- In the form *index1-index2*, all packets in the specified range of indexes, inclusive, are selected.
- In the form *index1-*, all packets starting at the specified index are selected, up to the end of the stream.

Several options `--interval` can be specified.

-l *label1*[-*label2*]
--label *label1*[-*label2*]

Select packets with any of the specified labels. Labels should have typically be set by a previous plugin in the chain.

Several `--label` options may be specified.

Note that the option `--label` is different from the generic option `--only-label`. The generic option `--only-label` acts at *tsp* level and controls which packets are passed to the plugin. All other packets are directly passed to the next plugin without going through this plugin. The option `--label`, on the other hand, is specific to the *filter* plugin and selects packets with specific labels among the packets which are passed to this plugin.

--max-adaptation-field-size *value*

Select packets with no adaptation field or with an adaptation field the size (in bytes) of which is not greater than the specified value.

--max-payload-size *value*

Select packets with no payload or with a payload the size (in bytes) of which is not greater than the specified value.

--min-adaptation-field-size *value*

Select packets with an adaptation field the size (in bytes) of which is equal to or greater than the specified value.

--min-payload-size *value*

Select packets with a payload the size (in bytes) of which is equal to or greater than the specified value.

-n
--negate

Negate the filter: specified packets are excluded.

--nullified

Select packets which were explicitly turned into null packets by some previous plugin in the chain (typically using a `--stuffing` option).

Be aware that these packets may no longer be null packets if some intermediate plugin injected data, replacing stuffing.

--pattern *value*

Select packets containing the specified pattern bytes. The value must be a string of hexadecimal digits specifying any number of bytes.

By default, the packet is selected when the value is anywhere inside the packet.

With option `--search-payload`, only search the pattern in the payload of the packet.

With option `--search-offset`, the packet is selected only if the pattern is at the specified offset in the packet.

When `--search-payload` and `--search-offset` are both specified, the packet is selected only if the pattern is at the specified offset in the payload.

--payload

Select packets with a payload.

--pcr

Select packets with PCR or OPCR.

--pes

Select packets with clear PES headers.

-p *pid1*[*-pid2*]
--pid *pid1*[*-pid2*]

PID filter: select packets with these PID values.

Several `-p` or `--pid` options may be specified.

--reset-label *label1*[*-label2*]

Clear the specified labels on the selected packets.

Do not drop unselected packets, simply clear one or more labels on selected ones.

Several `--reset-label` options may be specified.

--reset-permanent-label *label1*[*-label2*]

Clear the specified labels on all packets, selected and unselected ones, after at least one was selected.

Do not drop unselected packets, simply use selected ones as trigger.

Several `--reset-permanent-label` options may be specified.

`--scrambling-control` *`value`*

Select packets with the specified scrambling control value.

Valid values are 0 (clear), 1 (reserved), 2 (even key), 3 (odd key).

`--search-offset` *`value`*

With `--pattern`, only search the set of bytes at the specified offset in the packet (the default) or in the payload (with `--search-payload`).

`--search-payload`

With `--pattern`, only search the set of bytes in the payload of the packet. Do not search the pattern in the header or adaptation field.

`--set-label` *`label1`*`[-`*`label2`*`]`

Set the specified labels on the selected packets.

Do not drop unselected packets, simply mark selected ones with one or more labels.

Several `--set-label` options may be specified.

`--set-permanent-label` *`label1`*`[-`*`label2`*`]`

Set the specified labels on all packets, selected and unselected ones, after at least one was selected.

Do not drop unselected packets, simply use selected ones as trigger.

Several `--set-permanent-label` options may be specified.

`-s`
`--stuffing`

Replace excluded packets with stuffing (null packets) instead of removing them. Useful to preserve bitrate.

`--unit-start`

Select packets with payload unit start indicator.

`-v`
`--valid`

Select valid packets. A valid packet starts with `0x47` and has its *transport_error_indicator* cleared.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

`--help`

Display this help text.

`--only-label` *`label1`*`[-`*`label2`*`]`

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

fork (input)

Receive packets from a forked process

This input plugin forks a process and receives all TS packets from the standard output of this process.

Using this input plugin with *tsp* is equivalent to reading the input pipe. The following two commands have the same effect (the command "*receive*" being a fictitious one):

```
receive stream | tsp ...
tsp –I fork 'receive stream' ...
```

So, this plugin is redundant with the shell pipe features. However, this plugin is useful when the *tsp* process is created from another native application (not a shell script). In that case, it is much easier for this application to create a simple binary process rather than a shell and its commands.

Additionally, this input plugin becomes necessary with *tsswitch* which accepts several inputs. The following command has no equivalent with shell pipes:

```
tsswitch –I fork 'receive stream1' –I fork 'receive stream2' -O ...
```

Usage

```
tsp -I fork [options] 'command'
```

Parameter

The `'command'` parameter specifies the shell command to execute in the forked process. The standard output of this process is a pipe from which the TS packets are received by the input plugin. If the command contains spaces or shell special sequences, the complete command string must be surrounded by quotes.

If the command is too long or too complicated, it is recommended to use a script. If the created command is another TSDuck command, it is possible to shorten the command using partial command line redirection (see 3.1.5).

Options

-b *value*
--buffered-packets *value*

Windows only: Specifies the pipe buffer size in number of TS packets.

-n
--nowait

Do not wait for child process termination at end of input.

Generic common input plugin options

The following options are implicitly defined in all input plugins.

--help

Display plugin help text.

fork (output)

Send packets to a forked process

This output plugin forks a process and sends all TS packets to the standard input of this process.

Using this output plugin with *tsp* is equivalent to writing to the output pipe. The following two commands have the same effect (the command "*send*" being a fictitious one):

```
tsp ... | send stream
tsp ... -O fork 'send stream'
```

So, this plugin is redundant with the shell pipe features. However, this plugin is useful when the *tsp* process is created from another native application (not a shell script). In that case, it is much easier for this application to create a simple binary process rather than a shell and its commands.

Usage

```
tsp -O fork [options] 'command'
```

Parameter

The `'command'` parameter specifies the shell command to execute in the forked process. The standard input of this process is a pipe receiving the TS packets. If the command contains spaces or shell special sequences, the complete command string must be surrounded by quotes.

If the command is too long or too complicated, it is recommended to use a script. If the created command is another TSDuck command, it is possible to shorten the command using partial command line redirection (see 3.1.5).

Options

-b *value*
--buffered-packets *value*

Windows only: Specifies the pipe buffer size in number of TS packets.

-n
--nowait

Do not wait for child process termination at end of input.

Generic common output plugin options

The following options are implicitly defined in all output plugins.

--help

Display plugin help text.

fork (packet processing)

Redirect packets to a forked process

This plugin forks a process and sends all TS packets to the standard input of this process. The TS packets are also normally passed to the next processor in the chain.

This plugin can be used to duplicate the output stream at any point in the packet processing chain.

Usage

```
tsp -P fork [options] 'command'
```

Parameter

The `'command'` parameter specifies the shell command to execute in the forked process. The standard input of this process is a pipe receiving the TS packets. If the command contains spaces or shell special sequences, the complete command string must be surrounded by quotes.

If the command is too long or too complicated, it is recommended to use a script. If the created command is another TSDuck command, it is possible to shorten the command using partial command line redirection (see 3.1.5).

Options

-b *value*
--buffered-packets *value*

Specifies the number of TS packets to buffer before sending them through the pipe to the forked process. When set to zero, the packets are not buffered and sent one by one.

The default is 500 packets in real-time mode and 1000 packets in offline mode.

-i
--ignore-abort

Ignore early termination of child process. By default, if the child process aborts and no longer reads the packets, *tsp* also aborts.

-n
--nowait

Do not wait for child process termination at end of input.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

hides (output)

Send the Transport Stream to a HiDes Modulator Device

This plugin sends the output transport stream to a HiDes modulator device.

Usage

```
tsp -O hides [options]
```

Options

-a *value*
--adapter *value*

Specify the HiDes adapter number to use. By default, the first HiDes device is selected.

Use the command *tshides* to list all HiDes devices.

Use `--adapter` or `--device` but not both.

-b *value*
--bandwidth *value*

Bandwidth in MHz. Must be one of 5, 6, 7, 8.

The default is 8 MHz.

-c *value*
--constellation *value*

Constellation type. Must be one of `"QPSK"`, `"16-QAM"`, `"64-QAM"`.

The default is 64-QAM.

--dc-compensation *i-value/q-value*

Specify the DC offset compensation values for I and Q. Each offset value shall be in the range -512 to 512.

-d *"name"*
--device *"name"*

Specify the HiDes device name to use. By default, the first HiDes device is selected.

Use the command *tshides* to list all HiDes devices.

Use `--adapter` or `--device` but not both.

-f *value*
--frequency *value*

Frequency, in Hz, of the output carrier. There is no default, this is a mandatory parameter.

--gain *value*

Adjust the output gain to the specified value in dB.

-g *value*
--guard-interval *value*

Guard interval. Must be one of `"1/32"`, `"1/16"`, `"1/8"`, `"1/4"`.

The default is 1/32.

--h *value*
--high-priority-fec *value*

Error correction for high priority streams. Must be one of `"1/2"`, `"2/3"`, `"3/4"`, `"5/6"`, `"7/8"`.

The default is 2/3.

--s *value*
--spectral-inversion *value*

Spectral inversion. Must be one of `"off"`, `"on"`, `"auto"`.

The default is `"auto"`.

Note that this option is ignored on Windows.

-t *value*
--transmission-mode *value*

Transmission mode. Must be one of "4K", "2K", "8K". The default is "8K".

Generic output plugin options

The following options are implicitly defined in all ouput plugins.

--help

Display this help text.

history

Report a History of Major Events on the Transport Stream

This plugin reports a history of the major events on the transport stream: new PID's, new tables, clear ⇔ scrambled transitions, suspended and restarted PID's, etc.

By default, the messages are reported, like all other `tsp` messages, on the standard error file. Each output line is formatted as follow:

```
* history: packet-number: MESSAGE
```

With option `--milli-seconds`, the *packet-number* is replaced by a number of milliseconds (based on the TS bitrate).

Some events are detected only some time after they occurred (determining if a PID is suspended, for instance, is detected long after the last packet on this PID). As a consequence, some messages may be unsorted. To sort messages according to packet numbers, use a command like:

```
tsp -P history ...  2>&1 | grep '* history:' | sort -t : -k 2 -n
```

When an output file is specified using `--output-file`, the log prefix "* history:" is not present. In this case, the sort command becomes:

```
sort -n output-file-name
```

Usage

```
tsp -P history [options]
```

Options

-c
--cas

Report all CAS events (new ECM, crypto-period change). By default, only clear to/from scrambled transitions are reported.

-e
--eit

Report all EIT. By default, EIT are not reported.

-i
--ignore-stream-id-change

Do not report stream_id modifications in a stream. Some subtitle streams may constantly swap between "private stream" and "padding stream". This option suppresses these annoying messages.

-m
--milli-seconds

For each message, report time in milli-seconds from the beginning of the stream instead of the TS packet number. This time is a playback time based on the current TS bitrate (use plugin *pcrbitrate* just before plugin *history* when necessary).

-o *filename*
--output-file *filename*

Specify the output file for reporting history lines. By default, report history lines on standard error using the tsp logging mechanism.

-s *value*
--suspend-packet-threshold *value*

Number of packets in the TS after which a PID is considered as suspended. By default, if no packet is found in a PID during 60 seconds (according to the TS bitrate), the PID is considered as suspended.

-t
--time-all

Report all TDT and TOT. By default, only report TDT preceeding another event.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

hls (input)

Receive HTTP Live Streaming (HLS) media

This plugin reads a combined transport stream from an HLS streaming server. All media segments are concatenated into one single transport stream.

In the case of live content, the HLS playlist is reloaded as often as necessary to get a continuous content.

Restriction: The HLS specification allows two kinds of media content: TS and fMP4 (fragmented MP4). Since TSDuck is a *transport stream* toolkit, this plugin can only receive TS media.

Usage

```
tsp -I hls [options] url
```

Parameter

Specify the URL of an HLS manifest or playlist. This is typically an URL ending in `.m3u8`.

This can be a **master playlist**, referencing several versions of the same content (with various bitrates or resolutions). This can also be a **media playlist**, referencing all segments of one single content.

When the playlist is a master one, the first media playlist is selected by default. Specify options can be used to select a playlist based on criteria (bandwidth or resolution for instance) When a media playlist failed to load, the next one is used (with respect to the selection criteria), etc.

Options

`--connection-timeout` *value*

Specify the connection timeout in milliseconds.

By default, let the operating system decide.

`--highest-bitrate`

When the URL is a master playlist, use the content with the highest bitrate.

`--highest-resolution`

When the URL is a master playlist, use the content with the highest screen resolution.

`-l`
`--list-variants`

When the URL is a master playlist, list all possible streams bitrates and resolutions.

`--live`

Specify that the input is a live stream and the playout shall start at the last segment in the playlist.

This is an alias for `--start-segment -1`

`--lowest-bitrate`

When the URL is a master playlist, use the content with the lowest bitrate.

`--lowest-resolution`

When the URL is a master playlist, use the content with the lowest screen resolution.

`--max-bitrate` *value*

When the URL is a master playlist, select a content the bitrate of which is lower than the specified maximum.

`--max-height` *value*

When the URL is a master playlist, select a content the resolution of which has a lower height than the specified maximum.

`--max-queue` *value*

Specify the maximum number of queued TS packets before their insertion into the stream.

The default is 1000.

--max-width *value*

When the URL is a master playlist, select a content the resolution of which has a lower width than the specified maximum.

--min-bitrate *value*

When the URL is a master playlist, select a content the bitrate of which is higher than the specified minimum.

--min-height *value*

When the URL is a master playlist, select a content the resolution of which has a higher height than the specified minimum.

--min-width *value*

When the URL is a master playlist, select a content the resolution of which has a higher width than the specified minimum.

--proxy-host *name*

Optional proxy host name for Internet access.

--proxy-password *string*

Optional proxy password for Internet access (for use with `--proxy-user`).

--proxy-port *value*

Optional proxy port for Internet access (for use with `--proxy-host`).

--proxy-user *name*

Optional proxy user name for Internet access.

--receive-timeout *value*

Specify the data reception timeout in milliseconds. This timeout applies to each receive operation, individually.

By default, let the operating system decide.

--save-files *directory-name*

Specify a directory where all downloaded files, media segments and playlists, are saved before being passed to the next plugin.

This is typically a debug option to analyze the input HLS structure.

-s *value*
--segment-count *value*

Stop receiving the HLS stream after receiving the specified number of media segments.

By default, receive the complete content.

--start-segment *value*

Start at the specified segment in the initial playlist.

The value can be positive or negative. Positive values are indexes from the start of the playlist: 0 is the first segment (the default), +1 is the second segment, etc. Negative values are indexes from the end of the playlist: -1 is the last segment, -2 is the preceding segment, etc.

By default, start with the first media segment.

Generic common input plugin options

The following options are implicitly defined in all input plugins.

--help

Display plugin help text.

hls (output)

Generate HTTP Live Streaming (HLS) media

This output plugin generates HLS playlists and media segments on local files only. It can also purge obsolete media segments and regenerate live playlists. The plugin always generate media segments. The playlist generation is optional.

To setup a complete HLS server, it is necessary to setup an external HTTP server such as Apache which simply serves the files, playlist and media segments.

Usage

```
tsp -O hls [options] filename
```

Parameter

Specify the name template of the output media segment files. A number is automatically added to the name part so that successive segment files receive distinct names.

Example: if the specified file name is *foo-.ts*, the various segment files are named *foo-000000.ts*, *foo-000001.ts*, etc.

If the specified template already contains trailing digits, this unmodified name is used for the first segment. Then, the integer part is incremented.

Example: if the specified file name is *foo-027.ts*, the various segment files are named *foo-027.ts*, *foo-028.ts*, etc.

Options

-d *value*
--duration *value*

Specify the target duration in seconds of media segments.

The default is 10 seconds per segment for VoD streams and 5 seconds for live streams.

-f *value*
--fixed-segment-size *value*

Specify the size in bytes of all media segments. By default, the segment size is variable and based on the `--duration` parameter. When `--fixed-segment-size` is specified, the `--duration` parameter is only used as a hint in the playlist file.

-l *value*
--live *value*

Specify that the output is a live stream. The specified value indicates the number of simultaneously available media segments. Obsolete media segment files are automatically deleted.

By default, the output stream is considered as VoD and all created media segments are preserved.

-p *filename*
--playlist *filename*

Specify the name of the playlist file. The playlist file is rewritten each time a new segment file is completed or an obsolete one is deleted.

The playlist and the segment files can be written to distinct directories but, in all cases, the URI of the segment files in the playlist are always relative to the playlist location.

By default, no playlist file is created (media segments only).

-s *value*
--start-media-sequence *value*

Initial media sequence number in `#EXT-X-MEDIA-SEQUENCE` directive in the playlist.

The default is zero.

Generic common output plugin options

The following options are implicitly defined in all output plugins.

--help

Display plugin help text.

http (input)

Read a Transport Stream from an HTTP Server

This plugin reads a transport stream from a URL. The HTTP server is expected to send a valid transport stream without encapsulation.

It is possible to repeat the operation a number of times. In that case, the URL is re-opened each time and the content may be different if the served stream is not a static file.

The expected MIME type for an MPEG transport stream is `video/mp2t`. If a different type is reported by the server, a warning message is displayed but the content is accepted as long as it is a valid transport stream.

Usage

```
tsp -I http [options] url
```

Parameter

Specify the URL from which to read the transport stream.

Options

`--connection-timeout` *value*

Specify the connection timeout in milliseconds.

By default, let the operating system decide.

`--ignore-errors`

With `--repeat` or `--infinite`, repeat also in case of error.

By default, repetition stops on error.

`-i`
`--infinite`

Repeat the playout of the content infinitely (default: only once).

`--max-queue` *value*

Specify the maximum number of queued TS packets before their insertion into the stream.

The default is 1000.

`--proxy-host` *name*

Optional proxy host name for Internet access.

`--proxy-password` *string*

Optional proxy password for Internet access (for use with `--proxy-user`).

`--proxy-port` *value*

Optional proxy port for Internet access (for use with `--proxy-host`).

`--proxy-user` *name*

Optional proxy user name for Internet access.

`--receive-timeout` *value*

Specify the data reception timeout in milliseconds. This timeout applies to each receive operation, individually.

By default, let the operating system decide.

`--reconnect-delay` *value*

With `--repeat` or `--infinite`, wait the specified number of milliseconds.

By default, repeat immediately.

-r *count*
--repeat *count*

Repeat the playout of the content the specified number of times (default: only once).

Generic common input plugin options

The following options are implicitly defined in all input plugins.

--help

Display plugin help text.

inject

Inject Tables in a Transport Stream

This plugin injects MPEG tables and sections into a transport stream, replacing a PID or stealing packets from stuffing.

When the PID is replaced, all previous content of this PID is lost and all its packets are replaced at the same position in the stream. The bitrate of the PID is unchanged.

When a new PID is created, replacing some stuffing packets, its bitrate must be known. There are several explicit or implicit ways to specify the bitrate of the new PID. First, the option `--bitrate` can be used. Second, the option `--inter-packet` can be used to specify the placement of the packets in the stream. The last option is to specify an explicit repetition rate for each input section file.

Usage

```
tsp -P inject [options] input-file[=rate] ...
```

Parameters

input-file[=*rate*]

Binary or XML files containing one or more sections or tables. By default, files with a name ending in `.xml` are XML and files with a name ending in `.bin` are binary. For other file names, explicitly specify `--binary` or `--xml`.

If different repetition rates are required for different files, a parameter can be "*filename*=*value*" where *value* is the repetition rate in milliseconds for all sections in that file.

Options

--binary

Specify that all input files are binary, regardless of their file name.

-b *value*
--bitrate *value*

Specifies the bitrate for the new PID, in bits / second.

-e *value*
--evaluate-interval *value*

When used with `--replace` and when specific repetition rates are specified for some input files, the bitrate of the target PID is re-evaluated on a regular basis. The value of this option specifies the number of packet in the target PID before re-evaluating its bitrate. The default is 100 packets.

-f
--force-crc

Force recomputation of CRC32 in long sections. Ignore CRC32 values in input file.

-i *value*
--inter-packet *value*

Specifies the packet interval for the new PID, that is to say the number of TS packets in the transport between two packets of the new PID. Use instead of `--bitrate` if the global bitrate of the TS cannot be determined.

-j
--joint-termination

Perform a *joint termination* when section insersion is complete. Meaningful only when `--repeat` is specified. See the description of the `tsp` command for more details on *joint termination*.

-p *value*
--pid *value*

PID of the output TS packets. This is a required parameter, there is no default value. To replace the content of an existing PID, use option `--replace`. To steal stuffing packets and create a new

PID, use either option `--bitrate` or `--inter-packet`. Exactly one option `--replace`, `--bitrate` or `--inter-packet` must be specified.

`--poll-files`

Poll the presence and modification date of the input files at regular intervals. When a file is created, modified or deleted, reload all files at the next section boundary and restart the injection cycles. When a file is deleted, its sections are no longer injected. If the file reappears later, its sections will be injected again.

By default, all input files are loaded once at initialization time and an error is generated if a file is missing.

`--repeat` *`count`*

Repeat the insertion of a complete cycle of sections the specified number of times. By default, the sections are infinitely repeated.

`-r`
`--replace`

Replace the content of an existing PID. Do not steal stuffing.

`-s`
`--stuffing`

Insert stuffing at end of each section, up to the next TS packet boundary. By default, sections are packed and start in the middle of a TS packet, after the previous section. Note, however, that section headers are never scattered over a packet boundary.

`-t`
`--terminate`

Terminate packet processing when section insersion is complete. Meaningful only when `--repeat` is specified. By default, when section insertion is complete, the transmission continues and the stuffing is no longer modified (if `--replace` is specified, the PID is then replaced by stuffing).

`--xml`

Specify that all input files are XML, regardless of their file name.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

`--help`

Display this help text.

`--only-label` *`label1`*`[-`*`label2`*`]`

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

ip (input)

UDP/IP unicast or multicast input

This input plugin receives TS packets from UDP/IP, multicast or unicast.

The received UDP datagrams are analyzed and all TS packets are extracted. Optional extra data at the beginning of the datagram (such as RTP headers) are discarded.

Using this plugin forces *tsp* and all plugins to use their real-time defaults (see the reference documentation for *tsp).*

Usage

```
tsp -I ip [options] [[source@]address:]port
```

Parameter

The parameter [*address*:]*port* describes the destination of UDP packets to receive. The *port* part is mandatory and specifies the UDP port to listen on. The *address* part is optional. It specifies an IP multicast address to listen on. It can be also a host name that translates to a multicast address.

An optional source address can be specified as *source@address:port* in the case of source-specific multicast (SSM).

If the address is not specified, the plugin simply listens on the specified local port and receives the packets which are sent to one of the local (unicast) IP addresses of the system.

UDP reception options

-b *value*
--buffer-size *value*

Specify the UDP socket receive buffer size (socket option).

--default-interface

Let the system find the appropriate local interface on which to listen. By default, listen on all local interfaces.

-f
--first-source

Filter UDP packets based on the source address. Use the sender address of the first received packet as only allowed source.

This option is useful when several sources send packets to the same destination address and port. Accepting all packets could result in a corrupted stream and only one sender shall be accepted.

To allow a more precise selection of the sender, use option `--source`. Options `--first-source` and `--source` are mutually exclusive.

-l *address*
--local-address *address*

Specify the IP address of the local interface on which to listen. It can be also a host name that translates to a local address. By default, listen on all local interfaces.

--no-reuse-port

Disable the reuse port socket option. Do not use unless completely necessary.

--receive-timeout *value*

Specify the UDP reception timeout in milliseconds. This timeout applies to each receive operation, individually. By default, receive operations wait for data, possibly forever.

-r
--reuse-port

Set the reuse port socket option. This is now enabled by default, the option is present for legacy only.

-s *address[:port]*
--source *address[:port]*

Filter UDP packets based on the specified source address.

This option is useful when several sources send packets to the same destination address and port. Accepting all packets could result in a corrupted stream and only one sender shall be accepted.

Options `--first-source` and `--source` are mutually exclusive.

--ssm

This option forces the usage of source-specific multicast (SSM) using the source address which is specified by the option `--source`. Without `--ssm`, standard ("any-source') multicast is used and the option `--source` is used to filter incoming packets.

The `--ssm` option is implicit when the classical SSM syntax *source@address:port* is used.

Other options

-d *value*
--display-interval *value*

Specify the interval in seconds between two displays of the evaluated real-time input bitrate. The default is to never display the bitrate. This option is ignored if `--evaluation-interval` is not specified.

-e *value*
--evaluation-interval *value*

Specify that the real-time input bitrate shall be evaluated on a regular basis. The value specifies the number of seconds between two evaluations. By default, the real-time input bitrate is never evaluated and the input bitrate is evaluated from the PCR in the input packets.

Generic common input plugin options

The following options are implicitly defined in all input plugins.

--help

Display plugin help text.

ip (output)

UDP/IP unicast or multicast output

This output plugin sends TS packets using UDP/IP, multicast or unicast.

Each UDP datagram is filled with one or more TS packets (see option `--packet-burst`). By default, the datagrams contain TS packets without any extra information or encapsulation. Use the option `--rtp` to generate RTP datagrams.

Using this plugin forces *tsp* and all plugins to use their real-time defaults (see the reference documentation for *tsp).*

Usage

```
tsp -O ip [options] address:port
```

Parameter

The parameter *address*:*port* describes the destination for UDP packets. The *address* specifies an IP address which can be either unicast or multicast. It can be also a host name that translates to an IP address. The *port* specifies the destination UDP port.

Options

-e
--enforce-burst

Enforce that the number of TS packets per UDP packet is exactly what is specified in option `--packet-burst`. By default, this is only a maximum value.

For instance, without `--enforce-burst` and the default `--packet-burst` value (7 packets), if the output plugin receives 16 TS packets, it immediately sends 3 UDP packets containing 7, 7 and 2 TS packets respectively.

With option `--enforce-burst`, only the first 14 TS packets would be sent, using 2 UDP packets. The remaining 2 TS packets are buffered, delaying their departure until 5 more TS packets are available.

-p *value*
--packet-burst *value*

Specifies the maximum number of TS packets to be grouped into each UDP datagram.

The default is 7, the maximum is 128.

IP options

-l *address*
--local-address *address*

When the destination is a multicast address, specify the IP address of the outgoing local interface. It can be also a host name that translates to a local address.

-s *value*
--tos *value*

Specifies the TOS (Type-Of-Service) socket option. Depending on the specified value or on the operating system, this option may require privileges or may even have no effect at all.

-t *value*
--ttl *value*

Specifies the TTL (Time-To-Live) socket option. The actual option is either "Unicast TTL" or "Multicast TTL", depending on the destination address.

Warning: Remember than the default Multicast TTL is 1 on most systems.

RTP options

-r
--rtp

Use the Real-time Transport Protocol (RTP) in output UDP datagrams.

By default, TS packets are sent in UDP datagrams without encapsulation.

--payload-type *value*

With `--rtp`, specify the payload type.

By default, use 33, the standard RTP type for MPEG2-TS. Do not modify unless there is a good reason to do so.

--pcr-pid *value*

With `--rtp`, specify the PID containing the PCR's which are used as reference for RTP timestamps.

By default, use the first PID containing PCR's.

--ssrc-identifier *value*

With `--rtp`, specify the SSRC identifier.

By default, use a random value. Do not modify unless there is a good reason to do so.

--start-sequence-number *value*

With `--rtp`, specify the initial sequence number.

By default, use a random value. Do not modify unless there is a good reason to do so.

Generic common output plugin options

The following options are implicitly defined in all output plugins.

--help

Display plugin help text.

limit

Limit the global bitrate by dropping packets

This plugin limits the global bitrate of the transport stream. Packets are dropped when necessary to maintain the overall bitrate below a given maximum. The bitrate is computed from PCR's (the default) or from the processing wall clock time.

Packets are not dropped randomly. Some packets are more likely to be dropped than others. When the bitrate exceeds the maximum, the number of packets in excess is permanently recomputed. The type of packets to drop depends on the number of packets in excess. There are several thresholds which are specified by the corresponding options:

- Below `--threshold1`, only null packets are dropped.
- Below `--threshold2`, if `--pid` options are specified, video packets from the specified PID's are dropped (except packets containing a PUSI or a PCR).
- Below `--threshold3`, if `--pid` options are specified, all packets (not only video) from the specified PID's are dropped (except packets containing a PUSI or a PCR).
- Below `--threshold4`, packets from any video or audio PID are dropped (except packets containing a PUSI or a PCR).
- Above the last threshold, any packet can be dropped.

Note: All thresholds, except the last one, can be disabled using a zero value.

Usage

```
tsp –P limit [options]
```

Options

-b *value*
--bitrate *value*

Limit the overall bitrate of the transport stream to the specified value in bits/second. This is a mandatory option, there is no default.

-p *pid1*[*-pid2*]
--pid *pid1*[*-pid2*]

Specify PID’s the content of which can be dropped when the maximum bitrate is exceeded. Several `--pid` options can be specified.

-1 *value*
--threshold1 *value*

Specify the first threshold for the number of packets in excess. The default is 10 packets.

-2 *value*
--threshold2 *value*

Specify the second threshold for the number of packets in excess. The default is 100 packets.

-3 *value*
--threshold3 *value*

Specify the third threshold for the number of packets in excess. The default is 500 packets.

-4 *value*
--threshold4 *value*

Specify the fourth threshold for the number of packets in excess. The default is 1000 packets.

-w
--wall-clock

Compute bitrates based on real wall-clock time. The option is meaningful with live streams only. By default, compute bitrates based on PCR's.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

merge

Merge TS packets coming from the standard output of a command

This plugin creates a process. The process is expected to write TS packets on its standard output. The resulting transport stream is merged with the main transport stream which is processed by *tsp* and the chain of plugins.

As usual with *tsp* plugins, the TS packets which come from the merged transport stream are inserted into the main transport stream by replacing stuffing packets. The obvious requirement is that the stuffing bitrate of the main stream is greater than the overall bitrate of the merged stream. Otherwise, it wouldn't fit in the main transport steam.

Typically (although not required), the created process is another *tsp* command which prepares the merged transport stream. Preparing the transport stream may include reducing the bitrate by removing stuffing and services, remapping PID's or renaming services which could conflict with existing PID's or services in the other transport stream.

By default, the following operations are performed while merging transport streams. These defaults can be changed using appropriate options.

- The PAT and SDT from the merged transport stream are merged into the corresponding tables in the main transport stream. The merged services are consequently correctly referenced in the main transport stream.
- Similarly, the CAT is also merged so that EMM PID's from the merged transport stream are correctly referenced in the main transport stream.
  Warning: The CAT is an optional table and the *merge* plugin will not create one if there is none. If you want to make sure that a merged CAT will be present, use an instance of the *cat* plugin before *merge*.
- The PID's `0x00` to `0x1F` are dropped from the merged transport stream. These PID's contain the base PSI/SI which are normally present in all transport streams. Merging these PID's would create conflicts. Instead, the most important PSI/SI tables are correctly merged as previously described. All other PID's are passed. This can be modified using options `--drop` and `--pass`.
- PID conflicts are detected. If packets from the same PID are found in the two transport streams, the PID is dropped from the merged stream.
- In packets coming from the merged transport stream, the PCR's are restamped according to their new placement in the main transport stream.

Usage

```
tsp -P merge [options] 'command'
```

Parameter

The command parameter specifies the shell command to execute in the forked process. The standard output of this process is a pipe into which the TS packets are written. If the command contains spaces or shell special sequences, the complete command string must be surrounded by quotes.

If the command is too long or too complicated, it is recommended to use a script. If the created command is another TSDuck command, it is possible to shorten the command using partial command line redirection (see 3.1.5).

Options

`-d` *pid*`[-`*pid*`]`
`--drop` *pid*`[-`*pid*`]`

Drop the specified PID or range of PID's from the merged stream. Several options `--drop` can be specified.

`--ignore-conflicts`

Ignore PID conflicts. By default, when packets with the same PID are present in the two streams, the PID is dropped from the merged stream.

Warning: this is a dangerous option which can result in an inconsistent transport stream.

-j
--joint-termination

Perform a *joint termination* when the merged stream is terminated. See the description of the `tsp` command for more details on *joint termination*.

--max-queue *value*

Specify the maximum number of queued TS packets before their insertion into the stream. The default is 1000.

--no-pcr-restamp

Do not restamp PCR's from the merged TS into the main TS. By default, PCR's in the merged stream are restamped to match their position in the final stream.

The DTS and PTS are never restamped because they are independent from their position in the stream. When the PCR's in the merged stream have discontinuities (such as when cycling a TS file), restamping the PCR's can break the video playout since they become decorrelated with the DTS and PTS.

--no-psi-merge

Do not merge PSI/SI from the merged TS into the main TS. By default, the PAT, CAT and SDT are merged so that the services from the merged stream are properly referenced and PID's `0x00` to `0x1F` are dropped from the merged stream.

--no-wait

Do not wait for child process termination at end of processing.

-p *pid*[*-pid*]
--pass *pid*[*-pid*]

Pass the specified PID or range of PID's from the merged stream. Several options `--pass` can be specified.

--reset-label *label1*[*-label2*]

Clear the specified labels on the merged packets. Apply to original packets from the merged stream only, not to updated PSI packets.

Several `--reset-label` options may be specified.

--set-label *label1*[*-label2*]

Set the specified labels on the merged packets. Apply to original packets from the merged stream only, not to updated PSI packets.

Several `--set-label` options may be specified.

--terminate

Terminate packet processing when the merged stream is terminated. By default, when packet insertion is complete, the transmission continues and the stuffing is no longer modified.

-t
--transparent

Pass all PID's without logical transformation.

Equivalent to `--no-psi-merge --ignore-conflicts --pass 0x00-0x1F`.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[*-label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

mpe

Extract MPE (Multi-Protocol Encapsulation) datagrams

This plugin extracts MPE (Multi-Protocol Encapsulation) datagrams from one or more PID's. The extracted datagrams can be either forwarded on the local network, saved in a binary file or simply logged for monitoring. See [14] for more details on MPE.

The extracted datagrams must be valid UDP/IP datagrams. Otherwise, they are ignored. When saved in a binary file or forwarded on the network, only the UDP payload is used. The original IP and UDP headers are dropped.

If the extracted datagrams are forwarded on the local network, it is recommended to activate the real-time defaults of *tsp* using the option `--realtime` (see the reference documentation for *tsp*).

Usage

```
tsp -P mpe [options]
```

General options

-m *value*
--max-datagram *value*

Specify the maximum number of datagrams to extract, then stop. By default, all datagrams are extracted.

-p *pid1*[*-pid2*]
--pid *pid1*[*-pid2*]

Extract MPE datagrams from these PID's. Several `-p` or `--pid` options may be specified. When no PID is specified, use all PID's carrying MPE which are properly declared in the signalization.

MPE filtering options

-d *address*[*:port*]
--destination *address*[*:port*]

Filter MPE UDP datagrams based on the specified destination IP address.

--net-size *value*

Specify the exact size in bytes of the complete network datagrams to filter, including IP headers.

This option is incompatible with `--min-net-size` and `--max-net-size`.

--min-net-size *value*

Specify the minimum size in bytes of the complete network datagrams to filter.

--max-net-size *value*

Specify the maximum size in bytes of the complete network datagrams to filter.

-s *address*[*:port*]
--source *address*[*:port*]

Filter MPE UDP datagrams based on the specified source IP address.

--udp-size *value*

Specify the exact size in bytes of the UDP datagrams to filter.

This option is incompatible with `--min-udp-size` and `--max-udp-size`.

--min-udp-size *value*

Specify the minimum size in bytes of the UDP datagrams to filter.

--max-udp-size *value*

Specify the maximum size in bytes of the UDP datagrams to filter.

Display options

--dump-datagram

With --log, dump each complete network datagram.

--dump-udp

With --log, dump the UDP payload of each network datagram.

--dump-max *value*

With --dump-datagram or --dump-udp, specify the maximum number of bytes to dump. By default, dump everything.

-l
--log

Log all MPE datagrams using a short summary for each of them.

--skip *value*

With --output-file, --dump-datagram or --dump-udp, specify the initial number of bytes to skip. By default, save or dump from the beginning.

--sync-layout

With --log, display the layout of 0x47 sync bytes in the UDP payload.

Save options

-a
--append

With --output-file, if the file already exists, append to the end of the file. By default, existing files are overwritten.

-o *filename*
--output-file *filename*

Specify that the extracted UDP datagrams are saved in this file. The UDP messages are written without any encapsulation.

UDP forwarding options

--local-address *address*

With --udp-forward, specify the IP address of the outgoing local interface for multicast traffic. It can be also a host name that translates to a local address.

-r *address*[:*port*]
--redirect *address*[:*port*]

With --udp-forward, redirect all UDP datagrams to the specified socket address.

By default, all datagram are forwarded to their original destination address. If you specify a redirected address, it is recommended to use --destination to filter a specific stream.

If the port is not specified, the original destination port from the MPE datagram is used.

--ttl *value*

With --udp-forward, specify the TTL (Time-To-Live) socket option.

The actual option is either *Unicast TTL* or *Multicast TTL*, depending on the destination address.

By default, use the same TTL as specified in the received MPE encapsulated datagram.

-u
--udp-forward

Forward all received MPE encapsulated UDP datagrams on the local network.

By default, the destination address and port of each datagram is left unchanged. The source address of the forwarded datagrams will be the address of the local machine.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

mpeinject

Inject an incoming UDP stream into MPE (Multi-Protocol Encapsulation)

This plugin receives UDP datagrams from the local network, encapsulates them and inserts them in an MPE (Multi-Protocol Encapsulation) PID. See [14] for more details on MPE.

By default, the inserted PID containing MPE sections replaces null packets.

Using this plugin forces *tsp* and all plugins to use their real-time defaults (see the reference documentation for *tsp).*

Usage

```
tsp -P mpeinject [options] [[source@]address:]port
```

Parameter

The parameter [*address*:]*port* describes the destination of incoming UDP datagrams. All datagrams which are received on this stream will be MPE-encapsulated.

The *port* part is mandatory and specifies the UDP port to listen on. The *address* part is optional. It specifies an IP multicast address to listen on. It can be also a host name that translates to a multicast address.

An optional source address can be specified as *source@address:port* in the case of source-specific multicast (SSM).

If the address is not specified, the plugin simply listens on the specified local port and receives the packets which are sent to one of the local (unicast) IP addresses of the system.

UDP reception options

These options apply to the incoming UDP/IP stream from the local network.

-b *value*
--buffer-size *value*

Specify the UDP socket receive buffer size (socket option).

--default-interface

Let the system find the appropriate local interface on which to listen. By default, listen on all local interfaces.

-f
--first-source

Filter UDP packets based on the source address. Use the sender address of the first received packet as only allowed source.

This option is useful when several sources send packets to the same destination address and port. Accepting all packets could result in a corrupted stream and only one sender shall be accepted.

To allow a more precise selection of the sender, use option `--source`. Options `--first-source` and `--source` are mutually exclusive.

-l *address*
--local-address *address*

Specify the IP address of the local interface on which to listen. It can be also a host name that translates to a local address.

By default, listen on all local interfaces.

--no-reuse-port

Disable the reuse port socket option. Do not use unless completely necessary.

--receive-timeout *value*

Specify the UDP reception timeout in milliseconds. This timeout applies to each receive operation, individually.

By default, receive operations wait for data, possibly forever.

-r
--reuse-port

Set the reuse port socket option. This is now enabled by default, the option is present for legacy only.

-s *address[:port]*
--source *address[:port]*

Filter UDP packets based on the specified source address.

This option is useful when several sources send packets to the same destination address and port. Accepting all packets could result in a corrupted stream and only one sender shall be accepted.

Options `--first-source` and `--source` are mutually exclusive.

--ssm

This option forces the usage of source-specific multicast (SSM) using the source address which is specified by the option `--source`. Without `--ssm`, standard ("any-source') multicast is used and the option `--source` is used to filter incoming packets.

The `--ssm` option is implicit when the classical SSM syntax *source@address:port* is used.

MPE encapsulation options

These options specify how the incoming UDP datagrams are encapsulated into MPE sections.

--mac-address *nn:nn:nn:nn:nn:nn*

Specify the default destination MAC address to set in MPE sections for unicast IP packets. The default is `00:00:00:00:00:00`.

For multicast IP packets, the MAC address is automatically computed.

--new-destination *address[:port]*

Change the destination IP address and UDP port of the network datagram in MPE sections. If the port is not specified, the original destination port from the UDP datagram is used.

By default, the destination address is not modified.

--new-source *address[:port]*

Change the source IP address and UDP port of the network datagram in MPE sections. If the port is not specified, the original source port from the UDP datagram is used.

By default, the source address is not modified.

Other options

--max-queue *value*

Specify the maximum number of queued UDP datagrams before their insertion into the MPE stream. The default is 32.

If incoming datagrams arrive too fast and more than this number of UDP datagrams are internally buffered before having the opportunity to be inserted in the transport stream, additional datagrams are dropped and a warning message is reported.

-p *value*
--pid *value*

Specify the PID into which the MPE datagrams shall be inserted. This is a mandatory parameter.

--replace

Replace the target PID if it exists. By default, the plugin only replaces null packets and *tsp* stops with an error if incoming packets are found with the target PID.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

`--help`

Display this help text.

`--only-label` *`label1`*`[-`*`label2`*`]`

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

mux

Inject TS packets in a transport stream

This plugin injects TS packets from a file into a transport stream, replacing packets from stuffing.

Usage

```
tsp -P mux [options] input-file
```

Parameters

input-file

Binary file containing 188-byte transport packets.

Options

-b *value*
--bitrate *value*

Specifies the bitrate for the inserted packets, in bits/second. By default, all stuffing packets are replaced which means that the bitrate is neither constant nor guaranteed.

--byte-offset *value*

Start reading the file at the specified byte offset (default: 0). This option is allowed only if the input file is a regular file.

-i *value*
--inter-packet *value*

Specifies the packet interval for the inserted packets, that is to say the number of TS packets in the transport between two new packets. Use instead of `--bitrate` if the global bitrate of the TS cannot be determined.

--inter-time *value*

Specifies the time interval for the inserted packets, that is to say the difference between the nearest PCR clock value at the point of insertion in milliseconds.

Example: 1000 will keep roughly 1 second space between two inserted packets. The default is 0, it means inter-time is disabled. Use `--pts-pid` to specify the PID carrying the PCR clock of interest.

-j
--joint-termination

Perform a *joint termination* when the file insertion is complete. See the description of the `tsp` command for more details on *joint termination*.

--max-insert-count *value*

Stop inserting packets after this number of packets was inserted.

--max-pts *value*

Stop inserting packets when this PTS time has passed in the `--pts-pid`.

--min-pts *value*

Start inserting packets when this PTS time has passed in the `--pts-pid`.

--no-continuity-update

Do not update continuity counters in the inserted packets. By default, the continuity counters are updated in each inserted PID to preserve the continuity.

--no-pid-conflict-check

Do not check PID conflicts between the TS and the new inserted packets. By default, the processing is aborted if packets from the same PID are found both in the TS and the inserted packets.

--packet-offset *value*

Start reading the file at the specified TS packet (default: 0). This option is allowed only if the input file is a regular file.

-p *value*
--pid *value*

Force the PID value of all inserted packets.

--pts-pid *value*

Defines the PID carrying PCR or PTS values for `--min-pts` and `--max-pts`. When no PTS values are found, PCR are used. PCR values are divided by 300, the system clock sub-factor, to get the corresponding PTS values.

-r *count*
--repeat *count*

Repeat the playout of the file the specified number of times. By default, the file is infinitely repeated. This option is allowed only if the input file is a regular file.

--reset-label *label1*[-*label2*]

Clear the specified labels on the muxed packets.

Several `--reset-label` options may be specified.

--set-label *label1*[-*label2*]

Set the specified labels on the muxed packets.

Several `--set-label` options may be specified.

-t
--terminate

Terminate packet processing when the file insertion is complete. By default, when packet insertion is complete, the transmission continues and the stuffing is no longer modified.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

nit

Perform Various Transformations on a NIT

This plugin performs various transformations on a NIT, either the NIT Actual or some specific NIT Other. The other NIT's, if present, are left unchanged.

Usage

```
tsp -P nit [options]
```

Options

--bitrate *value*

Specifies the bitrate in bits / second of the PID containing the NIT if a new one is created.

The default is 3,000 b/s.

--cleanup-private-descriptors

Remove all private descriptors without preceding *private_data_specifier_descriptor*.

-c
--create

Create a new empty NIT if none was received after one second.

This is equivalent to `--create-after 1000`.

--create-after *milliseconds*

Create a new empty NIT if none was received after the specified number of milliseconds. If an actual NIT is received later, it will be used as the base for transformations instead of the empty one.

-i
--increment-version

Increment the version number of the NIT.

--inter-packet *value*

When a new NIT is created and `--bitrate` is not present, this option specifies the packet interval for the NIT PID, that is to say the number of TS packets in the transport between two packets of the PID.

Use instead of `--bitrate` if the global bitrate of the TS cannot be determined.

-l *value*
--lcn *value*

Specify which operation to perform on *logical_channel_number* (LCN) descriptors. The *value* is a positive integer:

1 : Remove all LCN descriptors.

2 : Remove one entry every two entries in each LCN descriptor.

3 : Duplicate one entry every two entries in each LCN descriptor.

--mpe-fec *value*

Set the *MPE-FEC_indicator* in all *terrestrial_delivery_system_descriptor*s to the specified value (0 or 1).

--network-id *id*

Set the specified new value as network id in the NIT.

--network-name *name*

Set the specified value as network name in the NIT. Any existing *network_name_descriptor* is removed. A new *network_name_descriptor* is created with the new name.

-v *value*
--new-version *value*

Specify a new value for the version of the NIT.

-o *id*
--other *id*
--nit-other *id*

Do not modify the NIT Actual. Modify the NIT Other with the specified network id.

--pds *value*

With option `--remove-descriptor`, specify the private data specifier which applies to the descriptor tag values above `0x80`.

-p *value*
--pid *value*

Specify the PID on which the NIT is expected.

By default, use PID 16 (`0x0010`), as specified for DVB-compliant networks.

--remove-descriptor *value*

Remove from the NIT all descriptors with the specified tag. Several `--remove-descriptor` options may be specified to remove several types of descriptors. See also option `--pds`.

-r *value*
--remove-service *value*

Remove the specified service_id from the following descriptors: *service_list_descriptor*, *logical_channel_number_ descriptor*. Several `--remove-service` options may be specified to remove several services.

--remove-ts *value*

Remove from the NIT all references to the transport stream with the specified *ts_id* value. Several `--remove-ts` options may be specified to remove several TS.

-s *value*
--sld *value*

Specify which operation to perform on *service_list_descriptors*. The *value* is a positive integer:

1 : Remove all *service_list_descriptors*.

2 : Remove one entry every two entries in each *service_list_descriptor*.

--time-slicing *value*

Set the *Time_Slicing_indicator* in all *terrestrial_delivery_system_descriptor*s to the specified value (0 or 1).

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[*-label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

nitscan

Scan NIT for Tuning Information

This plugin analyzes the NIT (Network Information Table) of the transport stream and outputs a list of tuning information, one per transport. The format of the tuning information is compatible with the *dvb* input plugin and the standard Linux utilities *szap*, *czap* and *tzap*.

Usage

```
tsp -P nitscan [options]
```

Options

-a
--all-nits

Analyze all NIT's ("NIT actual" and "NIT other").

By default, only the "NIT actual" is analyzed.

-c[*prefix*]
--comment[=*prefix*]

Add a comment line before each tuning information. The optional prefix designates the comment prefix. If the option `--comment` is present but the prefix is omitted, the default prefix is "`# `".

-d
--dvb-options

The characteristics of each transponder are formatted as a list of command-line options for the *dvb* input plugin such as `--frequency`, `--symbol-rate`, etc.

This is the default when no `--save-channels` or `--update-channels` is specified.

-n *value*
--network-id *value*

Specify the network-id of a "NIT other" to analyze instead of the "NIT actual".

By default, the "NIT actual" is analyzed.

-o *filename*
--output-file *filename*

Specify the output text file for the analysis result. By default, use the standard output.

Warning: if you do not specify this option, be sure to redirect the output plugin to something different from the default. Otherwise, the text output of the analysis will be mixed with the binary output of the TS packets!

-p *value*
--pid *value*

Specify the PID on which the NIT is expected. By default, the PAT is analyzed to get the PID of the NIT. DVB-compliant networks should use PID 16 (`0x0010`) for the NIT and signal it in the PAT.

--save-channels *filename*

Save the description of all transport streams in the specified XML file. See Appendix B, page 293, for more details on channels configuration files.

If the file name is "`-`", use the default tuning configuration file.

See also option `--update-channels`.

-t
--terminate

Stop the packet transmission after the first NIT is analyzed. Should be specified when *tsp* is used only to scan the NIT.

--update-channels *filename*

Update the description of all transport streams in the specified XML file. The content of each transport stream is preserved, only the tuning information is updated. If the file does not exist, it is created. See Appendix B, page 293, for more details on channels configuration files.

If the file name is "-", use the default tuning configuration file.

See also option --save-channels.

-v[*prefix*]
--variable[=*prefix*]

Each tuning information line is output as a shell environment variable definition. The name of each variable is built from a prefix and the TS id. The default prefix is "TS" and can be changed through the optional value of the option --variable.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several --only-label options may be specified.

null (input)

Null Input Packets Generator

This input module generates null packets.

Usage

```
tsp -I null [options] [count]
```

Parameters

count

Specify the number of null packets to generate. After the last packet, an end-of-file condition is generated. By default, if *count* is not specified, null packets are generated endlessly.

Options

-j
--joint-termination

When the number of null packets is specified, perform a *joint termination* when completed instead of unconditional termination. See the description of the `tsp` command for more details on *joint termination*.

Generic common input plugin options

The following options are implicitly defined in all input plugins.

--help

Display plugin help text.

pat

Perform Various Transformations on the PAT

This plugin performs various transformations on the PAT.

Usage

```
tsp -P pat [options]
```

Options

-a *sid/pid*
--add-service *sid/pid*

Add the specified *service_id* / *PMT-PID* in the PAT. Several `--add-service` options may be specified to add several services.

--bitrate *value*

Specifies the bitrate in bits / second of the PID containing the PAT if a new one is created.

The default is 3,000 b/s.

-c
--create

Create a new empty PAT if none was received after one second.

This is equivalent to `--create-after 1000`.

--create-after *milliseconds*

Create a new empty PAT if none was received after the specified number of milliseconds. If an actual PAT is received later, it will be used as the base for transformations instead of the empty one.

-i
--increment-version

Increment the version number of the PAT.

--inter-packet *value*

When a new PAT is created and `--bitrate` is not present, this option specifies the packet interval for the PAT PID, that is to say the number of TS packets in the transport between two packets of the PID.

Use instead of `--bitrate` if the global bitrate of the TS cannot be determined.

-n *pid*
--nit *pid*

Add or modify the NIT PID in the PAT.

-r *sid*
--remove-service *sid*

Remove the specified *service_id* from the PAT. Several `--remove-service` options may be specified to remove several services.

-u
--remove-nit

Remove the NIT PID from the PAT.

-t *id*
--ts-id *id*
--tsid *id*

Specify a new value for the transport stream id in the PAT.

-v *value*
--new-version *value*

Specify a new value for the version of the PAT.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

pattern

Replace Packet Payload with a Binary Pattern

This plugin replaces the payload of TS packets with a binary pattern on selected PID's. The resulting packets are meaningless on an MPEG standpoint but can be used to trace packets in order to debug transport stream routing problems either inside a transmission system or inside a set-top box.

Usage

```
tsp -P pattern [options] pattern
```

Parameter

Specifies the binary pattern to apply on TS packets payload. The value must be a string of hexadecimal digits specifying any number of bytes.

Options

-n
--negate

Negate the PID filter: modify packets on all PID's, expect the specified ones.

-o *value*

--offset-non-pusi *value*

Specify starting offset in payload of packets with the PUSI (payload unit start indicator) not set. By default, the pattern replacement starts at the beginning of the packet payload (offset 0).

-u *value*
--offset-pusi *value*

Specify starting offset in payload of packets with the PUSI (payload unit start indicator) set. By default, the pattern replacement starts at the beginning of the packet payload (offset 0).

-p *pid1*[*-pid2*]
--pid *pid1*[*-pid2*]

Select packets with these PID values. Several -p or --pid options may be specified to select multiple PID's. If no such option is specified, packets from all PID's are modified.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[*-label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

pcradjust

Adjust PCR's according to a constant bitrate

This plugin recomputes all PCR values, assuming that the transport stream has a constant bitrate.

In the general case, it is impossible to recompute PCR values in non-real-time streams with a variable bitrate because the instant bitrate is usually computed according to the PCR values which are found in the stream, hence assuming that these PCR values are correct and do not need any adjustment.

In each PID, the first PCR is left unmodified and all others are recomputed according to the constant bitrate and the distance between packets.

Usage

```
tsp -P pcradjust [options]
```

Options

-b value
--bitrate value

Specify a constant bitrate for the transport stream. The PCR values will be adjusted according to this bitrate. By default, use the input bitrate as reported by the input device or a previous plugin.

--ignore-dts

Do not modify DTS (decoding time stamps) values.

By default, the DTS are modified according to the PCR adjustment.

--ignore-pts

Do not modify PTS (presentation time stamps) values.

By default, the PTS are modified according to the PCR adjustment.

--ignore-scrambled

Do not modify PCR values on PID's containing scrambled packets.

By default, on scrambled PID's, the PCR's are modified but not the PTS and DTS since they are scrambled. This may result in problems when playing video and audio.

--min-ms-interval *milliseconds*

Specify the minimum interval between two PCR's in miliseconds.

On a given PID, if the interval between two PCR's is larger than the minimum, the next null packet will be replaced with an empty packet with a PCR for that PID.

-p *pid1*[*-pid2*]
--pid *pid1*[*-pid2*]

Specifies PID's where PCR, DTS and PTS values shall be adjusted.

Several `--pid` options may be specified.

By default, all PID's are modified.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[*-label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

pcrbitrate

Permanently Recompute Bitrate Based on PCR's

This plugin permanently recomputes the bitrate based on the analysis of PCR's on the packets. All packets are transparently passed.

Normally, tsp determines the input bitrate at the input plugin: either the input plugin itself can report the actual input bitrate (from a hardware device for instance) or tsp computes the bitrate based on PCR analysis. Then, the bitrate information is automatically propagated from one plugin to another, up to the output plugin. The output plugin may use or ignore this information. Typically, output to a file ignores the bitrate information while output to a hardware device (ASI or modulator) will use it as device parameter.

There may be a problem if some packet processor plugin drops packets from the transport stream. The *zap* plugin, for instance, creates an SPTS containing only one service, dropping all other packets.

Let's take an example: tsp is used to read a full MPTS from a file, extract one channel and send it to a Dektec ASI device. Tsp reads the input bitrate (here, it analyzes the PCR from the input file and finds, say, 38 Mb/s). Then, tsp propagates this bitrate along the plugin chain, up to the output plugin. By default, the output plugin will send the SPTS at 38 Mb/s, the bitrate of the original MPTS, which is a non-sense since the "normal" bitrate of the SPTS is more likely something like 3 or 4 Mb/s. By inserting the *pcrbitrate* plugin between the *zap* plugin and the *dektec* output plugin, the bitrate information will be altered and the output plugin receives a bitrate value which is consistent with the PCR's in the SPTS.

Usage

```
tsp -P pcrbitrate [options]
```

Options

-d
--dts

Use DTS (Decoding Time Stamps) from video PID's instead of PCR (Program Clock Reference) from the transport layer.

-i
--ignore-errors

Ignore transport stream errors such as discontinuities.

When errors are not ignored (the default), the bitrate of the original stream (before corruptions) is evaluated. When errors are ignored, the bitrate of the received stream is evaluated, missing packets being considered as non-existent.

--min-pcr *value*

Stop analysis when that number of PCR are read from the required minimum number of PID (default: 128).

--min-pid *value*

Minimum number of PID to get PCR from (default: 1).

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

pcrextract

Extracts PCR, OPCR, PTS, DTS from TS packets

This plugin extracts PCR, OPCR, PTS, DTS from TS packets. The output is typically suitable for analysis with tools like Microsoft Excel.

Usage

```
tsp -P pcrextract [options]
```

Options

-c
--csv

Report data in CSV (*comma-separated values*) format. All values are reported in decimal. This is the default output format. It is suitable for later analysis using tools such as Microsoft Excel.

-d
--dts

Report Decoding Time Stamps (DTS). By default, if none of `--pcr`, `--opcr`, `--pts`, `--dts` is specified, report them all.

-e
--evaluate-pcr-offset

Evaluate the offset from the PCR to PTS/DTS for packets with PTS/DTS but without PCR. This evaluation may be incorrect if the bitrate is not constant or incorrectly estimated.

By default, the offset is reported only for packets containing a PTS/DTS and a PCR.

-g
--good-pts-only

Keep only "good" PTS, ie. PTS which have a higher value than the previous good PTS. This eliminates PTS from out-of-sequence B-frames.

-l
--log

Report data in "*log*" format through the standard *tsp* logging system. All values are reported in hexadecimal.

-n
--noheader

Do not output initial header line in CSV format.

--opcr

Report Original Program Clock References (OPCR). By default, if none of `--pcr`, `--opcr`, `--pts`, `--dts` is specified, report them all.

-o *filename*
--output-file *filename*

Output file name for CSV format (standard error by default).

--pcr

Report Program Clock References (PCR). By default, if none of `--pcr`, `--opcr`, `--pts`, `--dts` is specified, report them all.

-p *pid1*`[`*-pid2*`]`
--pid *pid1*`[`*-pid2*`]`

Specifies PID’s to analyze. By default, all PID's are analyzed. Several `--pid` options may be specified.

--pts

Report Presentation Time Stamps (PTS). By default, if none of --pcr, --opcr, --pts, --dts is specified, report them all.

--scte35

Also detect and report PTS in SCTE 35 commands. This option forces --log and --pts.

If no --pid option is specified, detect all PID's carrying SCTE 35 splice information.

If some --pid options are specified, they designate PID's carrying PCR or PTS. In that case, SCTE 35 commands are analyzed only from PID's which are referenced by the same services as the specified --pid options.

-s *string*
--separator *string*

Field separator string in CSV format (default: ';').

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several --only-label options may be specified.

pcrverify

Verify the PCR's Values

This plugin verifies the values of all PCR's and report invalid values. Each PCR is compared to its expected theoretical value as computed from the previous PCR value and the transport bitrate.

Usage

```
tsp -P pcrverify [options]
```

Options

-a
--absolute

Use absolute values in PCR units. By default, use micro-second equivalent values (one micro-second = 27 PCR units).

-b *value*
--bitrate *value*

Verify the PCR's according to this transport bitrate. By default, use the input bitrate as reported by the input device.

-j *value*
--jitter-max *value*

Maximum allowed jitter. PCR's with a higher jitter are reported, others are ignored. If `--absolute`, the specified value is in PCR units, otherwise it is in micro-seconds. The default is 27,000 PCR units or 1,000 micro-seconds. Use `--jitter 0` to check that all PCR have their exact expected value.

-p *pid1*[*-pid2*]
--pid *pid1*[*-pid2*]

PID filter: select packets with these PID values. Several `-p` or `--pid` options may be specified. Without `-p` or `--pid` option, PCR's from all PID's are used.

-t
--time-stamp

Display time of each event.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[*-label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

pes

Analyze PES Packets

This plugin detects and analyzes PES packets in all selected PID's (all PID's by default). Note that, without any option, this plugin does not report anything, you need to specify what you want to analyze.

Usage

```
tsp -P pes [options]
```

Options

-a
--audio-attributes

Display audio attributes such as audio layer, stereo mode or sampling rate in MPEG-1 audio (ISO/IEC 11172-3), MPEG-2 audio (ISO/IEC 13818-3), AC-3 and Enhanced-AC-3 (ETSI TS 102 366).

--avc-access-unit

Dump all AVC (ISO/IEC 14496-10, ITU H.264) access units (aka "NALunits").

-b
--binary

Include binary dump in addition to hexadecimal.

-h
--header

Dump all PES packets header.

-x *value*
--max-dump-count *value*

Specify the maximum number of times data dump occurs with options `--trace-packets`, `--header`, `--payload`, `--start-code`, `--avc-access-unit`. Default: unlimited.

-m *value*
--max-dump-size *value*

Specify the maximum dump size for options `--header`, `--payload`, `--start-code`, `--avc-access-unit`. By default, the complete data section (payload, access unit, etc.) is displayed.

--max-payload-size *value*

Display PES packets with no payload or with a payload the size (in bytes) of which is not greater than the specified value.

--min-payload-size *value*

Display PES packets with a payload the size (in bytes) of which is equal to or greater than the specified value.

--nal-unit-type *value*

AVC NAL unit filter: with `--avc-access-unit`, select access units with this type (default: all access units). Several `--nal-unit-type` options may be specified.

--negate-nal-unit-type

Negate the AVC NAL unit filter: specified access units are excluded.

-n
--negate-pid

Negate the PID filter: specified PID's are excluded.

--nibble

Same as `--binary` but add separator between 4-bit nibbles.

-o *filename*
--output-file *filename*

Specify the output file for the report (default: standard output).

--packet-index

Display the index of the first and last TS packet of each displayed PES packet.

-p *pid1*[*-pid2*]
--pid *pid1*[*-pid2*]

PID filter: select packets with this PID value (default: all PID's containing PES packets). Several `-p` or `--pid` options may be specified.

--payload

Dump all PES packets payload.

--sei-avc

Dump all SEI (Supplemental Enhancement Information) in AVC / H.264 access units.

--sei-type *value*

SEI type filter: with `--sei-avc`, select SEI access units with this type (default: all SEI access units).

Several `--sei-type` options may be specified.

-s
--start-code

Dump all start codes in PES packet payload.

-t
--trace-packets

Trace all PES packets (display a one-line description per packet).

--uuid-sei *value*

AVC SEI filter: with `--sei-avc`, only select *user data unregistered* SEI access units with the specified UUID value. By default, with `--sei-avc`, all SEI are displayed.

Several `–uuid-sei` options may be specified.

The UUID value must be 16 bytes long. It must be either an ASCII string of exactly 16 characters or a hexadecimal value representing 16 bytes.

-v
--video-attributes

Display video attributes such as frame size, frame rate or profile in MPEG-1 video (ISO/IEC 11172-2), MPEG-2 video (ISO/IEC 13818-2) and AVC (ISO/IEC 14496-10, ITU H.264).

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[*-label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

play (output)

Play Output on a Media Player

This output plugin sends TS packets to a supported media player. It is typically used when one service was isolated on the transport stream and the resulting audio/video must be monitored.

The *play* plugin attempts to locate a media player application which can process MPEG-2 transport streams on its standard input. If one is found in the system, the plugin creates a process executing the media player (adding the required options if necessary) and sends the output stream to this process using a pipe.

This plugin is consequently is easier alternative to the *fork* plugin. The same operation could be achieved using the *fork* plugin but it requires to specify the complete media player command line with options.

Using this plugin forces *tsp* and all plugins to use their real-time defaults (see the reference documentation for *tsp).*

Usage

```
tsp -O play [options]
```

Options

-m
--mplayer

Linux only: Use *mplayer* for rendering. The default is to look for *vlc*, *mplayer* and *xine*, in this order, and use the first available one.

-x
--xine

Linux only: Use *xine* for rendering. The default is to look for *vlc*, *mplayer* and *xine*, in this order, and use the first available one.

Generic common output plugin options

The following options are implicitly defined in all output plugins.

--help

Display plugin help text.

Supported media players

- Linux: Look for VLC, *mplayer* and *xine*. Use the `PATH` environment variable to locate the applications.
- macOS: Same a Linux but also search into `/usr/local/bin` and `/Applications`.
- Windows: Look for VLC using the `Path` environment variable and various informations that are normally filled in the registry by the VLC installation procedure. See [28] for downloading and installing VLC Media Player.

To use another media player or with specific options, use the *fork* plugin instead:

```
tsp ... -P fork [options] "media player command line" -O drop
```

pmt

Perform Various Transformations on a PMT

This plugin performs various transformations on a PMT.

The PMT can be specified by PID, by service id or by service name.

Usage

```
tsp -P pmt [options]
```

Options

--ac3-atsc2dvb

Change the description of AC-3 (a.k.a. DD, Dolby Digital) audio streams from ATSC to DVB method. In details, this means that all components with stream_type `0x81` are modified with stream_type `0x06` (*PES private data*) and an *AC-3_descriptor* is added on this component (if none was already there).

--add-ca-descriptor *casid/pid[/private-data]*

Add a *CA_descriptor* at program-level in the PMT with the specified CA System Id and ECM PID. The optional private data must be a suite of hexadecimal digits. Several `–add-ca-descriptor` options may be specified to add several descriptors.

-a *pid/type*
--add-pid *pid/type*

Add the specified PID / stream-type component in the PMT. Both *PID* and *type* must be integer values, either decimal or hexadecimal. Several `--add-pid` options may be specified to add several components.

--add-pid-registration *pid/id*

Add a *registration_descriptor* in the descriptor list of the specified PID in the PMT.

The value is the *format_identifier* in the *registration_descriptor*, e.g. `0x43554549` for “CUEI”.

--add-registration *id*

Add a *registration_descriptor* in the program-level descriptor list in the PMT.

The value is the *format_identifier* in the *registration_descriptor*, e.g. `0x43554549` for “CUEI”.

--add-stream-identifier

Add a *stream_identifier_descriptor* on all components. The *component_tag* are uniquely allocated inside the service. Existing *stream_identifier_descriptor*s are left unmodified.

--audio-language ***language-code*[:*audio-type*[:*location*]]**

Specifies the language for an audio stream in the PMT. Several options can be specified to set the languages of several audio streams.

The *language-code* is a 3-character string. The *audio-type* is optional, its default value is zero. The *location* indicates how to locate the audio stream. Its format is either "P*n*" or "A*n*". In the first case, "*n*" designates a PID value and in the second case the audio stream number inside the PMT, starting with 1. The default location is "A1", ie. the first audio stream inside the PMT.

--bitrate *value*

Specifies the bitrate in bits / second of the PID containing the PMT if a new one is created.

The default is 3,000 b/s.

--cleanup-private-descriptors

Remove all private descriptors without preceding *private_data_specifier_descriptor*.

-c
--create

Create a new empty PMT if none was received after one second.

This is equivalent to `--create-after 1000`.

--create-after *milliseconds*

Create a new empty PMT if none was received after the specified number of milliseconds. If an actual PMT is received later, it will be used as the base for transformations instead of the empty one.

--eac3-atsc2dvb

Change the description of Enhanced-AC-3 (a.k.a. AC-3+, DD+, Dolby Digital+) audio streams from ATSC to DVB method. In details, this means that all components with stream_type `0x87` are modified with stream_type `0x06` (*PES private data*) and an *enhanced_AC-3_descriptor* is added on this component (if none was already there).

--increment-version

Increment the version number of the PMT.

--inter-packet *value*

When a new PMT is created and `--bitrate` is not present, this option specifies the packet interval for the PMT PID, that is to say the number of TS packets in the transport between two packets of the PID.

Use instead of `--bitrate` if the global bitrate of the TS cannot be determined.

-m *old-pid/new-pid*
--move-pid *old-pid/new-pid*

Change the PID value of a component in the PMT. Several `--move-pid` options may be specified to move several components.

-i *value*
--new-service-id *value*

Change the service id in the PMT.

--pds *value*

With option `--remove-descriptor`, specify the private data specifier which applies to the descriptor tag values above `0x80`.

-p *value*
--pmt-pid *value*

Specify the PID carrying the PMT to modify. All PMT's in this PID will be modified. Options `--pmt-pid` and `--service` are mutually exclusive. If neither are specified, the first service in the PAT is used.

--pcr-pid *value*

Change the PCR PID value in the PMT.

--remove-descriptor *value*

Remove from the PMT all descriptors with the specified tag. Several `--remove-descriptor` options may be specified to remove several types of descriptors. See also option `--pds`.

-r `pid1`*`[-pid2]`*
--remove-pid `pid1`*`[-pid2]`*

Remove the component with the specified PID's from the PMT. Several `--remove-pid` options may be specified to remove several components.

--remove-stream-type *value[-value]*

Remove all components with a stream type matching the specified values. Several `--remove-stream-type` options may be specified.

-s *name-or-id*
--service *name-or-id*

Specify the service the PMT of which must be modified. If the argument is an integer value (either decimal or hexadecimal), it is interpreted as a service id. Otherwise, it is interpreted as a service

name, as specified in the SDT. The name is not case sensitive and blanks are ignored. Options `--pmt-pid` and `--service` are mutually exclusive. If neither are specified, the first service in the PAT is used.

`--set-cue-type` *pid/type*

In the component with the specified PID, add an SCTE 35 *cue_identifier_descriptor* with the specified *cue_stream_type*. Several `--set-cue-type` options may be specified.

`--set-data-broadcast-id` *pid/id[/selector]*

In the component with the specified PID, add a *data_broadcast_id_descriptor* with the specified *data_broadcast_id*. The optional selector is a suite of hexadecimal characters representing the content of the selector bytes. Several `--set-data-broadcast-id` options may be specified.

`--set-stream-identifier` *pid/id*

In the component with the specified PID, add a *stream_identifier_descriptor* with the specified id as *component_tag*. Several `--set-stream-identifier` options may be specified.

`-v` *value*
`--new-version` *value*

Specify a new value for the version of the PAT.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

`--help`

Display this help text.

`--only-label` *label1*[*-label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

psi

Collect PSI Structure Information

This plugin extracts all PSI tables (PAT, CAT, PMT, NIT, BAT, SDT) from a transport stream. It is equivalent to the *tspsi* utility. Actually, the following two commands produce the same result:

```
tspsi options filename
tsp -I file filename -P psi options -O drop
```

Usage

```
tsp -P psi [options]
```

Options

The plugin accepts exactly the same options as the *tspsi* utility.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

psimerge

Merge PSI/SI from mixed streams

This plugin assumes that the PSI/SI for two independent streams are multiplexed in the same transport streams but the packets from each original stream are independently labelled. This plugin merges the PSI/SI from these two streams into one.

Usage

```
tsp -P psimerge [options]
```

Options

--main-label *value*

Specify the label which is set on packets from the *main* stream. The maximum label value is 31.

By default, the main stream is made of packets without label.

At least one of `--main-label` and `--merge-label` must be specified.

--merge-label *value*

Specify the label which is set on packets from the *merge* stream. The maximum label value is 31.

By default, the merge stream is made of packets without label.

At least one of `--main-label` and `--merge-label` must be specified.

--no-bat

Do not merge the BAT.

--no-cat

Do not merge the CAT.

--no-eit

Do not merge the EIT's.

--no-nit

Do not merge the NIT Actual.

--no-pat

Do not merge the PAT.

--no-sdt

Do not merge the SDT Actual.

--time-from-merge

Use the TDT/TOT time reference from the *merge* stream.

By default, use the TDT/TOT time reference from the *main* stream.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

reduce

Reduce the Bitrate by Removing Stuffing Packets

This plugin reduces the bitrate of the transport stream by removing stuffing packets.

Usage

```
tsp -P reduce [options] rempkt inpkt
```

Parameters

The parameters specify that *rempkt* TS packets must be automatically removed after every *inpkt* input TS packets in the transport stream. Only stuffing packets can be removed. Both *rempkt* and *inpkt* must be non-zero integer values.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

regulate

Regulate Packets Flow According to a Bitrate or PCR

This plugin regulates the TS packets flow according to a specified bitrate or based on the Program Clock Reference from the transport stream.

It is useful to play a non-regulated input (such as a TS file) to a non-regulated output (such as IP multicast). Without this plugin, in this example, the IP packets will be sent as fast as the TS packets are read from the file, that is to say at a very much higher bitrate than expected. When inserted between the input and the output plugins, the *regulate* plugin regularly suspends the *tsp* process to slow down the output.

Note that this plugin can only slow down the stream but not accelerate it (if the input is not fast enough, there is nothing that a plugin can do!)

By default, the plugin uses a bitrate value. The plugin suspends the execution at regular intervals to ensure that its output does not exceed the target bitrate. A fixed bitrate can be specified. Otherwise, the plugin uses the bitrate information coming from the previous plugins in the chain. In the latter case, the bitrate can be variable.

When the option `--pcr-synchronous` is specified, the plugin does not use any bitrate information. It regulates the flow to be synchronous with the Program Clock Reference (PCR) in the transport stream.

Using this plugin forces *tsp* and all plugins to use their real-time defaults (see the reference documentation for *tsp).*

Usage

```
tsp -P regulate [options]
```

Options

-b *value*
--bitrate *value*

Specify the bitrate in b/s. By default, use the input bitrate, typically resulting from the PCR analysis of the input stream. Note that this default is the bitrate which is presented by tsp at the input of the *regulate* plugin. This is not necessarily the bitrate at the input plugin if another plugin (such as *pcrbitrate*) has altered the bitrate between the input plugin and *regulate*.

-p *value*
--packet-burst *value*

Number of packets to burst at a time. Does not modify the average output bitrate but influence smoothing and CPU load. The default is 16 packets.

It is inefficient, and most of the time impossible, to suspend a process too often and for a too short time. To regulate a stream at 38 Mb/s, for instance, the process must be suspended 40 micro-seconds between each TS packets. This is not possible in practice on most Linux or Windows kernels with the default configuration. If the packet burst is set to 64, the wait time is 2.5 milli-seconds, which becomes feasible.

--pcr-synchronous

Regulate the flow based on the Program Clock Reference from the transport stream.

By default, use a bitrate, not PCR's.

--pid-pcr *value*

With `--pcr-synchronous`, specify the reference PID for the Program Clock Reference.

By default, use the first PID containing PCR's.

--wait-min *value*

With `--pcr-synchronous`, specify the minimum wait time in milli-seconds.

The default is 50 ms.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

`--help`

Display this help text.

`--only-label` *`label1`*`[-`*`label2`*`]`

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

remap

Generic PID Remapping

This plugin modifies the PID value in selected packets. By default, the PSI are modified accordingly to preserve the consistency of the transport stream.

Usage

```
tsp -P remap [options] [pid[-pid]=newpid ...]
```

Specifying PID remapping

Each remapping is specified as "*pid=newpid*" or "*pid1-pid2=newpid*". All PID's can be specified as decimal or hexadecimal values. More than one PID remapping can be specified.

In the first form, the PID *pid* is remapped to *newpid*.

In the latter form, all PID's within the range *pid1* to *pid2* (inclusive) are respectively remapped to *newpid*, *newpid*+1, etc. (this behaviour is changed using option `--single`).

The null PID `0x1FFF` cannot be remapped.

Options

`-n`
`--no-psi`

Do not modify the PSI.

By default, the PAT, CAT and PMT's are modified so that previous references to the remapped PID's will point to the new PID values.

`--reset-label` *label1*[-*label2*]

Clear the specified labels on the remapped packets.

Several `--reset-label` options may be specified.

`--set-label` *label1*[-*label2*]

Set the specified labels on the remapped packets.

Several `--set-label` options may be specified.

`-s`
`--single`

When a remapping is in the form "*pid1-pid2=newpid*", remap all input PID's within the range *pid1* to *pid2* to the same *newpid* value, not *newpid*, *newpid*+1, etc.

This option forces `--unchecked` since distinct PID's are remapped to the same one.

`-u`
`--unchecked`

Do not perform any consistency checking while remapping PID's:

- Remapping to or from a predefined PID is accepted.
- Remapping two PID's to the same PID or to a PID which is already present in the input is accepted.

Note that this option should be used with care since the resulting stream can be illegal or inconsistent.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

`--help`

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

rmorphan

Remove unreferenced PID's

This plugin removes unreferenced (aka "*orphan*") PID's from the transport stream. The plugin analyses the complete TS structure, starting from the PAT and the CAT. Any packet which neither belongs to a predefined PID's nor to a referenced PID in the TS structure is removed.

Usage

```
tsp -P rmorphan [options]
```

Options

-s
--stuffing

Replace excluded packets with stuffing (null packets) instead of removing them. Useful to preserve bitrate.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

rmsplice

Remove ads insertions using SCTE 35 splice information

This plugin removes part of a program (typically ads insertions) based on SCTE 35 splice cueing information.

According to the SCTE 35 standard (see [17]), a dedicated stream is declared in the PMT of a service, carrying private tables. These private tables describe upcoming *splice points*. They define specific points in the program where the audio and video can be "cut" and replaced by some alternate content, typically local ads sequences. *Splice out* points define places where the main program can be left to switch to local content. *Splice in* points define places where the content should return back to the original program.

The plugin *rmsplice* uses the specific SCTE 35 splice information stream to locate what could be uninteresting sequences of ads and simply removes the program content, audio, video, subtitles, during these sequences. The content of the program is not replaced, as originally intended by the SCTE 35 standard, it is simply removed. Consequently, using this plugin makes sense on SPTS only (see the plugin *zap* for instance).

The removal is based on Presentation Time Stamps (PTS) in the various content PID's of the program. The PTS of the starting (*splice out*) and ending (*splice in*) points are defined be the SCTE 35 commands in the dedicated stream. Currently, *rmsplice* removes entire PES packets and does not dig into the video encoding.

If the original video encoding is carefully performed to resist to identified splice points, the transition should be smooth. However, it has been observed transient glitches and macro blocks in the resulting stream after removing ads sequences, even though the PTS of the splice points exactly match the signalled PTS values. VLC reports one "*unref short failure*" at that point. It is currently unknown if this is due to a non-splice-resistant video encoding or if the cutting method of *rmsplice* is too harsh.

Usage

```
tsp -P rmsplice [options] [service]
```

Parameter

The optional parameter specifies the service to modify.

If this is an integer value (either decimal or hexadecimal), it is interpreted as a service id. Otherwise, it is interpreted as a service name, as specified in the SDT. The name is not case sensitive and blanks are ignored. If the input TS does not contain an SDT, use a service id.

When the parameter is omitted, the first service which is found in the PAT is selected.

Options

-a
--adjust-time

Adjust all time stamps (PCR, OPCR, PTS and DTS) after removing splice-out / splice-in sequences. This can be necessary to improve the video transition.

-c
--continue

Continue stream processing even if no "splice information stream" is found for the service. Without this information stream, ads cannot be located and consequently not removed. By default, *tsp* aborts when the splice information stream is not found in the PMT of the service.

-n
--dry-run

Perform a dry run, report what operations would be performed. Use with `--verbose`.

--event-id *pid1*[-*pid2*]

Only remove splices associated with the specified event ID's.

Several `--event-id` options may be specified.

-f
--fix-cc

Fix continuity counters after removing splice-out / splice-in sequences.

-s
--stuffing

Replace excluded packets with stuffing (null packets) instead of removing them. Useful to preserve bitrate.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

scrambler

DVB Scrambler

This plugin is a DVB scrambler, either using a static control word or using an external ECMG. In the latter case, the plugin generates the control words, schedules crypto-periods and inserts ECM's.

The control words are generated using the default pseudo-random number generator of the operating system with additional security improvements. Although these values are reasonably random, there is no security commitment and this scrambler should be used for test purpose only, not for production.

When inserting ECM's, the plugin uses the *delay_start* parameter, as returned by the ECMG, to synchronize the start of the crypto-period with the first insertion of an ECM. Both positive and negative *delay_start* values are supported.

Usage

```
tsp -P scrambler [options] [service]
```

Parameter

The optional parameter specifies the service to scramble. If no service is specified, a list of PID's to scramble must be provided using `--pid` options. When specific PID's are provided, fixed control words must be specified as well.

If no fixed CW is specified, a random CW is generated for each crypto-period and ECM's containing the current and next CW's are created and inserted in the stream. ECM's can be created only when a service is specified.

If the argument is an integer value (either decimal or hexadecimal), it is interpreted as a service id. Otherwise, it is interpreted as a service name, as specified in the SDT. The name is not case sensitive and blanks are ignored. If the input TS does not contain an SDT, use service ids only.

General options

-b *value*
--bitrate-ecm *value*

Specifies the bitrate for ECM PID's in bits / second. The default is 30,000 b/s.

-d *seconds*
--cp-duration *seconds*

Specifies the crypto-period duration in seconds (default: 10 seconds).

--ignore-scrambled

Ignore packets which are already scrambled. Since these packets are likely scrambled with a different control word, descrambling will not be possible the usual way.

--no-audio

Do not scramble audio components in the selected service. By default, all audio components are scrambled.

--no-video

Do not scramble video components in the selected service. By default, all video components are scrambled.

--partial-scrambling *count*

Do not scramble all packets, only one packet every *count* packets. The default value is 1, meaning that all packets are scrambled. Specifying higher values is a way to reduce the scrambling CPU load while keeping the service "mostly" scrambled.

-p *pid1*[*-pid2*]
--pid *pid1*[*-pid2*]

Scramble packets with these PID values. Several `-p` or `--pid` options may be specified. By default, scramble the specified service.

--pid-ecm *value*

Specifies the new ECM PID for the service. By defaut, use the first unused PID immediately following the PMT PID. Using the default, there is a risk to later discover that this PID is already used. In that case, specify `--pid-ecm` with a notoriously unused PID value.

--subtitles

Scramble subtitles components in the selected service. By default, the subtitles components are not scrambled.

--synchronous

Specify to synchronously generate the ECM's.

In real-time mode, the processing of packets continues in parallel while ECM's are generated in the ECMG. Use this option to force the stream processing to wait for ECM's.

In offline mode, this option is always on. This is usually the right thing to do. Otherwise, if an ECM takes too long to be generated, the stream processing may reach the first insertion point of the ECM before it is available.

DVB SimulCrypt options

-a *value*
--access-criteria *value*

Specifies the access criteria for the service as sent to the ECMG. The value must be a suite of hexadecimal digits.

--channel-id *value*

Specifies the DVB SimulCrypt *ECM_channel_id* for the ECMG (default: 1).

--component-level

Add *CA_descriptors* at component level in the PMT. By default, one *CA_descriptor* is added at program level.

-i *value*
--ecm-id *value*

Specifies the DVB SimulCrypt *ECM_id* for the ECMG (default: 1).

-e *host*:*port*
--ecmg *host*:*port*

Specify an ECM Generator host name (or IP address) and TCP port. Without ECMG, a fixed control word must be specified using `--control-word`.

-v *value*
--ecmg-scs-version *value*

Specifies the version of the ECMG <=> SCS DVB SimulCrypt protocol. Valid values are 2 and 3. The default is 2.

--log-data[=*level*]

Same as `--log-protocol` but applies to *CW_provision* and *ECM_response* messages only.

To debug the session management without being flooded by data messages, use `--log-protocol=info --log-data=debug`.

--log-protocol[=*level*]

Log all ECMG ⇔ SCS protocol messages using the specified level. If the option is not present, the messages are logged at debug level only. If the option is present without value, the messages are logged at info level. A level can be a numerical debug level or any of the following: `fatal`, `severe`, `error`, `warning`, `info`, `verbose`, `debug`.

--private-data *value*

Specifies the private data to insert in the *CA_descriptor* in the PMT. The value must be a suite of hexadecimal digits.

--stream-id *value*

Specifies the DVB SimulCrypt *ECM_stream_id* for the ECMG (default: 1).

-s *value*
--super-cas-id *value*

Specify the DVB SimulCrypt *Super_CAS_Id*. This is required when `--ecmg` is specified.

Transport stream scrambling options

--aes-cbc

Use AES-CBC scrambling instead of DVB-CSA2 (the default).

The control words are 16-byte long instead of 8-byte. The residue is left clear. Specify a fixed initialization vector using the `--iv` option.

Note that this is a non-standard TS scrambling mode. The only standard AES-based scrambling modes are ATIS-IDSA and DVB-CISSA (DVB-CISSA is the same as AES-CBC with a DVB-defined IV).

A *scrambling_descriptor* is automatically added to the PMT of the service to indicate the use of AES-CBC scrambling. Since there is no standard value for AES-CBC, the user-defined *scrambling_mode* value `0xF0` is used.

--aes-ctr

Use AES-CTR scrambling instead of DVB-CSA2 (the default).

The control words are 16-byte long instead of 8-byte. The residue is included in the scrambling. Specify a fixed initialization vector using the `--iv` option. See the option `--ctr-counter-bits` for the size of the counter part in the IV.

Note that this is a non-standard TS scrambling mode. The only standard AES-based scrambling modes are ATIS-IDSA and DVB-CISSA.

A *scrambling_descriptor* is automatically added to the PMT of the service to indicate the use of AES-CTR scrambling. Since there is no standard value for AES-CTR, the user-defined *scrambling_mode* value `0xF1` is used.

--atis-idsa

Use ATIS-IDSA scrambling (ATIS-0800006) instead of DVB-CSA2 (the default).

The control words are 16-byte long instead of 8-byte.

A *scrambling_descriptor* is automatically added to the PMT of the service to indicate the use of ATIS-IDSA scrambling.

--ctr-counter-bits *value*

With `--aes-ctr`, specifies the size in bits of the counter part.

In the initialization vector, the fixed nounce part uses the first 128 - *N* bits and the counter part uses the last *N* bits.

By default, the counter part uses the second half of the IV (64 bits).

-c *value*
--cw *value*

Specifies a fixed and constant control word (no crypto-period scheduling, no ECM insertion). The value must be a string of 16 hexadecimal digits (32 digits with `--atis-idsa` or `--dvb-cissa`).

When using this option, no Conditional Access System is used, meaning that no ECM or ECM PID is generated, no ECMG is allowed, all DVB SimulCrypt parameters are ignored and no CA_descriptor is inserted in the PMT.

--dvb-cissa

Use DVB-CISSA descrambling (see [16]) instead of DVB-CSA2 (the default).

The control words are 16-byte long instead of 8-byte.

A *scrambling_descriptor* is automatically added to the PMT of the service to indicate the use of DVB-CISSA scrambling.

--dvb-csa2

Use DVB-CSA2 scrambling. This is the default.

-f *name*
--cw-file *name*

Specifies a text file containing the list of control words to apply. Each line of the file must contain exactly 16 hexadecimal digits (32 digits with `--atis-idsa` or `--dvb-cissa`).

The next control word is used each time a new crypto-period is started. At the end of the list of control words, restart with the first one.

As with option `--cw`, no Conditional Access System is used, meaning that no ECM or ECM PID is generated, no ECMG is allowed, all DVB SimulCrypt parameters are ignored and no CA_descriptor is inserted in the PMT.

--iv *value*

With `--aes-cbc` or `--aes-ctr`, specifies a fixed initialization vector for all TS packets.

The value must be a string of 32 hexadecimal digits. The default IV is all zeroes.

-n
--no-entropy-reduction

With DVB-CSA2, do not perform control word entropy reduction to 48 bits, keep full 64-bit control words. This option is ignored with other scrambling modes.

--output-cw-file *name*

Specifies a text file to create with all control words. Each line of the file will contain a control word with 16 or 32 hexadecimal digits, depending on the scrambling algorithm. Each time a new control word is used to scramble packets, it is logged in the file.

This option is specifically useful when the control words are dynamically and randomly generated for insertion into ECM's. The created file can be used later to perform a direct descrambling test using the option `--cw-file` of the plugin *descrambler*.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

sdt

Perform Various Transformations on an SDT

This plugin performs various transformations on an SDT, either the SDT Actual or some specific SDT Other. The other SDT's, if present, are left unchanged.

Usage

```
tsp -P sdt [options]
```

Options

--bitrate *value*

Specifies the bitrate in bits / second of the PID containing the SDT if a new one is created.

The default is 3,000 b/s.

--cleanup-private-descriptors

Remove all private descriptors without preceding *private_data_specifier_descriptor*.

-c
--create

Create a new empty SDT if none was received after one second.

This is equivalent to `--create-after 1000`.

--create-after *milliseconds*

Create a new empty SDT if none was received after the specified number of milliseconds. If an actual SDT is received later, it will be used as the base for transformations instead of the empty one.

--eit-pf *value*

Specify a new *EIT_present_following_flag* value (0 or 1) for the added or modified service. For new services, the default is 0.

--eit-schedule *value*

Specify a new *EIT_schedule_flag* value (0 or 1) for the added or modified service. For new services, the default is 0.

-f *value*
--free-ca-mode *value*

Specify a new *free_CA_mode* value (0 or 1) for the added or modified service. For new services, the default is 0.

-i
--increment-version

Increment the version number of the SDT.

--inter-packet *value*

When a new SDT is created and `--bitrate` is not present, this option specifies the packet interval for the SDT PID, that is to say the number of TS packets in the transport between two packets of the PID.

Use instead of `--bitrate` if the global bitrate of the TS cannot be determined.

-n *value*
--name *value*

Specify a new service name for the added or modified service. For new services, the default is an empty string.

-v *value*
--new-version *value*

Specify a new value for the version of the SDT.

--original-network-id *id*

Modify the original network id in the SDT with the specified value.

-o *id*
--other *id*

Modify the SDT Other with the specified TS id. By default, modify the SDT Actual.

-p *value*
--provider *value*

Specify a new provider name for the added or modified service. For new services, the default is an empty string.

--remove-service *sid*

Remove the specified service-id from the SDT. Several `--remove-service` options may be specified to remove several services.

-r *value*
--running-status *value*

Specify a new *running_status* value (0 to 7) for the added or modified service. For new services, the default is 4 ("*running*").

-s *value*
--service-id *value*

Add a new service or modify the existing service with the specified service-id.

--ts-id *id*

Modify the transport stream id in the SDT with the specified value.

-t *value*
--type *value*

Specify a new service type for the added or modified service. For new services, the default is `0x01` ("*digital television service*").

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

sections

Remove or Merge Sections from Various PID's

This plugin extracts sections from one or more PID's and merges them inside an output PID.

Various filtering options can be used to selectively remove sections.

Usage

```
tsp -P sections [options]
```

Options

-e *id1*[*-id2*]
--etid-remove *id1*[*-id2*]

Remove all sections with the corresponding *extended table id* values. The value is a combination of the table id and the table id extension.

For example, the option `-e 0x4A1234` removes all BAT sections (table id `0x4A`) for bouquet id `0x1234` (table id extension).

Several options `--etid-remove` can be specified.

-n
--null-pid-reuse

With this option, null packets can be replaced by packets for the output PID.

By default, only packets from input PID's are replaced by output packets. This option may need to be used when `--stuffing` is specified and the input PID's contained packed sections. In that case, the output payload can be larger than the input and additional packets must be used.

-o *value*
--output-pid *value*

Specify the output PID. By default, the first input PID on the command line is used as output PID.

If the output PID is different from all input PID's and this output PID already exists in the transport stream, an error is generated.

-p *pid1*[*-pid2*]
--pid *pid1*[*-pid2*]

Specify input PID's. More than one input PID can be specified. All sections from all input PID's are merged into the output PID. At least one input PID must be specified.

-s
--stuffing

Insert stuffing at end of each section, up to the next TS packet boundary. By default, sections are packed and start in the middle of a TS packet, after the previous section. Note, however, that section headers are never scattered over a packet boundary.

-t *id1*[*-id2*]
--tid-remove *id1*[*-id2*]

Remove all sections with the corresponding table ids.

Several options `--tid-remove` can be specified.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[*-label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

sifilter

Extract PSI/SI PID's

This plugin filters PID's containing the specified PSI/SI. Other PID's are removed.

Extracting PSI/SI on predefined PID's (such as PAT or SDT) can also be performed using the plugin `filter --pid`. For these types of PSI/SI, the plugin `sifilter` is simply more user-friendly (`sifilter --sdt` instead of `filter --pid 0x0011`). But the plugin `sifilter` can also detect PSI/SI on non-predefined PID's (such as PMT, ECM or EMM). It can also filter CA-related SI according to the CA System Id or CA Operator (a vendor-dependent concept).

If you want to extract the PMT or ECM for one particular service, use the plugin `zap` before `sifilter` in the plugin chain.

Usage

```
tsp -P sifilter [options]
```

Options

--bat

Extract PID `0x0011` (SDT/BAT). Same as `--sdt`.

--cat

Extract PID `0x0001` (CAT).

--eit

Extract PID `0x0012` (EIT).

--nit

Extract PID `0x0010` (NIT).

--pat

Extract PID `0x0000` (PAT).

-p
--pmt

Extract all PMT PID's.

--rst

Extract PID `0x0013` (RST).

--sdt

Extract PID `0x0011` (SDT/BAT). Same as `--bat`.

-s
--stuffing

Replace excluded packets with stuffing (null packets) instead of removing them. Useful to preserve bitrate.

--tdt

Extract PID `0x0014` (TDT/TOT). Same as `--tot`.

--tot

Extract PID `0x0014` (TDT/TOT). Same as `--tdt`.

--tsdt

Extract PID `0x0002` (TSDT).

CAS selection options

--cas *value*

With options --ecm or --emm, select only ECM or EMM for the specified CA system id value. Equivalent to --min-cas value --max-cas value.

--ecm

Extract PID's containing ECM.

--emm

Extract PID's containing EMM.

--max-cas *value*

With options --ecm or --emm, select only ECM or EMM for the CA system id values in the range --min-cas to --max-cas.

--mediaguard

Equivalent to --min-cas 0x0100 --max-cas 0x01FF.

--min-cas *value*

With options --ecm or --emm, select only ECM or EMM for the CA system id values in the range --min-cas to --max-cas.

--nagravision

Equivalent to --min-cas 0x1800 --max-cas 0x18FF.

--operator *value*

With option --cas, select only ECM or EMM for the specified CAS operator. The "*CAS operator*" is a non-standard vendor-dependent concept and is recognized for some CAS only.

--safeaccess

Equivalent to --cas 0x4ADC.

--viaccess

Equivalent to --min-cas 0x0500 --max-cas 0x05FF.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several --only-label options may be specified.

skip

Skip Leading Packets in a TS

The plugin skips leading TS packets of a stream. The specified number of initial TS packets are dropped and not transmitted to the next plugin in the chain. After that, all packets are transparently passed.

Usage

```
tsp -P skip [options] count
```

Parameter

Number of leading TS packets to skip.

Options

-s
--stuffing

Replace excluded leading packets with stuffing (null packets) instead of removing them.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

slice

Pass or Drop Packets Based on Packet Numbers

This plugin passes or drops packets based on packet numbers or relative transport stream time. It can be used to extract selected portions of a TS and group them into one single output.

Usage

```
tsp -P slice [options]
```

Options

-d *value*
--drop *value*

All packets are dropped after the specified packet number. Several `--drop` options may be specified.

-i
--ignore-pcr

When `--seconds` or `--milli-seconds` is used, do not use PCR's to compute time values. Only rely on bitrate as determined by previous plugins in the chain.

-m
--milli-seconds

With options `--drop`, `--null`, `--pass` and `--stop`, interpret the integer values as milli-seconds from the beginning, not as packet numbers. Time is measured based on bitrate and packet count, not on real time.

-n *value*
--null *value*

All packets are replaced by null packets after the specified packet number. Several `--null` options may be specified.

-p *value*
--pass *value*

All packets are passed unmodified after the specified packet number. Several `--pass` options may be specified. This is the default for the initial packets.

--seconds

With options `--drop`, `--null`, `--pass` and `--stop`, interpret the integer values as seconds from the beginning, not as packet numbers. Time is measured based on bitrate and packet count, not on real time.

-s *value*
--stop *value*

Packet transmission stops after the specified packet number and *tsp* terminates.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[*-label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

spliceinject

Inject SCTE 35 splice commands in a transport stream

This plugin injects splice commands as *splice information sections*, as defined by the SCTE 35 standard [17]. All forms of splice information sections can be injected. The sections shall be provided by some external equipment, in real time. The format of the sections can be binary or XML (see section 2.2). All sections or tables shall be *splice information sections* (table id `0xFC`).

Injection principles

The whole point about splice information is synchronization with video. There are roughly two classes of splice events:

- Non-immediate *splice_insert()* commands. These commands contain a specific PTS value for the event. This PTS refers to a time stamp in the video and audio PID's of the service.
- Everything else.

Any splice command in the "everything else" category is injected as soon as possible after reception.

A non-immediate *splice_insert()* command is injected a specific number of times (2 by default) within a short period of time (2 seconds by default) preceding the specified PTS timestamp in the video stream.

When such a command is received in the plugin, the PTS of the event is compared with the current (or latest) PTS in the service. If the command is late and the PTS of the event is already in the past, the command is dropped. Otherwise, the command is placed in a waiting queue until the event time minus some predefined duration (see option `--start-delay`). At this time, the command is sent for the first time. It is later re-sent zero or more times. When the event time occurs, the command is no longer needed and is dropped.

Providing splice information tables

There are two possible mechanisms to provide the sections: files or UDP. The two options may be used simultaneously.

Files shall be specified as one single specification with optional wildcards. Example:

```
tsp ... -P spliceinject --files '/path/to/dir/splice*.xml' ...
```

All files named *splice*.xml* which are copied or updated into this directory are automatically loaded and injected. It is possible to automatically delete all files after being loaded.

UDP datagrams shall contain exactly one XML document or several binary sections. The XML document may contain several tables. The sections are injected upon reception. UDP reception is enabled by specifying a local port number. Example, listening on UDP port number 4444:

```
tsp ... -P spliceinject --udp 4444 ...
```

Using UDP usually provides a better reactivity than files. UDP messages are processed immediately after reception while files are detected on polling sequences only.

On a usability standpoint, remember that the *bash*[6] shell provides an easy way to send data or a file in an UDP message. So, sending a file though UDP is not more difficult than copying it to a directory. The following first command sends an XML file as one single UDP message on port 4444 to system 127.0.0.1 (the local host). The second command illustrates the file option.

```
cat splice_12.xml >/dev/udp/127.0.0.1/4444
cp splice_12.xml /path/to/dir
```

Usage

```
tsp -P spliceinject [options]
```

[6] This is a feature of *bash*, not a Linux feature. It is available on all platforms, including macOS or Cygwin.

General options

--inject-count *value*

For non-immediate *splice_insert()* commands, specifies the number of times the same splice information section is injected. The default is 2. Other splice commands are injected once only.

--inject-interval *value*

For non-immediate *splice_insert()* commands, specifies the interval in milliseconds between two insertions of the same splice information section. The default is 800 ms.

--pcr-pid *value*

Specifies the PID carrying the PCR reference clock. By default, use the PCR PID as declared in the PMT of the service.

-p *value*
--pid *value*

Specifies the PID for the injection of the splice information tables. By default, the injection of splice commands is done in the component of the service with a stream type equal to `0x86` in the PMT, as specified by SCTE 35 standard.

--pts-pid *value*

Specifies the PID carrying PTS reference clock. By default, use the video PID as declared in the PMT of the service.

--queue-size *value*

Specifies the maximum number of sections in the internal queue, sections which are received from files or UDP but not yet inserted into the TS. The default is 100.

-s *value*
--service *value*

Specifies the service for the insertion of the splice information tables. If the argument is an integer value (either decimal or hexadecimal), it is interpreted as a service id. Otherwise, it is interpreted as a service name, as specified in the SDT. The name is not case sensitive and blanks are ignored.

If no service is specified, the options `--pid` and `--pts-pid` must be specified (`--pcr-pid` is optional).

--start-delay *value*

For non-immediate *splice_insert()* commands, start to insert the first section this number of milliseconds before the specified splice PTS value. The default is 2000 ms.

-w
--wait-first-batch

When this option is specified, the start of the plugin is suspended until the first batch of splice commands is loaded and queued. Without this option, the input files or messages are loaded and queued asynchronously.

This option is typically useful when inserting splice commands from an XML file into a transport stream file. Since files are read much faster than the normal playout speed, it is possible that the splice points are already passed in the transport stream processing when the XML file is loaded. With this option, we have the guarantee that the XML file is loaded before the transport stream processing starts.

On the other hand, this option should not be used on live transport streams. In that case, the transport stream processing must be allowed to start without splice information tables to inject. These tables may be sent much later.

File input options

-d
--delete-files

Specifies that the files should be deleted after being loaded. By default, the files are left unmodified after being loaded. When a loaded file is modified later, it is reloaded and re-injected.

-f '*file-wildcard*'
--files '*file-wildcard*'

A file specification with optional wildcards indicating which files should be polled. When such a file is created or updated, it is loaded and its content is interpreted as binary or XML tables. All tables shall be splice information tables.

--max-file-size *value*

Files larger than the specified size are ignored. This avoids loading large spurious files which could clutter memory. The default is 2048 bytes.

--min-stable-delay *value*

A file size needs to be stable during that duration, in milliseconds, for the file to be reported as added or modified. This prevents too frequent poll notifications when a file is being written and his size modified at each poll. The default is 500 ms.

--poll-interval *value*

Specifies the interval in milliseconds between two poll operations. The default is 500 ms.

UDP input options

--buffer-size *value*

Specifies the UDP socket receive buffer size (socket option).

--no-reuse-port

Disable the reuse port socket option. Do not use unless completely necessary.

-r
--reuse-port

Set the reuse port socket option. This is now enabled by default, the option is present for legacy only.

-u [*address*:]*port*
--udp [*address*:]*port*

Specifies the local UDP port on which the plugin listens for incoming binary or XML splice information tables. When present, the optional address shall specify a local IP address or host name (by default, the plugin accepts connections on any local IP interface).

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

stuffanalyze

Analyze the level of stuffing in sections

This plugin analyzes the level of "stuffing" in sections in a list of selected PID's. A section is considered as "stuffing" when its payload is larger than 2 bytes and filled with the same byte value (all 0x00 or all 0xFF for instance).

The PID's to analyze can be selected manually or using CAS criteria.

Usage

```
tsp -P stuffanalyze [options]
```

Options

-o *filename*
--output-file *filename*

Specify the output text file for the analysis result. By default, use the standard output.

Warning: if you do not specify this option, be sure to redirect the output plugin to something different from the default. Otherwise, the text output of the analysis will be mixed with the binary output of the TS packets!

-p *pid1*[*-pid2*]
--pid *pid1*[*-pid2*]

Analyze all sections from these PID values. Several `-p` or `--pid` options may be specified.

CAS selection options

--cas *value*

With options `--ecm` or `--emm`, select only ECM or EMM for the specified CA system id value. Equivalent to `--min-cas value --max-cas value`.

--ecm

Extract PID's containing ECM.

--emm

Extract PID's containing EMM.

--max-cas *value*

With options `--ecm` or `--emm`, select only ECM or EMM for the CA system id values in the range `--min-cas` to `--max-cas`.

--mediaguard

Equivalent to `--min-cas 0x0100 --max-cas 0x01FF`.

--min-cas *value*

With options `--ecm` or `--emm`, select only ECM or EMM for the CA system id values in the range `--min-cas` to `--max-cas`.

--nagravision

Equivalent to `--min-cas 0x1800 --max-cas 0x18FF`.

--operator *value*

With option `--cas`, select only ECM or EMM for the specified CAS operator. The "*CAS operator*" is a non-standard vendor-dependent concept and is recognized for some CAS only.

--safeaccess

Equivalent to `--cas 0x4ADC`.

--viaccess

Equivalent to `--min-cas 0x0500 --max-cas 0x05FF`.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

`--help`

Display this help text.

`--only-label` *`label1`*`[-`*`label2`*`]`

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

svremove

Remove a Service

This plugin removes a service from the transport stream. The PAT, SDT Actual, NIT Actual and BAT are modified. The PMT and all components, including ECM streams, of the removed service are either removed or replaced by stuffing.

Usage

```
tsp -P svremove [options] service
```

Parameter

Specifies the service to remove. If the argument is an integer value (either decimal or hexadecimal), it is interpreted as a service id. Otherwise, it is interpreted as a service name, as specified in the SDT. The name is not case sensitive and blanks are ignored. If the input TS does not contain an SDT, use a service id.

Options

-a
--ignore-absent

Ignore service if not present in the transport stream. By default, *tsp* fails if the service is not found.

-b
--ignore-bat

Do not modify the BAT.

-e
--ignore-eit

Do not remove the EIT's for this service.

-n
--ignore-nit

Do not modify the NIT.

-s
--stuffing

Replace excluded packets with stuffing (null packets) instead of removing them. Useful to preserve bitrate.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

svrename

Rename a Service

This plugin renames a service. It assigns a new service name and/or a new service id.

The PAT, PMT of the service, SDT Actual, NIT Actual and BAT are modified.

The service id is modified in the PAT, PMT and SDT Actual. It is modified in the *service_list_descriptor* and *logical_channel_ number_descriptor* (EACEM/EICTA private descriptor) of the NIT Actual and the BAT. The service name is modified in the SDT Actual.

Usage

```
tsp -P svrename [options] service
```

Parameter

Specifies the service to rename. If the argument is an integer value (either decimal or hexadecimal), it is interpreted as a service id. Otherwise, it is interpreted as a service name, as specified in the SDT. The name is not case sensitive and blanks are ignored. If the input TS does not contain an SDT, use a service id.

Options

-f *value*
--free-ca-mode *value*

Specify a new *free_CA_mode* to set in the SDT (0 or 1).

-i *value*
--id *value*

Specify a new service id value.

--ignore-bat

Do not modify the BAT.

--ignore-eit

Do not modify the EIT's for this service.

--ignore-nit

Do not modify the NIT.

-l *value*
--lcn *value*

Specify a new logical channel number (LCN).

-n *name*
--name *name*

Specify a new service name.

-p *name*
--provider *name*

Specify a new provider name.

-r *value*
--running-status *value*

Specify a new *running_status* to set in the SDT (0 to 7).

-t *value*
--type *value*

Specify a new service type.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

t2mi

Extract T2-MI (DVB-T2 Modulator Interface) packets

This plugin extracts (or simply logs) T2-MI packets. T2-MI is the DVB-T2 Modulator Interface. This is a protocol which encapsulates DVT-T2 modulator commands (including TS packets) into one PID of a transport stream. See [10] and [11] for more details.

This plugin selects one PID from the input transport stream. This PID shall contain an encapsulated T2-MI stream. This plugin extracts the embedded transport stream from one PLP (Physical Layer Pipe) of the original PID. By default, the input transport stream is completely replaced with the extracted stream. Using the option `--output-file`, the extracted encapsulated transport stream is saved in a file and, in that case, the input transport stream is passed unmodified.

Alternatively, the `t2mi` plugin can simply log all T2-MI packets without replacing the input transport stream. This is typically useful for debug only.

Warning: This plugin is currently experimental and has some limitations. DVB-T2 is complex and this complexity has an impact on the encapsulation of TS packets inside a T2-MI stream. This plugin may not work with all mode or stream adaptations (see [11]). If you encounter problems with some T2-MI streams, please report an issue (see [33]) and provide a sample transport stream which exhibits the problem.

Usage

```
tsp -P t2mi [options]
```

Options

-a
--append

With `--output-file`, if the file already exists, append to the end of the file. By default, existing files are overwritten.

-e
--extract

Extract encapsulated TS packets from one PLP of a T2-MI stream. The transport stream is completely replaced by the extracted stream. This is the default if neither `--extract` nor `--log` nor `--identify` is specified.

-i
--identify

Identify all T2-MI PID's and PLP's.

If `--pid` is specified, only identify PLP's in this PID. If `--pid` is not specified, identify all PID's carrying T2-MI and their PLP's (require a fully compliant T2-MI signalization).

-k
--keep

With `--output-file`, keep existing file (abort if the specified file already exists). By default, existing files are overwritten.

-l
--log

Log all T2-MI packets using one single summary line per packet. This is typically useful for debug only.

If `--log` is specified without `--extract`, the input transport stream is passed unmodified. If both `--extract` and `--log` are specified, the T2-MI packets are logged and the encapsulated stream replaces the input stream.

-o *filename*
--output-file *filename*

Specify that the extracted stream is saved in this file. In that case, the main transport stream is passed unchanged to the next plugin.

-p *value*
--pid *value*

Specify the PID carrying the T2-MI encapsulated stream. By default, the plugin automatically locates and uses the first component with a *T2MI_descriptor* in the PMT of its service.

--plp *value*

Specify the PLP (Physical Layer Pipe) to extract from the T2-MI encapsulation. By default, use the first PLP which is found. This option is ignored if `--extract` is not used.

To determine which PID's carry T2-MI streams and what are the PLP's inside each stream, use the command `tsanalyze` or the plugin `analyse`.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

tables

Collect MPEG Tables

This plugin collects MPEG tables from a transport stream. The tables can be displayed or saved in a human readable format, saved in binary or XML files or sent over UDP/IP to some collecting server. It is equivalent to the *tstables* utility. Actually, the following two commands produce the same result:

```
tstables options filename
tsp -I file filename -P tables options -O drop
```

Usage

```
tsp -P tables [options]
```

Options

The plugin accepts exactly the same options as the *tstables* utility.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

teletext

Extract Teletext subtitles in SRT format

This plugin extracts a Teletext subtitle stream from a service and exports it in SRT format, also known as "SubRip" format. SRT is a text format which can be manipulated by many video processing tools.

Teletext subtitles are contained in a PID which is signalled in the PMT of the service. Unlike DVB subtitles, a single Teletext PID can contain more than one subtitle stream. Typically, one PID can contain a multiplex of the standard and "for hard of hearing" subtitles. Each subtitle stream is defined by its *Teletext Page* number. All page numbers inside a single Teletext PID are normally listed in a Teletext descriptor in the PMT of the service.

Usage

```
tsp -P teletext [options]
```

Options

-c
--colors

Add font color tags in the subtitles. By default, no color is specified.

-l *name*
--language *name*

Specify the language of the subtitles to select. This option is useful only with `--service`, when the PMT of the service declares Teletext subtitles in different languages.

-m *value*
--max-frames *value*

Specifies the maximum number of Teletext frames to extract. The processing is then stopped.

By default, all frames are extracted.

-o *filename*
--output-file *filename*

Specify the SRT output file name. This is a text file. By default, the SRT subtitles are displayed on the standard output.

--page *value*

Specify the Teletext page to extract. This option is useful only when the Teletext PID contains several pages. By default, the first Teletext frame defines the page to use.

-p *value*
--pid *value*

Specify the PID carrying Teletext subtitles.

Alternatively, if the Teletext PID is properly signalled in the PMT of its service, the option `--service` can be used instead.

-s *value*
--service *value*

Specify the service with Teletext subtitles. If the argument is an integer value (either decimal or hexadecimal), it is interpreted as a service id. Otherwise, it is interpreted as a service name, as specified in the SDT. The name is not case sensitive and blanks are ignored.

The first *teletext_descriptor* in the PMT of the service is used to identify the PID carrying Teletext subtitles.

If neither `--service` nor `--pid` is specified, the first service in the PAT is used.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

time

Schedule Packets Pass or Drop

This plugin schedules in time the processing of packets (drop packets, pass packets or replace them by null packets). This plugin may be used to schedule the recording of a program at a specified time, for instance.

Usage

```
tsp -P time [options]
```

Options

-d *time*
--drop *time*

All packets are dropped after the specified time. Several `--drop` options may be specified.

-n *time*
--null *time*

All packets are replaced by null packets after the specified time. Several `--null` options may be specified.

-p *time*
--pass *time*

All packets are passed unmodified after the specified time. Several `--pass` options may be specified.

-r
--relative

All time values are interpreted as a number of seconds relative to the *tsp* start time. By default, all time values are interpreted as an absolute time in the format "*year/month/day:hour:minute:second*". Option `--relative` is incompatible with `--tdt` or `--utc`.

-s *time*
--stop *time*

Packet transmission stops after the specified time and tsp terminates.

-t
--tdt

Use the Time & Date Table (TDT) from the transport stream as time reference instead of the system clock. Since the TDT contains UTC time, all time values in the command line must be UTC also.

-u
--utc

Specifies that all time values in the command line are in UTC. By default, the time values are interpreted as system local time.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[*-label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

Specifying time values

A time value must be in the format "*year/month/day:hour:minute:second*" (unless `--relative` is specified, in which case it is a number of seconds). An empty value ("") means "*from the beginning*", that is to say when *tsp* starts. By default, packets are passed when *tsp* starts.

timeref

Update TDT and TOT with a new time reference

This plugin updates all TDT and TOT (and optionally EIT) in the transport stream according to a new time reference. This new reference can be completely new or an offset from the original TS.

Usage

```
tsp -P timeref [options]
```

Options

-a *seconds*
--add *seconds*

Add the specified number of seconds to all UTC time. Specify a negative value to make the time reference go backward.

--eit

Update events start time in EIT's. By default, EIT's are not modified.

When `--add` is used, the specified offset is applied to all events start time.

When `--start` is used, EIT's are dropped until the first TDT or TOT is encountered. Then, the difference between the first TDT or TOT time and the new time reference at this point is applied.

--notdt

Do not update TDT.

--notot

Do not update TOT.

-s *time*
--start *time*

Specify a new UTC date & time reference for the first packet in the stream. Then, the time reference is updated according to the number of packets and the bitrate.

A time value must be in the format "*year/month/day:hour:minute:second*".

The predefined name “system” can be used to specify the current UTC time from the system clock (use `--start system`).

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[*-label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

trigger

Trigger actions on selected labeled TS packets

This plugin triggers an action (running an external command, sending an UDP packet) each time a TS packet is marked with a given label. Labels are typically set on packets by a previous plugin in the chain, such as the *filter* plugin.

Usage

```
tsp -P trigger [options]
```

Options

-a
--all-labels

All labels from options `--label` shall be set on a packet to be selected (logical '*and*').

By default, a packet is selected if any label is set (logical '*or*').

-e '*command*'
--execute '*command*'

Run the specified command when the current packet triggers the actions.

-l *label1*[-*label2*]
--label *label1*[-*label2*]

Trigger the actions on packets with any of the specified labels. Labels should have typically be set by a previous plugin in the chain.

By default, without option `--label`, the actions are triggered on all packets in the stream.

Several `--label` options may be specified.

Note that the option `--label` is different from the generic option `--only-label`. The generic option `--only-label` acts at *tsp* level and controls which packets are passed to the plugin. All other packets are directly passed to the next plugin without going through this plugin. The option `--label`, on the other hand, is specific to the *trigger* plugin and selects packets with specific labels among the packets which are passed to this plugin.

--local-address *address*

With `--udp`, when the destination is a multicast address, specify the IP address of the outgoing local interface. It can be also a host name that translates to a local address.

--min-inter-packet *count*

Specify the minimum number of packets between two triggered actions.

Actions which should be triggered in the meantime are ignored.

--min-inter-time *milliseconds*

Specify the minimum time, in milliseconds, between two triggered actions.

Actions which should be triggered in the meantime are ignored.

--ttl *value*

With `--udp`, specifies the TTL (Time-To-Live) socket option. The actual option is either "Unicast TTL" or "Multicast TTL", depending on the destination address. Remember that the default Multicast TTL is 1 on most systems.

-u *address*:*port*
--udp *address*:*port*

Send a UDP/IP message to the specified destination when the current packet triggers the actions. The *address* specifies an IP address which can be either unicast or multicast. It can be also a host name that translates to an IP address. The *port* specifies the destination UDP port.

--udp-message *hexa-string*

With --udp, specifies the binary message to send as UDP datagram. The value must be a string of hexadecimal digits specifying any number of bytes.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several --only-label options may be specified.

tsrename

Rename a transport stream

This plugin renames the transport stream. It assigns a new transport stream id and/or a original network id.

The PAT, SDT Actual, NIT Actual and BAT are modified.

Usage

```
tsp -P tsrename [options]
```

Options

-a
--add

Equivalent to `--add-bat --add-nit`.

--add-bat

Add a new entry for the renamed TS in the BAT and keep the previous entry. By default, the TS entry is renamed. Note that if no previous entry existed for this TS in the BAT, none is created.

--add-nit

Add a new entry for the renamed TS in the NIT and keep the previous entry. By default, the TS entry is renamed. Note that if no previous entry existed for this TS in the NIT, none is created.

--ignore-bat

Do not modify the BAT.

--ignore-eit

Do not modify the EIT's for this transport stream.

--ignore-nit

Do not modify the NIT.

-o *value*
--original-network-id *value*

Modify the original network id. By default, it is unchanged.

-t *value*
--ts-id *value*

Modify the transport stream id. By default, it is unchanged.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

until

Pass Packets Until Specified Condition

This plugin passes all TS packets to the next plugin in the chain, until one of the specified conditions is met. At this point, the plugin simulates an "end of input stream" and all subsequent packets are dropped. The previous plugins in the chain are notified to stop. When the next plugins in the chain finish the processing of the passed packet, tsp terminates.

Usage

```
tsp -P until [options]
```

Options

-b *value*
--bytes *value*

Stop after processing the specified number of bytes.

-e
--exclude-last

Exclude the last packet (the one which triggers the final condition).

-j
--joint-termination

When the final condition is triggered, perform a *joint termination* instead of unconditional termination. See the description of the `tsp` command for more details on *joint termination*.

-m *value*
--milli-seconds *value*

Stop the specified number of milli-seconds after receiving the first packet.

-n *value*
--null-sequence-count *value*

Stop when the specified number of sequences of consecutive null packets is encountered.

-p *value*
--packets *value*

Stop after the specified number of packets.

-s *value*
--seconds *value*

Stop the specified number of seconds after receiving the first packet.

-u *value*
--unit-start-count *value*

Stop when the specified number of packets containing a payload unit start indicator is encountered.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

zap

Zap on one service (create an SPTS)

This plugin "zaps" on one service: it produces a Single Program Transport Stream (SPTS) containing only the specified service. The PAT and SDT are modified in order to contain only the specified service. Unless specified otherwise (see the relevant options), the PMT and all elementary streams of the service are passed transparently. All other PID's in the transport streams are removed. If some elementary streams (audio, subtitles) must be removed from the service, the PMT is modified accordingly.

For ATSC transport streams, the service is extracted as with any MPEG-compliant transport stream but the PSIP signalization (PID `0x1FFB`) is not modified. The TVCT or CVCT still contains the description of all previous (and now removed) services in the transport stream.

Usage

```
tsp -P zap [options] service
```

Parameter

The parameter specifies the service to keep.

If the parameter is an integer value (either decimal or hexadecimal), it is interpreted as a service id. If its format is "*integer.integer*", it is interpreted as major and minor ids on ATSC streams. Otherwise, the parameter is interpreted as a service name, as specified in the SDT (DVB) or VCT (ATSC).

The name is not case sensitive and blanks are ignored.

If the input TS does not contain an SDT (DVB) or VCT (ATSC), use a service id.

Options

-a *name*
--audio *name*

Remove all audio components except the specified one. The name is a three-letters language code. By default, keep all audio components.

--audio-pid *value*

Remove all audio components except the specified audio PID. By default, keep all audio components.

This option and the `--audio` option are mutually exclusive.

-c
--cas

Keep Conditional Access System sections (CAT and EMM's). Remove them by default. Note that the ECM's for the specified service are always kept.

--eit

Keep EIT sections for the specified service. EIT sections for other services are removed. By default, all EIT's are removed.

-e
--no-ecm

Remove all ECM PID's. By default, keep all ECM PID's.

-n
--no-subtitles

Remove all subtitles. By default, keep all subtitles.

-p
--pes-only

Keep only the PES elementary streams (audio, video, subtitles). Remove all PSI/SI and CAS information.

-s
--stuffing

Replace excluded packets with stuffing (null packets) instead of removing them. Useful to preserve bitrate.

-t *name*
--subtitles *name*

Remove all subtitles except the specified one. The name is a three-letters language code. By default, keep all subtitles.

Generic packet processing plugin options

The following options are implicitly defined in all packet processing plugins.

--help

Display this help text.

--only-label *label1*[-*label2*]

Invoke this plugin only for packets with any of the specified labels. Other packets are transparently passed to the next plugin, without going through this one.

Several `--only-label` options may be specified.

5 Usage Examples

5.1 TSDuck Utilities

5.1.1 tsdektec examples

Listing all (`-a`) Dektec devices:

```
$ tsdektec -a
0: DTA-110 (DTA-110T Modulator with UHF Upconverter)
1: DTA-140 (DTA-140 DVB/ASI Input+Output)
$
```

Listing all (`-a`) Dektec devices in verbose format (`-v`):

```
$ tsdektec -av

DTAPI version: 4.1.1.108
PCI device driver: 2.2.0.124
USB device driver: unknown

* Device 0: DTA-110 (DTA-110T Modulator with UHF Upconverter)
  Physical ports: 1
  Channels: input: 0, output: 1
  Output 0: Port 1, Modulator, Failsafe, ATSC/VSB, DVB-T/DVB-H, DVB-C,
      QAM-B (USA), QAM-C (Japan), UHF
  Subsystem id: 0xD10A (DTA-110)
  Subsystem vendor id: 0x14B4
  Device id: 0x9056
  Vendor id: 0x10B5
  Serial number: 00000000F50268FF
  Firmware version: 4 (0x00000004)
  Firmware variant: 4 (0x00000004)
  PCI bus: 5, slot: 5
  Customer id: 301819
  Engineering change level: Rev 3
  Manufacture id: 03
  Production date: 2002.07
  Part number: DTA-110T
  Serial number: 4110575871
  Crystal stability: RF:1ppm;Sym:25ppm

* Device 1: DTA-140 (DTA-140 DVB/ASI Input+Output)
  Physical ports: 2
  Channels: input: 1, output: 1
  Input 0: Port 1, top socket, ASI/SDI, ASI
  Output 0: Port 2, ASI/SDI, ASI
  Subsystem id: 0xD128 (DTA-140)
  Subsystem vendor id: 0x14B4
  Device id: 0x9056
  Vendor id: 0x10B5
  Serial number: 00000000F6C458E8
  Firmware version: 2 (0x00000002)
  Firmware variant: 0 (0x00000000)
  PCI bus: 5, slot: 6
  Customer id: 301819
  Engineering change level: Rev 1A
  Manufacture id: 03
  Production date: 2003.05
  Part number: DTA-140
  Serial number: 4140062952
  Crystal stability: 10ppm

$
```

5.1.2 tslsdvb examples

Listing all DVB receiver devices on a Linux system with a dual-tuner Hauppauge Nova-T 500. Each tuner of the single PCI board is seen as one DVB receiver device:

```
$ tslsdvb
/dev/dvb/adapter0 (DiBcom 3000MC/P, DVB-T)
/dev/dvb/adapter1 (DiBcom 3000MC/P, DVB-T)
$
```

The DVB receiver device name is `/dev/dvb/adapter0` but it can also be specified using the option `--adapter` (`-a`) in all TSDuck commands: the options "`--device-name /dev/dvb/adapter1`" and "`-a 1`" are equivalent.

Listing all DVB receiver devices on a Windows system with one USB receiver:

```
C:\> tslsdvb
0: "Nova-T Stick DVB-T Tuner (Dev1 Path0)" (DVB-T)
C:\>
```

The DVB receiver device name is "`Nova-T Stick DVB-T Tuner (Dev1 Path0)`". This is the name of the DirectShow tuner filter supplied by the hardware vendor.

Listing all DVB receiver devices on a Windows system with two other USB receivers:

```
C:\> tslsdvb
0: "Cinergy T USB XE (MKII) Tuner" (DVB-T)
1: "PCTV DiBcom BDA Digital Tuner (Dev1 Path0)" (DVB-T)
C:\>
```

Listing all DVB receiver devices on a Linux system in verbose (`-v`) format. Note that the current modulation parameters are usually accessible on Linux systems only. On Windows systems, most tuner drivers do not return them and `tslsdvb` cannot display the characteristics of the current transponder.

```
$ tslsdvb -v

/dev/dvb/adapter0 (DiBcom 3000MC/P, DVB-T)

  Status: has signal, has carrier, has viterbi, has sync, has lock

  Bit error rate ............................... 0 (0%)
  Signal/noise ratio ........................... 0 (0%)
  Signal strength ........................... 39,586 (60%)
  Uncorrected blocks ........................... 0
  Frequencies:
    Current ............................ 562,000,000 Hz
    UHF channel ................................ 32
    Min ................................ 48,000,000 Hz
    Max ................................ 860,000,000 Hz
    Step ................................... 62,500 Hz
    Tolerance ................................... 0 Hz
  Spectral inversion ......................... auto
  Bandwidth ................................. 8-MHz
  FEC (high priority) ........................ 2/3
  FEC (low priority) ......................... 1/2
  Constellation ............................. 64-QAM
  Transmission mode ............................ 8K
  Guard interval ............................. 1/32
  Hierarchy ................................... none

  Capabilities: inversion auto, FEC 1/2, FEC 2/3, FEC 3/4, FEC 5/6, FEC 7/8,
    FEC auto, QPSK, 16-QAM, 64-QAM, QAM auto, transmission mode auto,
    guard interval auto, hierarchy auto, recover

/dev/dvb/adapter1 (DiBcom 3000MC/P, DVB-T)

  Status: has signal, has carrier, has viterbi, has sync, has lock
```

```
  Bit error rate ............................... 0 (0%)
  Signal/noise ratio ........................... 0 (0%)
  Signal strength ......................... 40,690 (62%)
  Uncorrected blocks ........................... 0
  Frequencies:
    Current ........................... 490,000,000 Hz
    UHF channel ................................ 23
    Min ............................... 48,000,000 Hz
    Max ............................... 860,000,000 Hz
    Step ................................. 62,500 Hz
    Tolerance ..................................... 0 Hz
  Spectral inversion ......................... auto
  Bandwidth ................................. 8-MHz
  FEC (high priority) ........................ 2/3
  FEC (low priority) ......................... 1/2
  Constellation ............................ 16-QAM
  Transmission mode ............................ 8K
  Guard interval ............................. 1/32
  Hierarchy ................................... none

  Capabilities: inversion auto, FEC 1/2, FEC 2/3, FEC 3/4, FEC 5/6, FEC 7/8,
    FEC auto, QPSK, 16-QAM, 64-QAM, QAM auto, transmission mode auto,
    guard interval auto, hierarchy auto, recover

$
```

5.1.3 tsscan examples

UHF-band scanning, including a global service list at end of network scanning:

```
$ tsscan -g
* UHF channel 21, offset +1 (474.166 MHz), strength: 59%
  Transport stream id: 2, 0x0002
* UHF channel 23, offset +1 (490.166 MHz), strength: 62%
  Transport stream id: 8, 0x0008
* UHF channel 24, offset +1 (498.166 MHz), strength: 62%
  Transport stream id: 4, 0x0004
* UHF channel 27, offset +1 (522.166 MHz), strength: 63%
  Transport stream id: 3, 0x0003
* UHF channel 32, offset +1 (562.166 MHz), strength: 61%
  Transport stream id: 6, 0x0006
* UHF channel 35, offset +1 (586.166 MHz), strength: 63%
  Transport stream id: 1, 0x0001

LCN Name             Provider ServId TSId   ONetId Type PMTPID
--- ---------------- -------- ------ ------ ------ ---- ------
  1 TF1              SMR6     0x0601 0x0006 0x20FA 0x01 0x0064
  2 France 2         GR1      0x0101 0x0001 0x20FA 0x01 0x006E
  3 France 3         GR1      0x0111 0x0001 0x20FA 0x01 0x00D2
  4 CANAL+           CNH      0x0301 0x0003 0x20FA 0x01 0x0500
  5 France 5         GR1      0x0104 0x0001 0x20FA 0x01 0x0136
  6 M6               MULTI4   0x0401 0x0004 0x20FA 0x01 0x006E
  7 ARTE             GR1      0x0105 0x0001 0x20FA 0x01 0x01FE
  8 Direct 8         NTN      0x0201 0x0002 0x20FA 0x01 0x0500
  9 W9               MULTI4   0x0402 0x0004 0x20FA 0x01 0x00D2
 10 TMC              SMR6     0x0606 0x0006 0x20FA 0x01 0x0258
 11 NT1              MULTI4   0x0403 0x0004 0x20FA 0x01 0x0136
 12 NRJ12            SMR6     0x0602 0x0006 0x20FA 0x01 0x00C8
 13 LCP              GR1      0x0106 0x0001 0x20FA 0x01 0x0262
 14 France 4         NTN      0x0207 0x0002 0x20FA 0x01 0x0506
 15 BFM TV           NTN      0x0203 0x0002 0x20FA 0x01 0x0502
 16 i>TELE           NTN      0x0204 0x0002 0x20FA 0x01 0x0503
 17 Virgin 17        NTN      0x0205 0x0002 0x20FA 0x01 0x0504
 18 Gulli            NTN      0x0206 0x0002 0x20FA 0x01 0x0505
 20 France Ô         GR1      0x0176 0x0001 0x20FA 0x01 0x02C6
 21 Canal 21         Multi-7  0x0802 0x0008 0x20FA 0x01 0x10E1
```

```
 22 IDF1             Multi-7  0x0803 0x0008 0x20FA 0x01 0x10E2
 23 NRJ Paris        Multi-7  0x0804 0x0008 0x20FA 0x01 0x10E3
 24 CAP 24           Multi-7  0x0805 0x0008 0x20FA 0x01 0x10E4
 30 TPS STAR         CNH      0x0306 0x0003 0x20FA 0x01 0x0505
 31 PARIS PREMIERE   MULTI4   0x0404 0x0004 0x20FA 0x01 0x019A
 32 CANAL+ SPORT     CNH      0x0303 0x0003 0x20FA 0x01 0x0502
 33 CANAL+ CINEMA    CNH      0x0302 0x0003 0x20FA 0x01 0x0501
 34 AB1              MULTI4   0x0406 0x0004 0x20FA 0x01 0x0262
 35 PLANETE          CNH      0x0304 0x0003 0x20FA 0x01 0x0503
 36 TF6              MULTI4   0x0405 0x0004 0x20FA 0x01 0x01FE
 37 CANAL J          CNH      0x0305 0x0003 0x20FA 0x01 0x0504
 38 LCI              SMR6     0x0603 0x0006 0x20FA 0x01 0x012C
 39 Eurosport France SMR6     0x0604 0x0006 0x20FA 0x01 0x0190
                              0x01FF 0x0001 0x20FA      0x03F2
                              0x02FF 0x0002 0x20FA      0x050A
                     CNH      0x03F0 0x0003 0x20FA 0x0C 0x050A
                     CNH      0x03F1 0x0003 0x20FA 0x0C 0x050B
                              0x04FF 0x0004 0x20FA 0x0C 0x03F2
$
```

UHF-band scanning, including modulation parameters information (usually unavailable on Windows, depending on the tuner driver):

```
$ tsscan -m
* UHF channel 21, offset +1 (474.166 MHz), strength: 59%
  Transport stream id: 2, 0x0002
  Carrier frequency: 474,166,666 Hz
  Constellation: 64-QAM
  HP streams FEC: 2/3
  LP streams FEC: 1/2
  Guard interval: 1/32
  Transmission mode: 8K
  Hierarchy: none
* UHF channel 23, offset +1 (490.166 MHz), strength: 62%
  Transport stream id: 8, 0x0008
  Carrier frequency: 490,166,666 Hz
  Constellation: 16-QAM
  HP streams FEC: 2/3
  LP streams FEC: 1/2
  Guard interval: 1/32
  Transmission mode: 8K
  Hierarchy: none
* UHF channel 24, offset +1 (498.166 MHz), strength: 62%
  Transport stream id: 4, 0x0004
  Carrier frequency: 498,166,666 Hz
  Constellation: 64-QAM
  HP streams FEC: 2/3
  LP streams FEC: 1/2
  Guard interval: 1/32
  Transmission mode: 8K
  Hierarchy: none
* UHF channel 27, offset +1 (522.166 MHz), strength: 63%
  Transport stream id: 3, 0x0003
  Carrier frequency: 522,166,666 Hz
  Constellation: 64-QAM
  HP streams FEC: 2/3
  LP streams FEC: 1/2
  Guard interval: 1/32
  Transmission mode: 8K
  Hierarchy: none
* UHF channel 32, offset +1 (562.166 MHz), strength: 61%
  Transport stream id: 6, 0x0006
  Carrier frequency: 562,166,666 Hz
  Constellation: 64-QAM
  HP streams FEC: 2/3
  LP streams FEC: 1/2
```

```
    Guard interval: 1/32
    Transmission mode: 8K
    Hierarchy: none
  * UHF channel 35, offset +1 (586.166 MHz), strength: 63%
    Transport stream id: 1, 0x0001
    Carrier frequency: 586,166,666 Hz
    Constellation: 64-QAM
    HP streams FEC: 3/4
    LP streams FEC: 1/2
    Guard interval: 1/8
    Transmission mode: 8K
    Hierarchy: none
  $
```

5.1.4 tssmartcard examples

Listing all smartcard readers in the system:

```
$ tssmartcard
OmniKey CardMan 3121 00 00
OmniKey CardMan 3121 01 00
OmniKey CardMan 3121 02 00
OmniKey CardMan 3121 03 00
$
```

Listing all smartcard readers in the system, in verbose (`-v`) format:

```
$ tssmartcard -v
OmniKey CardMan 3121 00 00: empty
OmniKey CardMan 3121 01 00: smartcard present
    ATR: 3B DE 18 00 40 11 90 28 43 29 4C 6F 67 69 77 61 79 73 AA 55
OmniKey CardMan 3121 02 00: empty
OmniKey CardMan 3121 03 00: smartcard present
    ATR: 3B DE 18 00 40 11 90 28 43 29 4C 6F 67 69 77 61 79 73 AA 55
$
```

Perform a warm (`-w`) reset on the second smartcard then list all readers in verbose format again: the smartcard now returns its “warm reset” ATR.

```
$ tssmartcard "OmniKey CardMan 3121 01 00" -w
$ tssmartcard -v
OmniKey CardMan 3121 00 00: empty
OmniKey CardMan 3121 01 00: smartcard present
    ATR: 3B D3 18 00 40 11 90 AA 55
OmniKey CardMan 3121 02 00: empty
OmniKey CardMan 3121 03 00: smartcard present
    ATR: 3B DE 18 00 40 11 90 28 43 29 4C 6F 67 69 77 61 79 73 AA 55
$
```

5.1.5 tsterinfo examples

Converting UHF channels to frequencies:

```
$ tsterinfo -u 21
Carrier Frequency: 474,000,000 Hz
$
$ tsterinfo -u 21 -o 1
Carrier Frequency: 474,166,666 Hz
$
$ tsterinfo -u 21 -o 1 -s
474166666
$
```

Converting frequencies to UHF channels:

```
$ tsterinfo -f 474166666
UHF channel: 21, offset: 1
$
```

```
$ tsterinfo -f 474166000
UHF channel: 21, offset: 1
Warning: exact frequency for channel 21, offset 1 is 474,166,666 Hz, differ by -666 Hz
$
```

Computing transport stream bitrate from OFDM modulation parameters:

```
$ tsterinfo -h 2/3 -g 1/32
Transport stream bitrate: 24,128,342 b/s
$
$ tsterinfo -h 2/3 -g 1/32 -c QPSK
Transport stream bitrate: 8,042,780 b/s
$
$ tsterinfo -h 2/3 -g 1/32 -c QPSK -s
8042780
$
```

Retrieving OFDM modulation parameters from the transport stream bitrate. Note that the second example gives two possible sets of parameters with the same bitrate difference.

```
$ tsterinfo -b 24128300
  Nominal bitrate .......... 24,128,342 b/s
  Bitrate difference ............... -42 b/s
  Bandwidth ...................... 8-MHz
  FEC (high priority) ............. 2/3
  Constellation ................. 64-QAM
  Guard interval .................. 1/32
$
$ tsterinfo -b 24882000
  Nominal bitrate .......... 24,882,352 b/s
  Bitrate difference ............... -352 b/s
  Bandwidth ...................... 8-MHz
  FEC (high priority) ............. 3/4
  Constellation ................. 64-QAM
  Guard interval .................. 1/8

  Nominal bitrate .......... 24,882,352 b/s
  Bitrate difference ............... -352 b/s
  Bandwidth ...................... 8-MHz
  FEC (high priority) ............. 5/6
  Constellation ................. 64-QAM
  Guard interval .................. 1/4
$
```

5.1.6 tshides examples

The command *tshides* lists the HiDes devices, typically cheap modulators. Since these devices are simple encapsulations around chips from ITE Technologies, using device drivers from ITE, they usually appear as ITE 950x, from the model name of the main chip in the HiDes device.

Using *tshides* on Windows:

```
C:\> tshides
0: "IT9507 TX Filter"

C:\> tshides -v
Found 1 HiDes device

Index ........... 0
Name ............ "IT9507 TX Filter"
Device .......... \\?\usb#vid_048d&pid_9507#ut100cv4201504240422#{fbf6f530-07b9-11d2-
a71e-0000f8004788}\{9963cc0e-ee70-11e0-ba8f-92d34824019b}
USB mode ........ 0x0200
Vendor id ....... 0x048D
Product id ...... 0x9507
Chip type ....... 0x9507
Device type ..... 11
```

```
Driver version .. 21.17.39.1
Link firmware ... 255.39.2.0
OFDM firmware ... 255.9.11.0
```

The option `--gain-range` is used to display the adjustable gain range for a given frequency and a given bandwidth. Sample usage on Windows, using the default values for frequency and bandwidth:

```
C:\> tshides --gain-range
Device: 0: "IT9507 TX Filter"
Frequency: 474,000,000 Hz
Bandwidth: 8-MHz
Min. gain: -52 dB
Max. gain: 6 dB
>
```

Using *tshides* on Linux with the same HiDes device. Notice the naming difference.

```
$ tshides
0: "usb-it950x0" (/dev/usb-it950x0)
$
$ tshides -v
Found 1 HiDes device

Index ........... 0
Name ............ "usb-it950x0"
Device .......... /dev/usb-it950x0
Chip type ....... 0x9507
Device type ..... 11
Driver version .. v16.11.10.1
API version ..... 1.3.20160929.0
Link firmware ... 255.39.2.0
OFDM firmware ... 255.9.11.0
Company ......... ITEtech
Hardware info ... Eagle DVBT

$
```

5.1.7 tsswitch examples

The following diagram illustrates a sample usage of the *tsswitch* command:

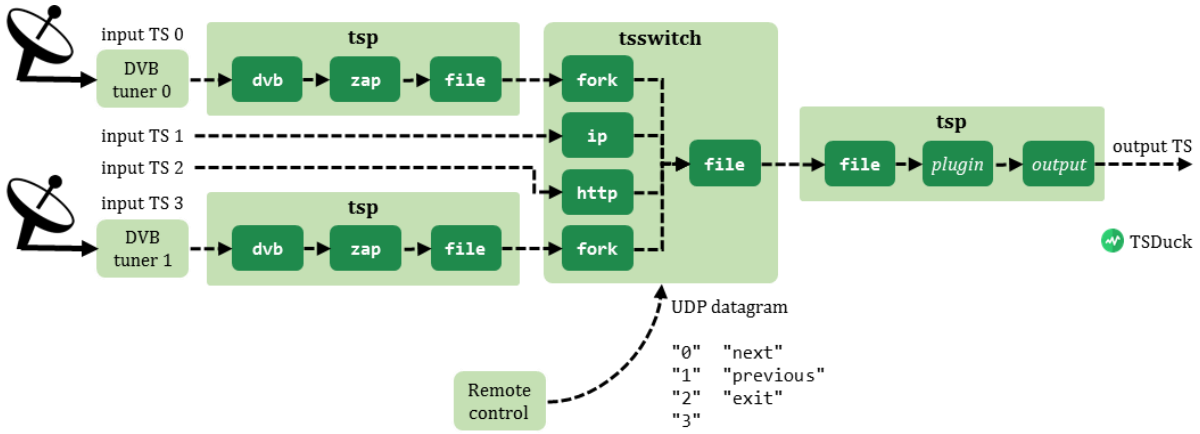

Figure 3: Sample input switching configuration

In the example above, four inputs are used. Each input contains an SPTS (single-program transport stream).

Two of these inputs are network streams already containing an SPTS. They can be directly received by an input plugin in *tsswitch* (the plugin *ip* is used to receive an UDP/IP multicast stream and the plugin *http* is used to receive an HTTP unicast stream).

The two other inputs are taken from broadcast transport streams which contain multiple services. The target service must be extracted before input to *tsswitch*. To achieve that, we run two *tsp* commands which extract the target services and we inject the output into an input of *tsswitch*.

The complete command skeleton is the following:

```
$ tsswitch --remote 4444 \
           -I fork 'tsp -I dvb ... -P zap service0' \
           -I ip 226.2.2.2:1234 \
           -I http --infinite http://server.foo.com/service2/ \
           -I fork 'tsp -I dvb ... -P zap service3' \
  | tsp -P ... -O ...
```

In this command, the remote control will send commands to UDP port 4444. For instance:

```
$ echo >/dev/udp/127.0.0.1/4444 2
$ echo >/dev/udp/127.0.0.1/4444 0
$ echo >/dev/udp/127.0.0.1/4444 next
$ echo >/dev/udp/127.0.0.1/4444 prev
```

5.2 TSP Examples

This section demonstrates the usage of the transport stream processor on some typical examples. Refer to the documentation of each specific plugin for more details.

5.2.1 Capturing a TS from an external source

The following example captures 20 seconds of the satellite transponder containing the Canal+ service and saves it into a file. We assume that we have a DVB-S adapter and a dish which is pointed to the Astra satellite.

```
tsp -I dvb --channel canal+ \
    -P until --seconds 20 \
    -O file ts_capture.mpg
```

Same example, using specific tuning information for the satellite transponder (carrier 11.858 MHz, vertical polarity, 27.5 mega-symbols / second):

```
tsp -I dvb --tune 11856:v:0:27500 \
    -P until --seconds 20 \
    -O file ts_capture.mpg
```

Same example using short names for options:

```
tsp -I dvb -t 11856:v:0:27500 -P until -s 20 -O file ts_capture.mpg
```

5.2.2 Routing a TS between several physical transports

The following example reads the same satellite transponder and redirects its content to the first Dektec DVB-ASI output device. The output bitrate of the ASI stream is locked to the input bitrate (from the satellite transponder).

```
tsp -I dvb -t 11856:v:0:27500 -O dektec
```

5.2.3 Using IP multicast

The following example reads a transport stream from the second Dektec DVB-ASI input device (“*device 1*”), extracts the service named “Arte”, with French audio track only (identified as “fra” in the PMT) and broadcasts the resulting SPTS on the LAN using multicast IP (port 1000 on multicast address 224.10.11.12).

```
tsp -I dektec -d 1 \
    -P zap arte -a fra \
    -O ip 224.10.11.12:1000
```

Then, the service Arte can be received from any workstation on the LAN using, for instance, the free VLC media player.

As an alternative to VLC, the Linux receivers may use the following example to view the channel using the standard Linux media player:

```
tsp -I ip 224.10.11.12:1000 | mplayer -
```

5.2.4 Regulating the output speed

The following example reads a captured transport stream file, extracts the service Arte and broadcasts it on the LAN.

```
tsp –I file –i ts_capture.mpg \
    -P zap arte \
    -P pcrbitrate \
    -P regulate \
    -O ip 224.10.11.12:1000
```

Since reading a file can be extremely fast, it is not reasonable to broadcast the TS packets without regulation. If the receivers wish to play the TV program, the TS packets arrive too fast. The *pcrbitrate* plugin re-computes the expected TS bitrate after extraction of the selected service. Then, the *regulate* plugin introduces wait periods to slow down the stream to the previously computed bitrate.

On the contrary, when the input source is a live transponder, this kind of regulation may be useless since the input source is already regulated at the appropriate speed.

Unfortunately, this is not completely true in all cases. The *average* bitrate is regulated by the source (the live transponder) but there is a potential burst problem. If the broadcaster system and all receivers use the same type in connection to the LAN (100 Mb/s for instance) and if the LAN backbone does not slow down the bandwidth, this is fine. However, there is a problem if the broadcaster has a faster connection to the LAN than the receivers (say 100 Mb/s vs. 10 Mb/s). Of course, 10 Mb/s is enough to receive one service which usually needs around 4 Mb/s. However, there is a potential burst problem.

To avoid burst in case of non-homogeneous access speed to the LAN, the broadcaster should smooth the flow at all stages, as illustrated in the following command

```
tsp --max-input-packets 128 \
    –I dvb –c arte \
    -P zap arte \
    -P pcrbitrate --min-pcr 256 \
    -P regulate --packet-burst 128 \
    -O ip 224.10.11.12:1000 --packet-burst 128
```

5.2.5 Scheduling the recording of a program

The following example records the contents of the channel named “France 2” between 17:15 and 17:30 the 6th of July 2006.

```
tsp -I dvb -c france2 \
    -P time -d "" -p "2006/07/06:17:15:00" -s "2006/07/06:17:30:00" \
    -P zap france2 \
    -O file program.ts
```

The `–I` option selects the first DVB input device, tuning on the transponder containing the channel named “France 2”.

The first `–P` option specifies to:

- Initially drop packets (`-d ""`)
- Start passing packets at 17:15 the 6th of July 2006.
- Stop packet processing (and make tsp terminate) at 17:30 the 6th of July 2006.

The second `–P` option extracts only the service named “France 2” and the `–O` option finally saves the resulting SPTS in the file `program.ts`.

5.2.6 Extracting selected packets

The following silly example dumps the content of the 20th TS packet with the *payload unit start indicator* set in PID 0x0208:

```
tsp -I file /data1/mpeg/test/frtv_tnt.mpg \
    -P filter --pid 0x208 \
    -P filter --unit-start \
    -P skip 19 \
    -P until --packets 1 | \
    tsdump
```

Note that the *filter* plugin selects packets matching any of the specified conditions (an "*or*" selection). Here, to select packets matching two conditions (an "*and*" selection), we chain two *filter* plugins.

5.2.7 Monitoring selected MPEG tables (here, EMM's)

The following example demonstrates how to monitor the EMM's for a given operator. The first command determines on which PID are sent the EMMs. This command analyzes the satellite transponder which carries the channel Canal+ during 2 seconds. Instead of the full human-readable analysis report, we ask for a "*normalized*" output format and we filter the conditions we need: a line starting with "`pid:`" for description of a PID, "`:emm:`" for a PID carrying EMM's, "`:cas=256:`" to filter EMM's for CA System Id 256 (0x100, ie. MediaGuard).

```
tsp -I dvb -c canal+ \
    -P until -s 2 \
    -P analyze --normalized \
    -O drop | \
    grep ^pid: | grep :emm: | grep :cas=256:
```

The output of this command is:

```
pid:pid=193:emm:cas=256:access=clear: […]
pid:pid=196:emm:cas=256:operator=129:access=clear: […]
```

We now know that PID 193 carries the MediaGuard individual EMM's and PID 196 carries the MediaGuard group EMM's for operator 129 (OPI of Canal+).

The second command, below, filters the contents of those two PID's and formats the contents of the MPEG tables that are carried in those PID's:

```
tsp -I dvb -c canal+ -P filter -p 193 -p 196 | tstables | less
```

Of course, since EMM's are ciphered, their contents are obscure to the user and the display looks like:

```
* EMM (0x82), TID 130 (0x82), PID 193 (0x00C1)
  Version: 0, sections: 1, total size: 117 bytes
  - Section 0:
    0000:  00 00 09 F3 87 00 00 80 00 B0 10 01 5E E7 07 85  ...ó.....°..^ç..
    0010:  22 C3 DB 13 75 43 3B 5C 1E 08 DC 4A 05 35 AD 54  "ÃÛ.uC;\..ÜJ.5-T
    0020:  B5 52 35 B1 61 FB 37 BB EC 6D 55 F5 21 B6 4C 58  µR5±aû7»ìmUõ!¶LX
    0030:  80 F4 FA FB D9 C5 D0 A2 C7 22 BA 77 51 B9 C8 96  .ôúûÙÅÐ¢Ç"ºwQ¹È.
    0040:  A3 79 9E 5A 24 74 2A 01 7D 00 62 A3 EC D4 AF DF  £y.Z$t*.}.b£ìÔ¯ß
    0050:  F2 43 B1 3A 72 B5 B3 E0 C9 22 68 2D 50 F0 FE 82  òC±:rµ³àÉ"h-Pðþ.
    0060:  47 1F AC 95 5F D2 59 E6 C8 C6 78 BE F3 C5 A9 CF  G.¬._ÒYæÈÆx¾óÅ©Ï
    0070:  05 90                                            ..

* EMM (0x82), TID 130 (0x82), PID 193 (0x00C1)
  Version: 0, sections: 1, total size: 105 bytes
  - Section 0:
    0000:  00 00 F1 F2 F3 F4 00 00 00 B0 10 01 98 3E EF 81  ..ñòóô...°...>ï.
    0010:  45 E1 A1 D3 76 B9 B0 21 D6 F9 5F AB 4B 07 9D 13  Eá¡Óv¹°!Öù_«K...
    ...
```

5.2.8 Scanning all services by CAS operator

The following complex example scans a complete satellite network, looking for the list of services which are scrambled for an operator.

We assume that we have a DVB-S adapter and a dish which is pointed to the Astra satellite.

The first command scans the NIT (Network Information Table) of a known transponder. The output is the list of all transponders in the network. This list is sorted and duplicate lines are removed ("sort –u").

Then, each transponder is analyzed during 3 seconds ("-P until -s 3") and the result of the analysis in normalized format is saved in a temporary file. From this analysis file, we extract the PID's carrying ECM's with CA system id 256 (MediaGuard) and MediaGuard OPI 128 (CanalSat). For each ECM PID, we extract the list of services this PID belongs to.

Thus, for each transponder, we get a list of services (actually, a list of *service ids*) which are scrambled for the CanalSat MediaGuard operator. Finally, we use again the transponder analysis in normalized format to get the service name for each of these service id.

```
inittune=11856:v:0:27500  # Initial transponder to scan the NIT
cas=256                   # MediaGuard CA system id
opi=128                   # MediaGuard OPI for CanalSat

tsp -I dvb -t $inittune -P nitscan -t -O drop | \
sort -u | \
while read tune; do
    tsp -I dvb -t $tune \
        -P until -s 3 \
        -P analyze --normalized -o tmp.tmp \
        -O drop
    grep "^pid:" tmp.tmp | \
    grep ":ecm:" | \
    grep ":cas=$cas:" | \
    grep ":operator=$opi:" | \
    sed -e 's/^.*:servlist=//' -e 's/:.*$//' -e 's/,/\n/' | \
    while read serv; do
        grep "^service:" tmp.tmp | \
        grep ":id=$serv:" | \
        sed -e "s/^.*:name=/Transponder: $tune  Service: /"
    done
    rm -f tmp.tmp
done
```

The output of this script gives the following output (107 lines):

```
Transponder: 11739:v:0:27500  Service: MTV F
Transponder: 11739:v:0:27500  Service: MTV HITS.
Transponder: 11739:v:0:27500  Service: MTV Base.
...
Transponder: 12640:v:0:22000  Service: TOON DISNEY
Transponder: 12640:v:0:22000  Service: MOTORS TV
Transponder: 12640:v:0:22000  Service: E! ENTERTAINMENT
```

5.2.9 On-the-fly replacement of an SI table

The following example tests an updated version of a *Bouquet Association Table* (BAT) on a live transport stream.

We assume to have a DVB-T tuner card to capture live streams and a Dektec DTA-110T DVB-T modulator (PCI card) to send the modified stream into a local distribution network (or even to one single directly-connected STB).

We capture one transport stream (the "R4" from the French DTTV network, on UHF channel 24). We remove the BAT of the *Tv Numéric* operator and we replace it with a new one, the table we wish to test. The new table is stored in binary section format into a file named `BAT_TvNumeric_V3.si`.

First, we capture all tables from the PID `0x0011` (the one which carries the SDT's and the BAT's).

```
rm -f r4_p0011_*.si  # remove previous files if any
tsp -I dvb -u 24 -P until -s 10 -P filter -p 0x011 | tstables -m -b r4.si
rm -f r4_p0011_t4A_e0086_*.si # remove current Tv Numeric BAT
```

These commands capture and save all tables (SDT's and BAT's) in binary files named `r4_p0011_*.psi` during 10 seconds. Each section is stored in a separate file (option `-m` in *tstables*). The current TV Numeric BAT is removed. Note the file name `r4_p0011_t4A_e0086_*.si` which means all sections from PID `0x0011` with TID `0x4A` (BAT) and TID extension `0x0086` (bouquet identifier for operator TV Numeric).

The following command now performs the live replacement. The *inject* plugin is used to replace the content of PID `0x0011` with the sections in all the specified files. These files are all the previously captured sections from this PID (minus the previous BAT which was deleted) and the new BAT.

```
tsp -I dvb –u 24 \
    -P inject --replace 0x0011 r4_p0011_*.si BAT_TvNumeric_V3.si \
    -O dektec –u 24 --convolution 2/3 --guard 1/32
```

5.2.10 Performing the global analysis of a transponder

The following command receives a DVB-T transport stream from UHF channel 35 during 100 seconds and produces an analysis report in the text file *R1.analysis*. The first 5000 packets are ignored since the signal may not be quite stable right after the tuning operation.

```
tsp -I dvb -u 35 \
    -P skip 5000 \
    -P until -s 100 \
    -P analyze --title "R1 (Channel 35)" -o R1.analysis \
    -O drop
```

The report file is quite large:

```
================================================================================
|  TRANSPORT STREAM ANALYSIS REPORT                          R1 (Channel 35)  |
|==============================================================================|
|  Transport Stream Id: ..... 1 (0x0001)  |  PID's: Total: ..... 35  |
|  Bytes: ..................... 317,825,468  |         Clear: ..... 35  |
|  TS packets: ................... 1,690,561  |         Scrambled: ..... 0  |
|  Invalid TS packets: ................... 0  |         With PCR's: ..... 6  |
|  Services: ............................. 7  |         Unreferenced: ... 0  |
|------------------------------------------------------------------------------|
|  Transport stream bitrate, based on ....... 188 bytes/pkt    204 bytes/pkt  |
|  User-specified: ......................... 24,882,352 b/s   26,999,998 b/s  |
|  Estimated based on PCR's: ............... 24,882,351 b/s   26,999,998 b/s  |
|------------------------------------------------------------------------------|
|  Broadcast time: ................................. 102 sec (1 mn 42 sec)  |
|  First TDT time stamp: ............................... 2008/06/11 09:34:25  |
|  Last TDT time stamp: ................................ 2008/06/11 09:35:37  |
|  TOT country code: ..................................................... FRA  |
|------------------------------------------------------------------------------|
| Serv.Id  Service Name                                Access          Bitrate  |
|  0x0101  France 2 ....................................  C    3,637,078 b/s  |
|  0x0104  France 5 ....................................  C    4,567,443 b/s  |
|  0x0105  ARTE ........................................  C    3,688,018 b/s  |
|  0x0106  LCP .........................................  C    3,554,581 b/s  |
|  0x0111  France 3 ....................................  C    4,828,238 b/s  |
|  0x0176  .France Ô ...................................  C    3,286,441 b/s  |
|  0x01FF  (System Software Update) ....................  C       35,015 b/s  |
|                                                                              |
|  Note 1: C=Clear, S=Scrambled                                                |
|  Note 2: Unless explicitely specified otherwise, all bitrates are based on  |
|  188 bytes per packet.                                                       |
================================================================================


================================================================================
|  SERVICES ANALYSIS REPORT                                  R1 (Channel 35)  |
|==============================================================================|
|  Global PID's                                                                |
|  TS packets: 87,342, PID's: 7 (clear: 7, scrambled: 0)                       |
|------------------------------------------------------------------------------|
|     PID  Usage                                       Access          Bitrate  |
|   Total  Global PID's  ...............................  C    1,285,534 b/s  |
|  0x0000  PAT  ........................................  C       15,027 b/s  |
|  0x0010  DVB-NIT  ....................................  C        4,503 b/s  |
|  0x0011  SDT/BAT  ....................................  C          750 b/s  |
|  0x0012  EIT  ........................................  C       37,075 b/s  |
```

```
|  0x0014  TDT/TOT  ..................................  C          132 b/s  |
|  0x0015  Network Synchronization  ...................  C        2,737 b/s  |
|  0x1FFF  Stuffing  .................................  C    1,225,306 b/s  |
|=============================================================================|
|  Service: 257 (0x0101), TS: 1 (0x0001), Original Netw: 8442 (0x20FA)        |
|  Service name: France 2, provider: GR1                                      |
|  Service type: 1 (0x01), Digital television service                         |
|  TS packets: 247,111, PID's: 4 (clear: 4, scrambled: 0)                     |
|  PMT PID: 110 (0x006E), PCR PID: 120 (0x0078)                               |
|-----------------------------------------------------------------------------|
|     PID  Usage                                        Access          Bitrate  |
|   Total  Digital television service  .................  C    3,637,078 b/s  |
|  0x006E  PMT  ......................................  C       15,042 b/s  |
|  0x0078  MPEG-2 Video  .............................  C    3,404,836 b/s  |
|  0x0082  MPEG-1 Audio (fra)  .......................  C      198,433 b/s  |
|  0x008C  Subtitles (fra, DVB subtitles, no aspect rati  C       18,765 b/s  |
|          (C=Clear, S=Scrambled, +=Shared)                                   |
|=============================================================================|
|  Service: 260 (0x0104), TS: 1 (0x0001), Original Netw: 8442 (0x20FA)        |
```

... more services skipped ...

```
|=============================================================================|
|  Service: 511 (0x01FF), TS: 1 (0x0001), Original Netw: 8442 (0x20FA)        |
|  Service name: (System Software Update), provider: (unknown)                |
|  Service type: 0 (0x00), Reserved service type 0x00                         |
|  TS packets: 2,379, PID's: 2 (clear: 2, scrambled: 0)                       |
|  PMT PID: 1010 (0x03F2), PCR PID: None                                      |
|-----------------------------------------------------------------------------|
|     PID  Usage                                        Access          Bitrate  |
|   Total  Reserved service type 0x00  .................  C       35,015 b/s  |
|  0x0294  DSM-CC U-N (SSU Sagem Communication)  .......  C       19,987 b/s  |
|  0x03F2  PMT  ......................................  C       15,027 b/s  |
|          (C=Clear, S=Scrambled, +=Shared)                                   |
===============================================================================


===============================================================================
|  PIDS ANALYSIS REPORT                                    R1 (Channel 35)  |
|=============================================================================|
|  PID: 0 (0x0000)                                                      PAT  |
|-----------------------------------------------------------------------------|
|  Global PID               Transport:               Discontinuities:       |
|  Bitrate: .... 15,027 b/s  Packets: ......... 1,021  Expected: ......... 0  |
|  Access: Clear            Adapt.F.: ............ 0  Unexpect: ......... 0  |
|                           Duplicated: .......... 0  Sections:              |
|                           PCR: ................. 0  Unit start: ... 1,021  |
|                                                                             |
|=============================================================================|
|  PID: 16 (0x0010)                                                  DVB-NIT  |
```

... more PID's skipped ...

```
|=============================================================================|
|  PID: 8191 (0x1FFF)                                                Stuffing  |
|-----------------------------------------------------------------------------|
|  Global PID               Transport:               Discontinuities:       |
|  Bitrate: . 1,225,306 b/s  Packets: ........ 83,250  Expected: ......... 0  |
|  Access: Clear            Adapt.F.: ............ 0  Unexpect: ......... 0  |
|                           Duplicated: .......... 0  Sections:              |
|                           PCR: ................. 0  Unit start: ....... 0  |
|                                                                             |
===============================================================================
```

5.2.11 Performing the global analysis of a network

This section presents an automated way to analyze a network (here, the French terrestrial network) using a GNU makefile.

Using the simple command "`make`", each known transport stream (designated by its UHF channel number) is analyzed. For each TS, for instance the one named R1, the following text files are created:

- `R1.analysis` : Global analysis of the TS in human-readable format, as in 5.2.10.
- `R1.anl` : Global analysis of the TS in normalized format, for use by other scripts.
- `R1.psi` : Analysis of the main PSI/SI tables (PAT, CAT, PMT, SDT, NIT, BAT).

Individual targets, such as "`make R1`" can be used to analyze only one TS. Use the make option `-B` to force the analysis again when the files already exist.

The command "`make capture`" captures 120 seconds of each TS in files named `R1.ts`, `R2.ts`, etc. Similarly, commands like "`make R1.ts`" capture only one TS.

The content of the makefile follows:

```
# === This is a GNU makefile ===

# List of UHF channels:

ALL_CHAN = R1 R2 R3 R4 R5 R6 L8

R1_CHAN = 35
R2_CHAN = 21
R3_CHAN = 27
R4_CHAN = 24
R5_CHAN = 29
R6_CHAN = 32
L8_CHAN = 23

# Channel full names:

$(foreach R,$(ALL_CHAN),$(eval $R_NAME=$R (Channel $($R_CHAN))))

# Default target is analysis of all TS

all: $(ALL_CHAN)

$(ALL_CHAN): %: %.analysis %.anl %.psi

%.analysis %.services %.anl %.psi:
    tsp -I dvb $(DEVICE) -u $($(*F)_CHAN) \
        -P skip 5000 \
        -P until -s 100 \
        -P analyze --title "$($(*F)_NAME)" -o $*.analysis \
        -P analyze --title "$($(*F)_NAME)" -o $*.anl --normalized \
        -P psi -a -o $*.psi \
        -O drop

# Capture TS content:

capture: $(foreach R,$(ALL_CHAN),$R.ts)

%.ts:
    tsp -I dvb $(DEVICE) -u $($(*F)_CHAN) \
        -P skip 5000 \
        -P until -s 120 \
        -O file $@
```

5.2.12 Monitoring the stuffing rate of all transponders in a network

The following script monitors the stuffing bitrate of a list of selected transport streams. The output is suitable for importation into Excel so that further analysis can be performed. It can be executed on Linux or Windows (using the Cygwin shell).

In this script, the transport streams are designated by a list of UHF channels, meaning DVB-T only. Here, the UHF channels represent the 5 main MUX of the French DTTV in the Paris area.

```
# List of UHF channels
UHF_CHANNELS="35 21 27 24 32"

# Analysis time per TS, in seconds
ANALYSIS_TIME=20

# Sample interval, in seconds
SAMPLE_INTERVAL=300

# Excel separator character for "csv" files (depends on Excel locale)
EXCEL_SEPARATOR=';'

# Main loop
while true; do

    # Current date in seconds since epoch
    curtime=$(date "+%s")

    # Loop on all TS
    outline=
    for uhf in $UHF_CHANNELS; do
        stuffing=$(
            tsp -I dvb -u $uhf \
                -P until -s $ANALYSIS_TIME \
                -P analyze --normalized \
                -O drop | \
                grep '^pid:' | \
                grep ':pid=8191:' | \
                sed -e 's/^.*:bitrate=//' -e 's/:.*//')
        outline="${outline}${EXCEL_SEPARATOR}${stuffing}"
    done

    # Current date and stuffing rates in Excel format
    echo "$(date -d @$curtime '+%d/%m/%Y %H:%M')${outline}"

    # Sleep until next sample time
    sleeptime=$(( $curtime + $SAMPLE_INTERVAL - $(date "+%s") ))
    [[ $sleeptime -le 0 ]] || sleep $sleeptime
done
```

The script runs infinitely and produces the following output:

```
12/06/2008 14:01;1208706;4501497;3762828;626932;1145037
12/06/2008 14:06;1232543;4505620;3782431;621524;1172479
12/06/2008 14:11;1225293;4505553;3487315;613616;1151119
12/06/2008 14:16;1231288;4505958;3415868;665393;1156933
....
```

It may be imported into Microsoft Excel to produce the following graph:

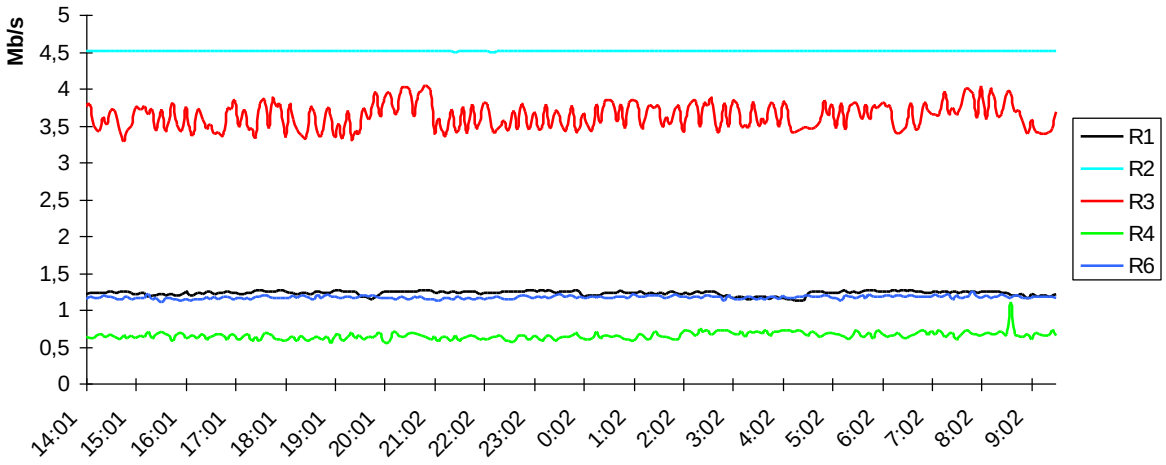

Figure 4: Stuffing bitrate sample diagram

5.2.13 Analyzing the bitrate of all services in a network

The following script demonstrates a way to produce a report of the bitrate of all services in a network. First, you need to analyze all TS in the network and get the result in *normalized format* (see 5.2.11 for an example). Then run the following script on all normalized analysis files.

```
echo "MUX  Service                    Bitrate   Video bitrate  Access"
echo "---  ------------------  --------------  --------------  ---------"

for f in $*; do
    tsid=$(grep '^ts:' $f | sed -e 's/.*:id=//' -e 's/:.*//')
    grep '^service:' $f | grep ':servtype=1:' | \
    while read line; do
        name=$(sed <<<"$line" -e 's/.*:name=//')
        bitrate=$(sed <<<"$line" -e 's/.*:bitrate=//' -e 's/:.*//')
        access=$(sed <<<"$line" -e 's/.*:access=//' -e 's/:.*//')
        pidgrep=$(sed <<<"$line" -e 's/.*:pidlist=//' -e 's/:.*//' \
                                 -e 's/^/-e :pid=/' \
                                 -e 's/,/: -e :pid=/g' -e 's/$/:/')
        vbitrate=0
        for br in $(grep '^pid:' $f | grep $pidgrep | grep ':video:' | \
                    sed -e 's/.*:bitrate=//' -e 's/:.*//')
        do
            vbitrate=$(( $vbitrate + $br ))
        done
        printf "R%d   %-18s  %'10d b/s  %'10d b/s  %s\n" \
            $tsid "$name" $bitrate $vbitrate $access
    done
done
```

When used in conjunction with the makefile from 5.2.11, you get:

```
make -f Makefile.tnt
...
bitrate-summary *.anl
MUX  Service                    Bitrate   Video bitrate  Access
---  ------------------  --------------  --------------  ---------
R8   Canal 21             2,803,938 b/s   2,588,374 b/s  clear
R8   IDF1                 3,502,350 b/s   3,254,550 b/s  clear
R8   NRJ Paris            6,462,333 b/s   6,214,518 b/s  clear
R8   CAP 24               2,929,000 b/s   2,681,200 b/s  clear
R1   France 2             3,655,962 b/s   3,419,466 b/s  clear
R1   France 5             4,600,309 b/s   4,379,003 b/s  clear
R1   ARTE                 5,052,002 b/s   4,627,464 b/s  clear
R1   LCP                  2,867,453 b/s   2,649,782 b/s  clear
```

```
R1   France 3             3,510,985 b/s   3,293,801 b/s  clear
R1   .France Ô            3,857,456 b/s   3,643,981 b/s  clear
R2   Direct 8             2,740,873 b/s   2,432,179 b/s  clear
R2   BFM TV               3,120,068 b/s   2,913,715 b/s  clear
R2   i>TELE               2,699,497 b/s   2,493,143 b/s  clear
R2   Virgin 17            4,947,397 b/s   4,676,283 b/s  clear
R2   Gulli                3,280,344 b/s   3,036,397 b/s  clear
R2   France 4             2,748,753 b/s   2,477,639 b/s  clear
R3   CANAL+               8,369,816 b/s   7,477,442 b/s  scrambled
R3   CANAL+ CINEMA        2,975,779 b/s   2,531,416 b/s  scrambled
R3   CANAL+ SPORT         2,930,938 b/s   2,493,595 b/s  scrambled
R3   PLANETE              2,340,974 b/s   2,095,053 b/s  scrambled
R3   CANAL J              2,609,858 b/s   2,371,848 b/s  scrambled
R3   TPS STAR             3,203,408 b/s   2,779,778 b/s  scrambled
R4   M6                   4,628,819 b/s   3,834,868 b/s  clear
R4   W9                   3,231,344 b/s   2,694,826 b/s  clear
R4   NT1                  3,278,883 b/s   2,887,844 b/s  clear
R4   PARIS PREMIERE       4,009,594 b/s   3,404,277 b/s  scrambled
R4   ARTE HD              7,725,247 b/s   7,171,310 b/s  clear
R5   TF1 HD               9,032,166 b/s   8,635,108 b/s  clear
R5   France 2 HD          7,593,045 b/s   7,080,227 b/s  clear
R5   M6HD                 7,301,165 b/s   6,714,945 b/s  clear
R6   TF1                  5,022,465 b/s   3,951,056 b/s  clear
R6   NRJ12                6,883,049 b/s   6,026,657 b/s  clear
R6   LCI                  1,379,288 b/s   1,224,422 b/s  scrambled
R6   Eurosport            3,535,155 b/s   3,380,304 b/s  scrambled
R6   TF6                  1,701,739 b/s   1,543,181 b/s  scrambled
R6   TMC                  4,103,693 b/s   3,890,212 b/s  clear
```

5.2.14 Analyzing the number of PCR per second

It is sometimes useful to get a complete overview of the number of PCR per second in each service of a network. The following script illustrates this. First, you need to analyze all TS in the network and get the result in *normalized format* (see 5.2.11 for an example). Then run the following script on all normalized analysis files.

```
for file in $*; do
    sec=$(grep '^ts:' $file | grep ':duration=' | \
          sed -e 's/.*:duration=//' -e 's/:.*//')
    if [[ "$sec" -gt 1 ]]; then
        grep '^service:' $file | grep ':pcrpid=' | grep ':name=' |
        while read line; do
            pid=$(sed <<<$line -e 's/.*:pcrpid=//' -e 's/:.*//')
            name=$(sed <<<$line -e 's/.*:name=//')
            count=$(grep '^pid:' $file | grep ":pid=$pid:" | grep ':pcr=' | \
                    sed -e 's/.*:pcr=//' -e 's/:.*//')
            if [[ "$count" -ne 0 ]]; then
                printf "%4d PCR/s - %s\n" \
                       $((($count + ($sec / 2)) / $sec)) "$name"
            fi
        done
    fi
done | sort
```

When used in conjunction with the makefile from 5.2.11, you get:

```
make -f Makefile.tnt
...
pcrrate *.anl
  29 PCR/s - France 2 HD
  29 PCR/s - TF1 HD
  30 PCR/s - ARTE
  30 PCR/s - ARTE HD
  30 PCR/s - BFM TV
  30 PCR/s - Canal 21
  30 PCR/s - CAP 24
```

```
30 PCR/s - Direct 8
30 PCR/s - France 2
30 PCR/s - France 3
30 PCR/s - France 4
30 PCR/s - France 5
30 PCR/s - .France Ô
30 PCR/s - Gulli
30 PCR/s - IDF1
30 PCR/s - i>TELE
30 PCR/s - LCP
30 PCR/s - M6
30 PCR/s - M6HD
30 PCR/s - NRJ12
30 PCR/s - NRJ Paris
30 PCR/s - NT1
30 PCR/s - Virgin 17
30 PCR/s - W9
31 PCR/s - CANAL+
31 PCR/s - TF1
31 PCR/s - TMC
50 PCR/s - CANAL+ CINEMA
50 PCR/s - CANAL J
50 PCR/s - CANAL+ SPORT
50 PCR/s - Eurosport
50 PCR/s - LCI
50 PCR/s - PARIS PREMIERE
50 PCR/s - PLANETE
50 PCR/s - TF6
50 PCR/s - TPS STAR
```

5.2.15 Injecting a System Software Update (SSU) service into a transport stream

This example illustrates how to inject a new System Software Update (SSU) service into a transport stream as defined in [9]. This type of procedure can be used to test the SSU capabilities of a Set Top Box in real conditions, using a live transport stream.

The test is the following:

- A DVB-T transport stream is received on UHF channel 24.
- This transport stream has at least 56 kb/s of stuffing packets (much more actually). Our *tsp* command steals 56 kb/s of stuffing and replaces them with a new service (16 kb/s for the new service's PMT and 40 kb/s for the SSU data PID).
- The STB software provider delivers three types of SSU tables: a DSI, a DII and a lot of DDB's. The tables are provided as binary files containing the sections. There is one file `dsi.bin` containing the DSI section, one file `dii.bin` containing the DII section and one file `ddb.bin` containing all DDB sections.
- These tables are multiplexed in the same SSU data PID but have different repetition rates constraints. Here, we use 14 seconds for the DSI and 60 seconds for the DII. The DDB use the rest of the available bitrate in the SSU data PID.
- After analysis of the transport stream, the new SSU service will use the service id `0x04F0` and PID values `0x1F00` (SSU data) and `0x1F01` (PMT). These values are chosen since they are not used in the original transport stream.
- The resulting transport stream with the added SSU service in sent to an embedded Dektec OFDM modulator on the same frequency as the original service. The output of the modulator can be directly connected to a STB.

The PMT of the service is defined as follow in file `pmt.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<tsduck>
  <PMT service_id="0x04F0">
    <component elementary_PID="0x1F00" stream_type="0x0B">
      <data_broadcast_id_descriptor data_broadcast_id="0x000A">
```

```
        <selector_bytes>
          0C 00 12 22 F1 DF 06 FF FF FF FF F0 F0
        </selector_bytes>
      </data_broadcast_id_descriptor>
    </component>
  </PMT>

</tsduck>
```

In this example, the specified OUI value and selector bytes are those which are used by Logiways SSU on Skardin-based STB.

The binary version of the PMT is generated in file `pmt.bin` by the table compiler:

```
tstabcomp pmt.xml
```

The files `pmt.bin`, `dsi.bin`, `dii.bin` and `ddb.bin` are injected in the transport stream using the following command:

```
tsp -I dvb -u 24 \
    -P pat -v 31 -a 0x04F0/0x1F01 \
    -P inject -b 16000 -p 0x1F01 -s pmt.bin \
    -P inject -b 40000 -p 0x1F00 -s dsi.bin=14000 dii.bin=60000 ddb.bin \
    -O dektec -u 24 --convolution 2/3 --guard 1/32
```

Notes: We have previously checked in the TS that the PAT version was not 31. By assigning the new version 31 to the PAT, we state that the content of the PAT has changed. Thus, the STB will analyze it again and will discover the new service.

In the case were the transport stream does not initially contain enough stuffing to inject the SSU service, it is possible to remove a service and replace it with stuffing. In the following command, the service named AB1 is first replaced by stuffing, representing a stuffing increase of 4 Mb/s.

```
tsp -I dvb -u 24 \
    -P svremove –s AB1 \
    -P pat -v 31 -a 0x04F0/0x1F01 \
    -P inject -b 16000 -p 0x1F01 -s pmt.bin \
    -P inject -b 40000 -p 0x1F00 -s dsi.bin=14000 dii.bin=60000 ddb.bin \
    -O dektec -u 24 --convolution 2/3 --guard 1/32
```

5.2.16 Analyzing EPG data

This example illustrates how to analyze EIT sections and report which service supports EPG data (EIT schedule) and for how many days. The command analyzes the content of UHF channel 27 (DVB-T) during 30 seconds and reports a summary of EIT analysis.

```
$ tsp -I dvb -u 27 -P until -s 30 -P eit -O drop
Summary
-------
TS id:         3 (0x0003)
Last UTC:      2008/08/13 14:19:28
EITp/f actual: 186
EITp/f other:  435
EITs actual:   461
EITs other:    0

TS      Services  With EITp/f  With EITs  EPG days
------  --------  -----------  ---------  --------
Actual         8            6          6         3
Other         66           66          0         0

A/O  TS Id   Srv Id  Name          EITp/f  EITs  EPG days
---  ------  ------  ------------  ------  ----  --------
Oth  0x0001  0x0101                Yes     No           0
Oth  0x0001  0x0104                Yes     No           0
Oth  0x0001  0x0105                Yes     No           0
Oth  0x0001  0x0106                Yes     No           0
Oth  0x0001  0x0110                Yes     No           0
```

```
Oth  0x0001  0x0111                Yes     No          0
Oth  0x0001  0x0112                Yes     No          0
Oth  0x0001  0x0113                Yes     No          0
Oth  0x0001  0x0114                Yes     No          0
Oth  0x0001  0x0115                Yes     No          0
Oth  0x0001  0x0116                Yes     No          0
Oth  0x0001  0x0117                Yes     No          0
Oth  0x0001  0x0118                Yes     No          0
Oth  0x0001  0x0119                Yes     No          0
Oth  0x0001  0x011A                Yes     No          0
Oth  0x0001  0x011B                Yes     No          0
Oth  0x0001  0x011C                Yes     No          0
Oth  0x0001  0x011D                Yes     No          0
Oth  0x0001  0x011E                Yes     No          0
Oth  0x0001  0x011F                Yes     No          0
Oth  0x0001  0x0120                Yes     No          0
Oth  0x0001  0x0121                Yes     No          0
Oth  0x0001  0x0122                Yes     No          0
Oth  0x0001  0x0123                Yes     No          0
Oth  0x0001  0x0124                Yes     No          0
Oth  0x0001  0x0125                Yes     No          0
Oth  0x0001  0x0126                Yes     No          0
Oth  0x0001  0x0127                Yes     No          0
Oth  0x0001  0x0128                Yes     No          0
Oth  0x0001  0x0129                Yes     No          0
Oth  0x0001  0x012A                Yes     No          0
Oth  0x0001  0x012B                Yes     No          0
Oth  0x0001  0x012C                Yes     No          0
Oth  0x0001  0x012D                Yes     No          0
Oth  0x0001  0x012E                Yes     No          0
Oth  0x0001  0x012F                Yes     No          0
Oth  0x0001  0x0130                Yes     No          0
Oth  0x0001  0x0131                Yes     No          0
Oth  0x0001  0x0132                Yes     No          0
Oth  0x0001  0x0133                Yes     No          0
Oth  0x0001  0x0134                Yes     No          0
Oth  0x0001  0x0135                Yes     No          0
Oth  0x0001  0x0136                Yes     No          0
Oth  0x0001  0x0137                Yes     No          0
Oth  0x0001  0x0138                Yes     No          0
Oth  0x0001  0x0139                Yes     No          0
Oth  0x0001  0x013A                Yes     No          0
Oth  0x0001  0x013B                Yes     No          0
Oth  0x0001  0x0176                Yes     No          0
Oth  0x0002  0x0201                Yes     No          0
Oth  0x0002  0x0203                Yes     No          0
Oth  0x0002  0x0204                Yes     No          0
Oth  0x0002  0x0205                Yes     No          0
Oth  0x0002  0x0206                Yes     No          0
Oth  0x0002  0x0207                Yes     No          0
Act  0x0003  0x0301  CANAL+        Yes     Yes         3
Act  0x0003  0x0302  CANAL+ CINEMA Yes     Yes         3
Act  0x0003  0x0303  CANAL+ SPORT  Yes     Yes         3
Act  0x0003  0x0304  PLANETE       Yes     Yes         3
Act  0x0003  0x0305  CANAL J       Yes     Yes         3
Act  0x0003  0x0306  TPS STAR      Yes     Yes         3
Act  0x0003  0x03F0                No      No          0
Act  0x0003  0x03F1                No      No          0
Oth  0x0004  0x0401                Yes     No          0
Oth  0x0004  0x0402                Yes     No          0
Oth  0x0004  0x0403                Yes     No          0
Oth  0x0004  0x0404                Yes     No          0
Oth  0x0004  0x0405                Yes     No          0
Oth  0x0004  0x0406                Yes     No          0
Oth  0x0006  0x0601                Yes     No          0
Oth  0x0006  0x0602                Yes     No          0
```

```
Oth  0x0006  0x0603                Yes     No          0
Oth  0x0006  0x0604                Yes     No          0
Oth  0x0006  0x0606                Yes     No          0
$
```

5.2.17 Analyzing audio and video attributes

This example illustrates how to display the audio and video attributes from a captured transport stream file.

```
$ tsp –I file cap.ts –P pes –a –v –O drop
* PID 0x0083, stream_id 0xC0 (Audio 0), audio attributes:
  Audio layer II, 160 kb/s, sampled at 48,000 Hz, stereo
* PID 0x014A, stream_id 0xC0 (Audio 0), audio attributes:
  Audio layer II, 192 kb/s, sampled at 48,000 Hz, stereo
* PID 0x0085, stream_id 0xC0 (Audio 0), audio attributes:
  Audio layer II, 64 kb/s, sampled at 48,000 Hz, single channel
* PID 0x0082, stream_id 0xC0 (Audio 0), audio attributes:
  Audio layer II, 192 kb/s, sampled at 48,000 Hz, stereo
* PID 0x0276, stream_id 0xC0 (Audio 0), audio attributes:
  Audio layer II, 192 kb/s, sampled at 48,000 Hz, stereo
* PID 0x01AE, stream_id 0xC0 (Audio 0), audio attributes:
  Audio layer II, 256 kb/s, sampled at 48,000 Hz, stereo
* PID 0x00E6, stream_id 0xC0 (Audio 0), audio attributes:
  Audio layer II, 256 kb/s, sampled at 48,000 Hz, stereo
* PID 0x0078, stream_id 0xE0 (Video 0), video attributes:
  720x576i, 25 Hz, 16/9, 4:2:0
  Maximum bitrate: 15,000,000 b/s, VBV buffer size: 1,835,008 bits
* PID 0x01A4, stream_id 0xE0 (Video 0), AVC video attributes:
  720x576, AVC main profile (77), level 30
* PID 0x00DC, stream_id 0xE0 (Video 0), video attributes:
  720x576i, 25 Hz, 16/9, 4:2:0
  Maximum bitrate: 15,000,000 b/s, VBV buffer size: 1,835,008 bits
* PID 0x026C, stream_id 0xE0 (Video 0), video attributes:
  720x576i, 24 Hz, 4/3, 4:2:0
  Maximum bitrate: 15,000,000 b/s, VBV buffer size: 1,835,008 bits
* PID 0x0140, stream_id 0xE0 (Video 0), AVC video attributes:
  704x576, AVC main profile (77), level 30
$
```

5.2.18 Conditional Access System scrambling and ECM functional tests

The following command receives a DVB-T live stream on UHF channel 21 and remodulates it on the same frequency using a Dektec modulator. In the middle, the service named BFM TV is scrambled. An external ECMG is used (host name `ecmg1` on TCP port 10000). The crypto-periods are scheduled using the default duration of 10 seconds. A new control word is generated for each crypto-period. The corresponding ECM's are generated using the specified ECMG (*Super_CAS_Id* and access criteria specified by options `-s` and `-a`) and inserted in the TS. The PMT of the service is modified to include a *CA_descriptor*. The private part of this descriptor is specified using option `-p`.

```
tsp -I dvb –u 21 \
    -P scrambler bfmtv \
        -e ecmg1:10000 \
        -s 0x4ADC0001 \
-a 6B0A0101000000000000000006B0A0102000000000000000006105000000500066040000000 2 \
        -p FE \
    -O dektec –u 21 --conv 2/3 --guard 1/32
```

5.2.19 Complete Conditional Access System test bed

The following commands implements a complete Conditional Access System test bed in one single *tsp* process. It emulates all functions of a MUX system for testing a CAS.

The command uses the French DVB-T network but it can be easily adapted to any environment.

The command transforms the R2 MUX into a new R9 MUX with new services (actually renamed services from R2) and outputs the resulting TS to a modulator on a different UHF channel. In the meantime, the service named "*Gulli Test*" is scrambled using an external ECMG and EMM injection is allowed from an external EMMG.

The modulated output stream can be used alone (direct connection to STB) or mixed with the public antenna signals using a UHF coupler.

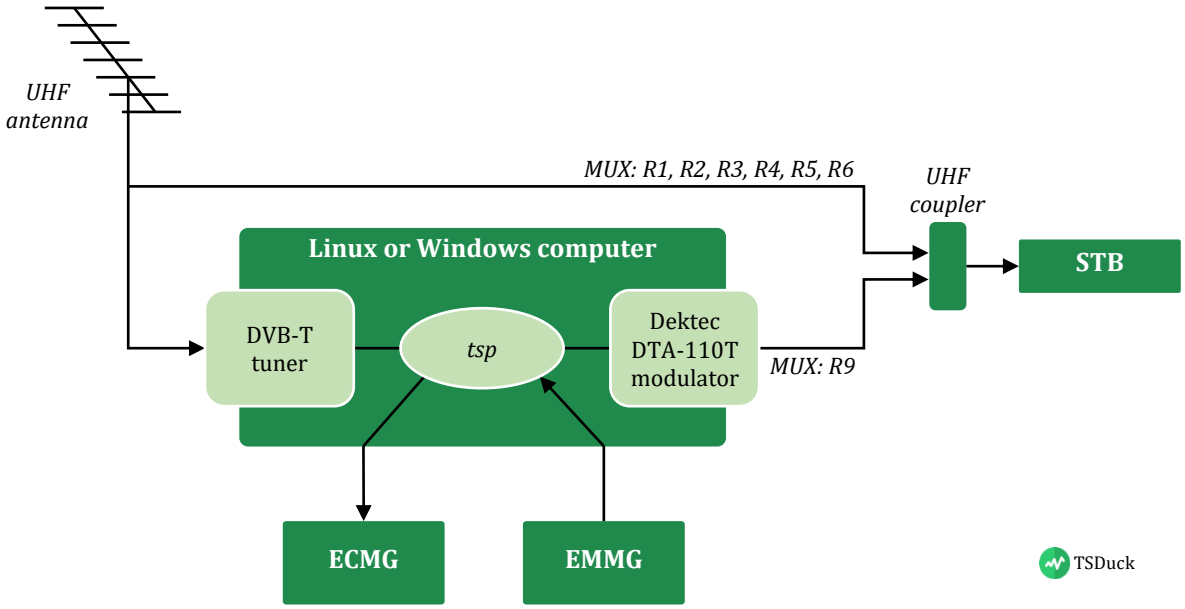

Figure 5: Conditional Access System sample test bed

For the sake of clarity of this example, all significant parameters are first assigned into environment variables, then the tsp command references these variables.

```
# Transmission parameters:
UHF_INPUT=21
UHF_OUTPUT=60

# EMM parameters
MUX_SERVER_PORT=32000
CAS_ID=0x4ADC
EMM_PID=0x01F0
EMM_MAX_BITRATE=50000
CAT_CADESC_PRIVATE=FF0001

# ECM parameters
ECMG=ecmg1:10000
SUPER_CAS_ID=0x4ADC0001
ECM_PID=0x01F1
ECM_BITRATE=30000
PMT_CADESC_PRIVATE=FE
AC=6B0A0101000000000000000006B0A0102000000000000000006105000000500066040000000 2

# One single command implementing the CAS test bed:
tsp -v \
    -I dvb -u $UHF_INPUT \
    -P tsrename -t 9 -a \
    -P svrename direct8  -i 0x0901 -l 41 -n "Direct 8 Test" \
    -P svrename bfmtv    -i 0x0903 -l 42 -n "BFM TV Test" \
    -P svrename 'i>tele' -i 0x0904 -l 43 -n "i>TELE Test" \
    -P svrename virgin17 -i 0x0905 -l 44 -n "Virgin 17 Test" \
    -P svrename gulli    -i 0x0906 -l 45 -n "Gulli Test" \
```

```
    -P svrename france4  -i 0x0907 -l 46 -n "France 4 Test" \
    -P svrename 0x02FF   -i 0x09FF \
    -P scrambler GulliTest -e $ECMG -s $SUPER_CAS_ID -p $PMT_CADESC_PRIVATE \
                -a $AC -b $ECM_BITRATE --pid $ECM_PID \
    -P cat -c -a $CAS_ID/$EMM_PID/$CAT_CADESC_PRIVATE \
    -P datainject -r -s $MUX_SERVER_PORT -b $EMM_MAX_BITRATE -p $EMM_PID \
    -O dektec --uhf $UHF_OUTPUT --convolution 2/3 --guard 1/32
```

5.2.20 Emulation of a Conditional Access head-end

This example is a variant of the previous one. Instead of using a real ECMG from a real Conditional Access System, we use the command *tsecmg*.

The utility *tsecmg* implements the DVB SimulCrypt ECMG⇔SCS protocol and behaves like a real ECMG. All ECM generation requests are accepted but, instead of generating robust ciphered proprietary ECM's, *tsecmg* returns pseudo ECM's which contain the control words and the access criteria in the clear.

The utility *tsecmg* can be used anywhere a DVB SimulCrypt ECMG can be used. Consequently, it can be used from any real MUX or from the *tsp* plugin *scrambler*. Used from a real MUX, *tsecmg* becomes a useful debugging tool. All ECMG⇔SCS messages are displayed (using option `--verbose`). The returned ECM's are inserted in the stream like any real ECM. Since these ECM's contain the access criteria in the clear, this is also a useful debug tool for the EIS or ACG or both.

Important: Note that the control words are also inserted in the clear. It is consequently obvious that *tsecmg* shall never be used on a production system, alone or in addition to any real operational CAS.

The *tsp* plugin *descrambler* is normally a static descrambler using fixed control words. But, to facilitate the prototyping of end-to-end systems, the plugin *descrambler* can also recognize the clear ECM's which are generated by *tsecmg* and use their control words to descramble the stream.

Thus, it is easy to build a complete end-to-end Conditional Access System using TSDuck components only. This kind of configuration is mainly useless in itself (except maybe as a tutorial for DVB SimulCrypt). But because all components are replaceable, this can become a very useful integration framework. First, start with a complete configuration using TSDuck components only. Verify that the system works as expected. Then, replace the TSDuck components one by one with the real components which shall be tested.

Sample configurations:

- Testing a MUX : Replace the plugin *scrambler* with the real MUX. Use *tsecmg* to generate ECM's. Use the plugin *analyze* to analyze the output of the MUX. Use the plugin *descrambler* to verify the insertion and synchronization of ECM's.
- Testing a CAS : Replace *tsecmg* with the real ECMG. Replace the plugin *descrambler* with a real set-top box. Use plugin *scrambler* to make the link between to two end-points of the CAS (ECMG and STB).

Let's have a look at a real demo.

First, run the utility *tsecmg*. Without option, it simply creates a TCP server on port 2222. The option `--verbose` (or simply `–v`) is useful to dump all protocol exchanges.

```
tsecmg -v
* TCP server listening on 0.0.0.0:2222, using ECMG <=> SCS protocol version 2
```

Then, the following command performs a complete end-to-end CAS demo in one single process, using a live satellite stream as input:

```
tsp -v \
    -I dvb --freq 12,012,000,000 --symbol 29,700,000 --fec 5/6 --polarity vertical \
           --delivery DVB-S2 --modulation QPSK \
    -P scrambler cnews --ecmg localhost:2222 --super-cas-id 0xDEADBEEF \
           --access-criteria 0123456789 \
    -P analyze --interval 30 -o cas_scrambled.txt \
    -P descrambler cnews \
    -P analyze --interval 30 -o cas_descrambled.txt \
    -P zap cnews \
    -O play
```

The first plugin receives a live transport stream from a DVB-S2 satellite. In this TS, there is a clear service named “CNEWS”. We are going to use this clear channel as a test.

The next plugin scrambles the service using our instance of *tsecmg* on the same system. The Super CAS Id is here a fake value (*tsecmg*, unlike a real ECMG, accepts to serve any Super CAS Id). The access criteria are also fake values.

The next plugin permanently analyzes the stream at this point in the chain and produces a report every 30 seconds in a text file named *cas_scrambled.txt*. Looking at this text file, we can see that the service CNEWS is now scrambled and there is an ECM stream in its PMT with CA_system_id 0xDEAD (the MSB part of the Super CAS Id). The bitrate of the ECM stream is reported as 30 kb/s, the default ECM bitrate for the plugin *scrambler*.

The next plugin is a descrambler. Only the service name is required. The plugin automatically locates the ECM stream in the PMT (there is only one here), collects the ECM’s and uses the clear control words from these fake ECM’s to descramble the stream.

The next plugin performs the same periodic analysis as the previous one. This time, the report demonstrates that the service CNEWS is back in the clear.

Finally, the plugin *zap* extracts the service CNEWS and the output plugin *play* sends the output to a media player (VLC by default).

We can see that the service is in the clear and plays correctly. If we restart the command without the plugin *descrambler*, the player stays with a black screen because the service stays scrambled.

If we are interested in the DVB SimulCrypt ECMG⇔SCS protocol, the option `--verbose` of *tsecmg* displays all exchanges, as listed below. This can be useful to debug an ECMG⇔SCS integration.

```
* 127.0.0.1:1302: 2018/04/10 23:11:58: session started
* 127.0.0.1:1302: 2018/04/10 23:11:58: received message:
    channel_setup (ECMG<=>SCS)
    protocol_version = 0x02
    message_type = 0x0001
    ECM_channel_id = 0x0001
    Super_CAS_id = 0xDEADBEEF

* 127.0.0.1:1302: 2018/04/10 23:11:58: sending message:
    channel_status (ECMG<=>SCS)
    protocol_version = 0x02
    message_type = 0x0003
    ECM_channel_id = 0x0001
    section_TSpkt_flag = 1
    AC_delay_start = 200
    AC_delay_stop = 200
    delay_start = 200
    delay_stop = 200
    transition_delay_start = -500
    transition_delay_stop = 0
    ECM_rep_period = 100
    max_streams = 0
    min_CP_duration = 10
    lead_CW = 1
    CW_per_msg = 2
    max_comp_time = 100

* 127.0.0.1:1302: 2018/04/10 23:11:58: received message:
    stream_setup (ECMG<=>SCS)
    protocol_version = 0x02
    message_type = 0x0101
    ECM_channel_id = 0x0001
    ECM_stream_id = 0x0001
    ECM_id = 0x0001
    nominal_CP_duration = 100

* 127.0.0.1:1302: 2018/04/10 23:11:58: sending message:
    stream_status (ECMG<=>SCS)
```

```
    protocol_version = 0x02
    message_type = 0x0103
    ECM_channel_id = 0x0001
    ECM_stream_id = 0x0001
    ECM_id = 0x0001
    access_criteria_transfer_mode = 0

* 127.0.0.1:1302: 2018/04/10 23:11:58: received message:
    CW_provision (ECMG<=>SCS)
    protocol_version = 0x02
    message_type = 0x0201
    ECM_channel_id = 0x0001
    ECM_stream_id = 0x0001
    CP_number = 0
    CP_duration = 100
    access_criteria (5 bytes) =
        01 23 45 67 89
    CP = 0
    CW (8 bytes) = 26 E9 2C D9 C8 96 06 B2
    CP = 1
    CW (8 bytes) = 8B 37 0B 94 69 64 93 CE

* 127.0.0.1:1302: 2018/04/10 23:11:58: sending message:
    ECM_response (ECMG<=>SCS)
    protocol_version = 0x02
    message_type = 0x0202
    ECM_channel_id = 0x0001
    ECM_stream_id = 0x0001
    CP_number = 0
    ECM_datagram (188 bytes) =
        47 5F FF 10 00 80 70 26 80 AA 03 00 21 00 10 00 08 26 E9 2C D9 C8
        96 06 B2 00 11 00 08 8B 37 0B 94 69 64 93 CE 00 12 00 05 01 23 45
        67 89 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
        FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
        FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
        FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
        FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
        FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
        FF FF FF FF FF FF FF FF FF FF FF FF
```

The returned ECM is a TS packet containing a section with *table_id* 0x80 (an ECM). The payload of the ECM is a TLV structure following the same syntax as DVB SimulCrypt protocols. The command and parameter tags are private to TSDuck and documented in its development documentation (Doxygen-generated, available online).

To create a more realistic environment, we can split the big command into two parts, a “head-end part” which can be replaced by a real MUX and a “set-top box part” which can be replaced by a real STB. The communication between the two parts can be done using a modulator-tuner pair, an ASI link or UDP/IP. All these interconnections are supported by TSDuck and can be driven directly from *tsp*.

Let’s have a look at the head-end emulation command, using an UDP/IP output link:

```
tsp -v \
    -I dvb --freq 12,012,000,000 --symbol 29,700,000 --fec 5/6 --polarity vertical \
           --delivery DVB-S2 --modulation QPSK \
    -P scrambler cnews --ecmg localhost:2222 --super-cas-id 0xDEADBEEF \
           --access-criteria 0123456789 --atis-idsa \
    -P zap cnews \
    -O ip 224.10.11.12:9999
```

The output is a multicast address.

Also note that we used the option `--atis-idsa` in the plugin *scrambler*. This means that we use the ATIS IIF Default Scrambling Algorithm (IDSA) instead of the default DVB Common Scrambling Algorithm (CSA2). ATIS being based on AES-128, we can see in the ECMG⇔SCS exchanges that the control words are now 16-byte long. There is no particular reason to use ATIS in this demo (except that ATIS is typically used in IP-TV while DVB-CSA2 is mainly used in broadcast).

The set-top box emulation command is simply:

```
tsp -v \
    -I ip 224.10.11.12:9999 \
    -P descrambler cnews \
    -O play
```

Note that we do not need to specify `--atis-idsa` in the plugin *descrambler*. During the scrambling, the plugin *scrambler* has inserted a *scrambling_descriptor* in the PMT of the service to indicate the non-default scrambling type. This descriptor is automatically recognized by the plugin *descrambler* and the right descrambling algorithm is used, just like any properly integrated set-top box would do.

5.2.21 Multi-Protocol Encapsulation (MPE)

This example describes a test bed or demo infrastructure for MPE injection and MPE extraction. See [14] for more details on MPE.

The network infrastructure is illustrated in the diagram below.

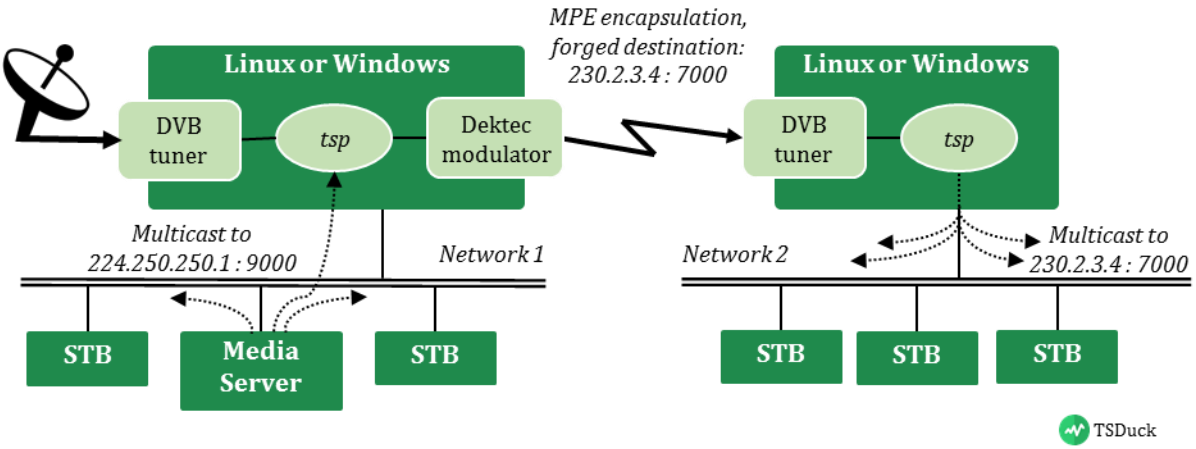

Figure 6: Multi-Protocol Encapsulation (MPE) sample test bed

In network 1, a media server multicasts a transport stream on address 224.250.250.1, port 9000.

We want to encapsulate this UDP multicast stream in an existing transport stream using MPE. We do this using *tsp*. We also change the multicast destination address for the UDP stream to 230.2.3.4, port 7000, in the MPE-encapsulated datagrams. There is no particular reason for this, we just illustrate the feasibility.

The resulting transport stream with embedded MPE is then broadcast. Here, the broadcast network is a Dektec modulator, followed by another computer using a DVB tuner.

This computer is connected to a second network. Another instance of *tsp* extracts the datagrams from the MPE stream and multicasts them on its network using the modified destination address.

Let's review the various steps and commands in details.

The existing transport stream is here a live satellite TS which is received on a Linux or Windows computer using a DVB tuner. The insertion of the MPE stream adds two new services. We carefully select service ids and PID's which are not used in the existing transport stream.

- A service carrying the IP/MAC Notification Table (INT).
  - Service id: 700
  - Service name: "Demo INT"
  - PMT PID: 5000
  - PID of the component carrying the INT: 5001
- A service carrying the MPE stream. Such a service may carry many MPE streams. Here, we use only one.
  - Service id: 701
  - Service name: "Demo MPE"

- PMT PID: 5002
- PID of the component carrying the MPE stream: 5003

We need to create three tables from scratch, the PMT's of the two new services and the INT. We create them using XML files.

PMT of the service carrying the INT (file `pmt-int.xml`):

```
<?xml version="1.0" encoding="UTF-8"?>
<tsduck>
  <!-- See ETSI EN 301 192, section 8.3 -->
  <PMT service_id="700">
    <component elementary_PID="5001" stream_type="0x05">
      <data_broadcast_id_descriptor data_broadcast_id="0x000B"/>
    </component>
  </PMT>
</tsduck>
```

PMT of the service carrying the MPE stream (file `pmt-mpe.xml`):

```
<?xml version="1.0" encoding="UTF-8"?>
<tsduck>
  <!-- See ETSI EN 301 192, section 7.2 -->
  <PMT service_id="701">
    <component elementary_PID="5003" stream_type="0x0D">
      <stream_identifier_descriptor component_tag="1"/>
      <data_broadcast_id_descriptor data_broadcast_id="0x0005"/>
    </component>
  </PMT>
</tsduck>
```

IP/MAC Notification Table (file `int.xml`):

```
<?xml version="1.0" encoding="UTF-8"?>
<tsduck>
  <!-- See ETSI EN 301 192, section 8.4 -->
  <INT platform_id="0x123456">
    <IPMAC_platform_name_descriptor language_code="eng" text="Demo"/>
    <IPMAC_platform_provider_name_descriptor language_code="eng" text="TSDuck"/>
    <device>
      <target>
        <target_IP_slash_descriptor>
          <address IPv4_addr="230.2.3.4" IPv4_slash_mask="32"/>
        </target_IP_slash_descriptor>
      </target>
      <operational>
        <IPMAC_stream_location_descriptor
            network_id="1"
            original_network_id="1"
            transport_stream_id="1080"
            service_id="701"
            component_tag="1"/>
      </operational>
    </device>
  </INT>
</tsduck>
```

On the first system, the following command is used to insert the MPE stream:

```
tsp -I dvb --frequency ... \
    -P svremove Service1 --stuffing \
    -P pat --add-service 700/5000 --add-service 701/5002 \
    -P inject pmt-int.xml --pid 5000 --bitrate 15000 \
    -P inject int.xml     --pid 5001 --bitrate 15000 \
    -P inject pmt-mpe.xml --pid 5002 --bitrate 15000 \
    -P sdt --service-id 700 --name "Demo INT" --provider "TSDuck" --type 0x0C \
    -P sdt --service-id 701 --name "Demo MPE" --provider "TSDuck" --type 0x0C \
    -P mpeinject 224.250.250.1:9000 --max-queue 512 \
                 --new-destination 230.2.3.4:7000 --pid 5003 \
```

```
    -O dektec --frequency ...
```

The following chain of plugins is used:

- The input plugin *dvb* receives an existing satellite stream.
- The plugin *svremove* removes one service from the TS and replaces it with stuffing. We are going to insert an MPE stream and we need bandwidth for it. If the existing TS does not have enough stuffing bandwidth, we need to create some. Depending on the target MPE bandwidth, we may need to remove several existing services.
- The plugin *pat* adds the two new services in the PAT.
- The three plugin *inject* insert the three XML tables we created, each one on its own PID.
- The two plugins *sdt* add the descriptions of the two new services in the SDT.
- The plugin *mpeinject* inserts the MPE stream. It receives the UDP multicast datagrams for address 224.250.250.1, port 9000. In each datagram, the destination address is modified as 230.2.3.4, port 7000. The UDP datagrams are encapsulated into MPE sections which are injected in PID 5003. The option `--max-queue` is a tuning parameter. It specifies the number of UDP datagrams which can be buffered before insertion in the MPE stream. The parameter shall be tuned according to the receiving multicast rate and bursts and the placement of stuffing packets in the exiting TS. We need to tune it when we get "*UDP overflow*" messages.
- Finally, the plugin *dektec* sends the resulting TS on a modulator.

On the second system, the following command is used to extract the MPE stream and to re-multicast it on the network 2:

```
tsp -I dvb --frequency ... -P mpe --udp-forward -O drop
```

Here, the command is simple since we assume that there is only one MPE stream in the TS and it is properly signaled in the PSI/SI. If there are several MPE streams in the TS, more options are required in the plugin *mpe*.

The option `--udp-forward` specifies that the UDP datagrams shall be forwarded on the local network. Note that when the UDP packets are multicast and the system running *tsp* has several network interfaces, it may be necessary to specify the `--local-address` option to select through which local interface the multicast packets shall be sent.

We may want to use *tsanalyze* on the intermediate transport stream. The two services we created are described as follow:

```
|*****************************************************************************|
|  Service: 0x02BC (700), TS: 0x0438 (1080), Original Netw: 0x0001 (1)        |
|  Service name: Demo INT, provider: TSDuck                                   |
|  Service type: 0x0C (Data broadcast service)                                |
|  TS packets: 600, PID's: 2 (clear: 2, scrambled: 0)                         |
|  PMT PID: 0x1388 (5000), PCR PID: None                                      |
|-----------------------------------------------------------------------------|
|     PID  Usage                                      Access          Bitrate |
|   Total  Data broadcast service ..................... C       29,938 b/s  |
|  0x1388  PMT ......................................... C       14,969 b/s  |
|  0x1389  MPEG-2 Private sections (INT, IP/MAC Notifica  C       14,969 b/s  |
|          (C=Clear, S=Scrambled, +=Shared)                                   |
|*****************************************************************************|
|  Service: 0x02BD (701), TS: 0x0438 (1080), Original Netw: 0x0001 (1)        |
|  Service name: Demo MPE, provider: TSDuck                                   |
|  Service type: 0x0C (Data broadcast service)                                |
|  TS packets: 154,507, PID's: 2 (clear: 2, scrambled: 0)                     |
|  PMT PID: 0x138A (5002), PCR PID: None                                      |
|-----------------------------------------------------------------------------|
|     PID  Usage                                      Access          Bitrate |
|   Total  Data broadcast service ..................... C    7,709,471 b/s  |
|  0x138A  PMT ......................................... C       14,969 b/s  |
|  0x138B  DSM-CC Sections (MPE) ....................... C    7,694,502 b/s  |
|          (C=Clear, S=Scrambled, +=Shared)                                   |
|*****************************************************************************|
```

5.2.22 DVB-T2 Modulator Interface (T2-MI)

A DVB T2-MI stream is encapsulated into one PID of a TS. A DVB-T2 stream may contain several Physical Layer Pipes (PLP). Each PLP contains a complete TS. The plugin *t2mi* is designed to extract the TS from a PLP of a T2-MI stream.

With a fully DVB-compliant signalization, the PID carrying T2-MI is signaled in the PMT of its service using a T2-MI descriptor.

Sample PMT using *tstables*:

```
* PMT, TID 2 (0x02), PID 33 (0x0021)
  Version: 11, sections: 1, total size: 27 bytes
  - Section 0:
    Program: 800 (0x0320), PCR PID: none
    Elementary stream: type 0x06 (MPEG-2 PES private data), PID: 64 (0x0040)
    - Descriptor 0: Extension Descriptor (0x7F, 127), 4 bytes
      Extended descriptor: T2MI (0x11, 17)
      T2-MI stream id: 0, T2-MI stream count: 1, PCR/ISCR common clock: no
```

Excerpt from *tsanalyze* for the service containing the T2-MI stream:

```
|=============================================================================|
|  Service: 0x0320 (800), TS: 0x03A2 (930), Original Netw: 0x0000 (0)         |
|  Service name: (T2-MI), provider: (unknown)                                 |
|  Service type: 0x00 (unknown)                                               |
|  TS packets: 145,024, PID's: 2 (clear: 2, scrambled: 0)                     |
|  PMT PID: 0x0021 (33), PCR PID: None                                        |
|-----------------------------------------------------------------------------|
|     PID  Usage                                      Access          Bitrate |
|   Total  unknown (0x00) ............................ C          Unknown  |
|  0x0021  PMT ....................................... C          Unknown  |
|  0x0040  T2-MI (PLP: 0x66 (102)) ................... C          Unknown  |
|          (C=Clear, S=Scrambled, +=Shared)                                   |
|=============================================================================|
```

The option `--identify` of the plugin *t2mi* lists the same information. With this option, the plugin does not modify the stream, it only identify T2-MI PID's and PLP's.

```
$ tsp -I dvb ... -P t2mi --identify -O drop
* t2mi: found T2-MI PID 0x0040 (64)
* t2mi: PID 0x0040 (64), found PLP 102
^C
* tsp: user interrupt, terminating...
* t2mi: summary: found 1 PID's with T2-MI
* t2mi: PID 0x0040 (64): PLP 102
$
```

But since T2-MI streams are received by designated professional equipment, many operators do not setup the required signalization and it is necessary to guess which PID in which service may carry T2-MI.

Example service which is a good candidate for T2-MI:

```
|=============================================================================|
|  Service: 0x0320 (800), TS: 0x03A2 (930), Original Netw: 0x0000 (0)         |
|  Service name: (unknown), provider: (unknown)                               |
|  Service type: 0x00 (unknown)                                               |
|  TS packets: 27,805,661, PID's: 2 (clear: 2, scrambled: 0)                  |
|  PMT PID: 0x0100 (256), PCR PID: None                                       |
|-----------------------------------------------------------------------------|
|     PID  Usage                                      Access          Bitrate |
|   Total  unknown (0x00) ............................ C          Unknown  |
|  0x0100  PMT ....................................... C          Unknown  |
|  0x1000  MPEG-2 PES private data ................... C          Unknown  |
|          (C=Clear, S=Scrambled, +=Shared)                                   |
|=============================================================================|
```

In this example, the TS contains only one service. This service contains only one component and it carries private sections. Since there is no video PID, there is no PCR and *tsanalyze* is not able to compute bitrates.

If we know, from other sources, that the TS contains T2-MI, it must be there. In this case, we need to explicitly provide the PID number to the plugin *t2mi*:

```
$ tsp -I dvb ... -P t2mi --pid 0x1000 --identify -O drop
* t2mi: PID 0x1000 (4096), found PLP 0
* t2mi: PID 0x1000 (4096), found PLP 2
* t2mi: PID 0x1000 (4096), found PLP 1
^C
* tsp: user interrupt, terminating...
* t2mi: summary: found 1 PID's with T2-MI
* t2mi: PID 0x1000 (4096): PLP 0, 1, 2
$
```

If we want to redistribute on a local DVB network one of these PLP's, the command is the following:

```
$ tsp -I dvb ... -P t2mi --pid 0x1000 --plp 1 -O dektec ...
```

Without the option `--identify`, the plugin *t2mi* extracts the TS from the specified PLP and completely replaces the TS with the extracted one. The output of the plugin is the extracted TS, the original TS carrying T2-MI has disappeared. The final output is a Dektec modulator (or ASI board) which broadcasts the extracted TS.

The plugin *t2mi* can extract only one PLP because this is the basic principle of *tsp*: end-to-end processing of one single TS. Even if one plugin produces a radical transformation such as completely replacing the TS with another one (here, the extracted PLP), there is only one TS at all points in the chain.

If we want to process all PLP's at the same time, we must re-route the original TS in parallel instances of *tsp* using the plugin *fork*. Each instance of *tsp* extracts one PLP.

This is illustrated by the following command:

```
tsp -I dvb ... \
    -P until –seconds 30 \
    -P fork 'tsp -P t2mi --pid 0x1000 --plp 0 -P analyze -o plp0.txt -O drop' \
    -P fork 'tsp -P t2mi --pid 0x1000 --plp 1 -P analyze -o plp1.txt -O drop' \
    -P fork 'tsp -P t2mi --pid 0x1000 --plp 2 -P analyze -o plp2.txt -O drop' \
    -P analyze -o main.txt -O drop
```

This command analyzes the enclosing stream and the three different PLP's in parallel during 30 seconds. Each plugin *fork* creates a process and passes the complete TS to this process. Each created process runs another instance of *tsp* which extracts one PLP. Note that the default input plugin of *tsp* is the plugin *file* which, by default, reads the standard input.

5.2.23 Merging transport streams

The plugin *merge* can be used to merge a transport stream into another one. The service references are correctly merged into the final transport stream.

Let's illustrate this using two live transport streams from satellite Astra 19.2 E. We use one transport stream as base. We remove one service from this stream and we replace it with another live service coming from another transport stream.

We use the transport stream with id 1028 as base. Using the plugin *dvb*, the tuning options are:

```
--freq 11,626,500,000 --symbol 22,000,000 --fec 5/6 --polarity vertical --delivery DVB-S
```

To simplify the command lines, we save these options, one per line, in a text file named `ts1028.txt` to be used by partial command line redirection (see 3.1.5).

The structure of this transport stream can be seen using the plugin *analyze*. Here is the list of services from the analyze output:

```
|-------------------------------------------------------------------------|
|  Srv Id  Service Name                              Access          Bitrate  |
|  0x1131  TVE INTERNACIONAL EUROPA ..................... C    3,572,410 b/s  |
|  0x1132  CANAL 24 HORAS ............................... C    3,423,043 b/s  |
|  0x113B  RNE RADIO 1 .................................. C      140,998 b/s  |
|  0x113C  RNE RADIO 3 .................................. C      278,941 b/s  |
|  0x113D  RNE RADIO 4 .................................. C      141,045 b/s  |
```

```
|  0x113E  RNE RADIO 5 TODO NOTICIAS ..................... C      141,092 b/s  |
|  0x113F  RNE RADIO CLASICA ............................. C      347,819 b/s  |
|  0x1140  RNE RADIO EXTERIOR DE ESPAÑA .................. C      141,139 b/s  |
|  0x1146  CNN Int. ...................................... C    4,008,806 b/s  |
|  0x114E  DW (English) .................................. C    3,488,065 b/s  |
|  0x1158  Al Jazeera English ............................ C    3,803,631 b/s  |
|-------------------------------------------------------------------------------|
```

We use the transport stream with id 1022 to extract a service and inject it into the previous transport stream. The *dvb* tuning options are:

```
--freq 11,538,000,000 --symbol 22,000,000 --fec 5/6 --polarity vertical --delivery DVB-S
```

Again, we save them, one by line, in a text file named `ts1022.txt`.

The list of services from transport stream 1022 is shown below. Note that we use transport streams with clear channels only to be able to watch the result.

```
|-------------------------------------------------------------------------------|
|  Srv Id  Service Name                                Access          Bitrate  |
|  0x1AF4  DATASYSTEM .................................. C       44,884 b/s  |
|  0x1AF8  Russia Today ................................ C    4,038,071 b/s  |
|  0x1AF9  France 24 (en Français) ..................... C    2,703,659 b/s  |
|  0x1AFA  France 24 (in English) ...................... C    2,700,842 b/s  |
|  0x1AFE  France 24 (in Arabic) ....................... C    2,465,576 b/s  |
|  0x1B00  CGTN Documentary ............................ C    2,504,263 b/s  |
|  0x1B01  CGTN F ...................................... C    2,247,071 b/s  |
|  0x1B02  CGTN ........................................ C    2,443,463 b/s  |
|  0x1B03  TV5MONDE EUROPE ............................. C    3,619,747 b/s  |
|  0x1B06  TRT World HD ................................ C    5,984,240 b/s  |
|-------------------------------------------------------------------------------|
```

We assume that we have two satellite tuners in the system. Adapter 0 will be used to receive TS 1022 and adapter 1 will be used to receive TS 1028.

We also have a Dektec modulator to redistribute the resulting transport stream. Again, to shorten the command line, we place all modulation options into one text file named `modulation.txt`.

In our example, we extract the service *TV5MONDE EUROPE* from TS 1022 and we merge it into TS 1028. To make sure that the transport stream has enough free space, we remove the service *Al Jazeera English* from TS 1028 before the merge[7].

The merging command is the following:

```
tsp -I dvb -a 1 @ts1028.txt \
    -P svremove -s AlJazeeraEnglish \
    -P merge "tsp -I dvb -a 0 @ts1022.txt -P zap TV5MondeEurope" \
    -P analyze -i 30 -o merged.txt \
    -O dektec @modulation.txt
```

Note that the service *Al Jazeera English* is replaced by stuffing (option `-s`) in TS 1028.

In the created command, everything is removed from TS 1022, except service *TV5MONDE EUROPE* (the service names are not case-sensitive and spaces are ignored).

The final plugin *analyze* continuously analyzes the output stream and produces a report file every 30 seconds. Here is the merged list of services from this report:

```
|-------------------------------------------------------------------------------|
|  Srv Id  Service Name                                Access          Bitrate  |
|  0x1131  TVE INTERNACIONAL EUROPA ..................... C    3,529,507 b/s  |
|  0x1132  CANAL 24 HORAS ............................... C    3,382,237 b/s  |
|  0x113B  RNE RADIO 1 .................................. C      139,475 b/s  |
|  0x113C  RNE RADIO 3 .................................. C      275,915 b/s  |
```

[7] In practice, on the day of this experiment, the service TS 1028 had more than enough stuffing to insert one or two services without removing any other. However, in the general case, we need to make some room first. So, let's do it anyway.

```
|  0x113D  RNE RADIO 4 ................................. C      139,425 b/s  |
|  0x113E  RNE RADIO 5 TODO NOTICIAS .................... C      139,374 b/s  |
|  0x113F  RNE RADIO CLASICA ............................ C      343,831 b/s  |
|  0x1140  RNE RADIO EXTERIOR DE ESPAÑA ................. C      139,425 b/s  |
|  0x1146  CNN Int. ..................................... C    3,963,218 b/s  |
|  0x114E  DW (English) ................................. C    3,448,180 b/s  |
|  0x1B03  TV5MONDE EUROPE .............................. C    3,566,805 b/s  |
|-------------------------------------------------------------------------------|
```

We can see that the service *Al Jazeera English* has been replaced with *TV5MONDE EUROPE*.

Luckily, there was no PID or service id conflict between the two transport streams. If the same service id or PID had existed in the two streams, the plugin *merge* would have reported an error and the component from the merged stream would have been dropped. In case of conflict, we use the plugin *remap* to modify PID's or the plugin *svrename* to rename a service (including modifying its service id).

5.2.24 Injecting SCTE 35 cue information

SCTE 35 cue information are single-section tables which are sent in one dedicated PID in a service. These commands are used to signal *video splicing points* where alternate content (typically ads) can replace the original video content. The video splicing points are defined by PTS (presentation time stamp) values in the video PID of the service.

Inserting SCTE 35 cue information is consequently different from traditional signalization. Each section defines one specific splicing event. It is not cycled. It is inserted once or twice only in the PID. The traffic on this PID is very low and not regular (it depends on the occurrences of the splicing events).

The splicing points are usually defined on the fly, with the cooperation of the video encoder. The exact PTS values of the splicing points are defined in real time. It is usually impossible to define in advance the list of all splicing events in the life of a service. Moreover, inserting cue information section needs to be synchronized with the associated video PID. Typically, a splice event is signaled twice, once two seconds before the event and once one second before.

Because of this dynamics, there are two distinct use cases: real-time live streams and offline test files.

5.2.24.1 Real-time live stream

This example illustrates the insertion of cue information in a real-time live transport stream. We receive a DTTV stream from a DVB-T tuner, we insert cue information for one service and we restream the result through a DVB-T modulator.

The transmission chain is processed by a *tsp* command. The generation of the cue information is externally performed by some real-time system, cooperating with the content management system and the video encoder. The format of the splice commands is defined by SCTE 35 [17]. Splice information sections can be provided in binary or XML format (see 2.2).

Here is an XML example of a pair of splice commands, a "splice out" event, followed 20 seconds later by a "splice in" event. This is typically an ads replacement opportunity. Here, the sections are minimal. You may want to add "break duration" information or additional descriptors.

```
<?xml version="1.0" encoding="UTF-8"?>
<tsduck>

  <splice_information_table>
    <splice_insert splice_event_id="100" unique_program_id="1"
                   out_of_network="true" pts_time="0x078CA4459"/>
  </splice_information_table>

  <splice_information_table>
    <splice_insert splice_event_id="100" unique_program_id="1"
                   out_of_network="false" pts_time="0x78E5BB99"/>
  </splice_information_table>

</tsduck>
```

The PTS timestamps are synchronized with the video PID. Moreover, these timestamp shall correspond to *video splice points*, ie. frames where the binary replacement of the encoded video content is smooth. This is why splicing shall be done with the cooperation of the video encoder.

The following command performs the real-time processing.

```
tsp -I dvb -u 24 \
    -P pmt --service 1010 --add-programinfo-id 0x43554549 --add-pid 600/0x86 \
    -P spliceinject --service 1010 --files 'splice-*.xml' --udp 4444 \
    -O dektec -u 24 --convolution 2/3 --guard 1/32
```

The input plugin *dvb* reads a DVB-T live stream from UHF channel 24. At the end of the processing, the output plugin *dektec* sends the stream to a Dektec DVB-T modulator on the same frequency.

The transport stream contains several services. We will add cue information on service id 1010. We could process multiple services using successive instances of the plugin *spliceinject* in the same command.

The plugin *pmt* modifies the PMT of the target service on the fly. To comply with the SCTE 35 standard, we add a *registration_descriptor* with id `0x43554549` and we add the declaration of the PID 600 with stream type `0x86` (meaning SCTE 35 cue information).

The plugin *spliceinject* performs the injection. The service id is sufficient to locate the target PID: the plugin searches the service and then searches a component with stream type `0x86` in its PMT.

The splice information sections can be supplied in real time using two methods: file (binary or XML) and UDP datagrams. Here, for the sake of the example, we use both. We can also use only one. The file specification is a wildcard because different files can be provided. It is also possible to rewrite the same file. Each time a file is modified, it is reloaded. We can also receive UDP datagrams (here on port number 4444). The datagram can contain binary or XML sections.

See the reference documentation of the plugin *spliceinject* for more details.

5.2.24.2 Cue insertion in offline files

Sometimes, it is necessary to prepare a transport stream file for demo or test. Usually, the same *tsp* command can be used indifferently on live streams and offline files. For SCTE 35 cue information, this is a bit different because of the dynamics.

The plugin *spliceinject* processes a transport stream and, on the other hand, it asynchronously receives splice information sections. On a real time stream, the events are received slightly in advance but in a timely fashion. The plugin does not expect any section file to be present at the time the processing starts. It does not wait for the section files.

With an offline transport stream file, the processing is very fast, running at the speed of the disk storage. So, even if the splice information section files are already present, they are read asynchronously from the transport stream processing. Specifically, they can be read *after* the processing the target event in the stream. In that case, no section would be injected. To avoid this problem, we use the option `--wait-first-batch` which forces the transport stream processing to wait for at least the first batch of splice section files. Thus, if the section files are present at the time the command is run, it is guaranteed that they are loaded and injected.

Additionally, in the following example, we use an SPTS file (containing only one service). These files usually contain no stuffing. However, *tsp* cannot insert new packets in a transport stream. It can only replace stuffing packets. So, unlike broadcast transport streams which always contain some stuffing, nothing can be injected in such a stream. To make room for insertion, we use the *tsp* option `--add-input-stuffing` which artificially injects null packets at input level (here one null packet every 10 input packets). This artificial stuffing will be used by *spliceinject* to insert its sections. Note that we remove the extra unused stuffing before the output plugin using the plugin *filter*.

The rest of the command is similar to the previous example.

```
tsp --add-input-stuffing 1/10 \
    -I file spts.ts \
    -P pmt --service 1010 --add-programinfo-id 0x43554549 --add-pid 600/0x86 \
    -P spliceinject --service 1010 --files splice.xml --wait-first-batch \
    -P filter --negate --pid 0x1FFF \
    -O file spts-out.ts
```

5.2.25 Encapsulating PID's into a private tunnel

In this example, we start from a transport stream containing two services, CANAL+ DECALE (id 0x2262) and CNEWS (id 0x226A). We want to process the first service through some external equipment (transcoder, transrater, etc.) However, this equipment can only process SPTS (single-program transport stream).

To preserve the structure of the transport stream while crossing this equipment, we use the following trick. We encapsulate the second service into one single "tunnel" PID and then we erase this service from the structure of the TS. This tunnel PID is added as a private component of the first service. Now, we have a true SPTS which can be processed by the external equipment.

The structure of the service CNEWS is reported as follow by *tsanalyze*:

```
|=============================================================================|
|  Service: 0x226A (8810), TS: 0x0438 (1080), Original Netw: 0x0001 (1)       |
|  Service name: CNEWS, provider: CSAT                                        |
|  Service type: 0x19 (Advanced codec HD digital television service)          |
|  TS packets: 9,298, PID's: 4 (clear: 4, scrambled: 0)                       |
|  PMT PID: 0x03E8 (1000), PCR PID: 0x03F2 (1010)                             |
|-----------------------------------------------------------------------------|
|     PID  Usage                                        Access          Bitrate  |
|   Total  Advanced codec HD digital television service . C    3,983,427 b/s  |
|  0x03E8  PMT ..................................... C       14,994 b/s  |
|  0x03F2  AVC video (1920x1080, high profile, level 4.0  C    3,771,361 b/s  |
|  0x03FD  AC-3 Audio (fra, AC-3, stereo (L,R), @48,000   C      197,072 b/s  |
|  0x0413  Subtitles (fra, DVB subtitles for hard of hea  C            0 b/s  |
|          (C=Clear, S=Scrambled, +=Shared)                                   |
|=============================================================================|
```

The first step is performed using the following command.

```
tsp -I ... \
    -P encap -o 0x1000 -p 0x03E8 -p 0x03F2 -p 0x03FD -p 0x0413 \
    -P pat --remove-service 0x226A \
    -P sdt --remove-service 0x226A \
    -P pmt -s 0x2262 -a 0x1000/0x99 \
    -O ...
```

The plugin *encap* creates a tunnel PID `0x1000`. This tunnel contains the 4 PID's of the service CNEWS, PMT, video, audio and subtitles. The plugins *pat* and *sdt* remove the service from the PAT and SDT, respectively. At this point, the service CNEWS has disappeared but the new PID `0x1000` is orphan. To preserve this PID through the processing of the service CANAL+ DECALE, we add it to the PMT of the service using the plugin *pmt*. We use the reserve stream type `0x99` for this PID to indicate some private type (any other reserved stream type should be OK).

After processing the SPTS through the external equipment (and hoping that the equipment has preserved the private components of the service), we restore the structure of the transport stream using the following command:

```
tsp -I ... \
    -P decap -p 0x1000 \
    -P pat -a 0x226A/0x03E8 \
    -P sdt -s 0x226A -n CNEWS -p CSAT \
    -P pmt -s 0x2262 -r 0x1000 \
    -O ...
```

The plugin *decap* decapsulates the content of the tunnel PID 0x1000. It is replaced by all original PID's of the service CNEWS. Then, we need to restore the reference to the service in the PAT and SDT using the plugins *pat* and *sdt*. Finally, since the private component 0x1000 no longer exists, we remove it from the PMT of the service CANAL+ DECALE using the plugin *pmt*.

Note: in a real-life example, the MPTS would probably contain more than two services. In that case, we would encapsulate all other services in the private tunnel PID as well. The principle remains the same. The command is only a bit longer.

5.2.26 Interleaving input files and merging their PSI

This example command reads two input files *file1.ts* and *file2.ts*. We assume here that the two files have no conflicting allocation of service ids and no conflicting PID's into services. Only the PSI/SI PID's are common.

Using the option `--interleave` in the *file* input plugin, we read one packet of *file1.ts*, then one packet of *file2.ts*, then one packet of *file1.ts again*, and so on. So, the two input files are multiplexed one by one.

Using the option `--label-base`, we assign the label 1 to all packets from *file1.ts* and label 2 to all packets from *file2.ts*.

```
tsp -I file --interleave file1.ts file2.ts --label-base 1 \
    -P psimerge --main-label 1 --merge-label 2 \
    ...
```

The plugin *psimerge* uses all packets with label 1 as a "main" stream and all packets with label 2 as a "merge" stream. It merges the PSI/SI from the two streams, creating a PAT and SDT-Actual containing all services from the two streams, creating a CAT containing all EMM PID's from the two streams. All EIT sections are multiplexed into the EIT PID.

6 Hardware Device Support

6.1 DVB Receiver Devices

6.1.1 Overview

The DVB receiver devices are specialized hardware devices which receive DVB-T, DVB-S, DVB-C, DVB-H or ATSC signals and transmit the demodulated binary transport stream to the computer system.

The input of a DVB receiver device is the antenna cable. The receiver device has either an F-connector (DVB-S, DVB-C) or a standard TV connector (DVB-T, DVB-C).

Most DVB-T receivers come with a small linear antenna. The usage of such an antenna should be avoided when possible since the reception is usually very poor. Always use the signal coming from a classical roof TV antenna when available (wall TV socket).

The physical output of a DVB receiver is a standard PC bus: PCI, USB, PCMCIA (PC Card) or Express Card. Some PCI devices are actually composed of one or more USB receivers and a USB-to-PCI bridge.

Most DVB receivers simply contain a tuner and a demodulator. They transmit the complete transport stream over the bus (PCI, USB, etc.) The demultiplexing and MPEG audio / video decoding is performed by some software, either in the kernel of the operating system or in a user-space application. Since TSDuck works on transport streams, the embedded hardware demux are never used. So, the simplest and cheapest receivers are usually fine for TSDuck.

Some DVB receivers contain two tuners in order to receive two independent transport streams. They usually appear as two distinct devices in the operating system.

Some recent DVB receivers support multiple protocols, for instance both DVB-T and DVB-C or both DVB-S and DVB-S2. This type of adapters is currently not properly supported by TSDuck.

6.1.2 Operating System Integration

6.1.2.1 Linux Platforms

The DVB receiver devices are managed by Linux under a common DVB framework.

Drivers:

The drivers for the DVB receiver devices come with the Linux kernel.

The drivers for recent devices may not be integrated yet into the mainstream Linux kernel, see [27] for details on how to install the latest Linux drivers for DVB devices.

Firmware:

Some devices need a firmware file in `/lib/firmware` which is loaded by the driver when the system boots or when the device is plugged-in (USB device for instance).

Some firmware files are packaged with the Linux kernel, but only when no copyright applies. Most firmware files are extracted from the proprietary Windows drivers of the device and are not free. Consequently, they are not included in the kernel distributions. Such proprietary firmware files must be fetched from various sites all over the Web.

Device naming:

The DVB devices are identified as `/dev/dvb/adapter`*N*, where *N* is a number between 0 and the number of DVB adapters in the system.

When several DVB devices are present in the system, the allocation of the adapter numbers depends on the kernel initialization sequence, the PCI slots, the way the USB devices are plugged and unplugged. It is possible to assign a specific adapter number to each device using the `adapter_nr` parameter in the relevant drivers (kernel modules) configuration.

For instance, let's take the example of a system with a Hauppauge WinTV Nova-T-500 (dual DVB-T tuner) and a Hauppauge WinTV Nova-HD-S2 (DVB-S/S2 tuner). The two tuners in the DVB-T PCI board are actually USB devices with an embedded USB hub and the numbering of the tuners is not deterministic. The adapter number for each tuner may vary after each boot. To always allocate adapter numbers 0 and 1 to the DVB-T dual tuner and adapter number 2 to the DVB-S tuner, add the following lines to a *modprobe* configuration file, for instance `/etc/modprobe.d/local.conf`:

```
options dvb-usb-dib0700 adapter_nr=0,1
options cx88-dvb adapter_nr=2
```

Then, the following allocation is always used:

```
$ tslsdvb
/dev/dvb/adapter0 (DiBcom 3000MC/P, DVB-T)
/dev/dvb/adapter1 (DiBcom 3000MC/P, DVB-T)
/dev/dvb/adapter2 (Conexant CX24116/CX24118, DVB-S)
```

6.1.2.2 Microsoft Windows Platforms

DirectShow framework:

On Windows XP and higher, the DVB devices are managed by "*DirectShow*", a Microsoft framework for multimedia. The specific subsystem of DirectShow for DVB receiver devices is BDA (*Broadcast Device Architecture*). Most of the time, the hardware vendors provide BDA drivers for their receivers. Windows does not include any predefined BDA driver.

On Windows Vista, a new "*Media Foundation*" framework has been introduced by Microsoft. On the long term, Media Foundation is supposed to supersede DirectShow but its current features are reputed to be inferior. DirectShow is still present on Windows 7 and 10 and is supposed to remain on subsequent versions of Windows.

On all Windows platforms, TSDuck uses basic DirectShow features to access the BDA drivers of the receiver devices.

DVB-S2 support:

Microsoft DirectShow implements DVB-S2 on Windows 7 and higher only. It is not possible to use DVB-S2 tuners on Windows XP or Vista.

DiSEqC support:

There is no standard support for DiSEqC with DVB-S/S2 tuners in the BDA architecture, which makes Windows useless when capturing behind a DiSEqC switch with multiple dishes.

Note that almost every driver provides a non-standard, non-documented and vendor-specific API to select a DiSEqC port but this usually works only with vendor-specific software, like TV viewing applications which are provided with the tuner device.

Since TSDuck only uses the standard BDA interfaces on Windows systems, it is not possible to select a DiSEqC port other than zero (option `--satellite-number` in *tsp* plugin `dvb` has usually no effect). If the tuner is connected to a DiSEqC switch, capturing on the first DiSEqC port (satellite number zero) usually works.

Retrieving actual modulation parameters:

On Windows, it is not possible to retrieve the actual tuning parameters of a transport stream as detected by the tuner device.

This can be annoying in a DVB-T environment where many transmission parameters may be inaccurate but the tuner device will detect the actual parameters. For instance, you may tune on a transport specifying a FEC 2/3 and a guard interval 1/32. If the actual signal uses a FEC 3/4 and a guard interval 1/8, the tuner device will automatically adjust the parameters. On Linux, the command "`tslsdvb -v`" displays the actual parameters, as reported by the tuner device. Moreover, the `dvb` plugin can compute the exact theoretical bitrate of the transport stream based on the actual transmission parameters. On Windows, it is not possible to query the tuner device for the actual parameters. It is not possible to

display the actual transmission parameters. The `dvb` plugin must use the analysis of PCR's to evaluate the bitrate.

32 vs. 64 bits:

TSDuck for Windows is available in two versions, 32 and 64 bits. On Windows 64 bits, the two versions can be used. If you use DVB tuners, carefully check the provided drivers and DirectShow filters. Some DVB tuners provide 32-bit filters only. In that case, you must use the 32-bit version of TSDuck. The 64-bit version of TSDuck will not work with 32-bit DirectShow filters.

6.1.2.3 MacOS Platforms

There is no uniform or standard software framework to support DVB tuners on macOS. Some tuners are officially supported on macOS but they are shipped with proprietary drivers and proprietary TV-watching applications. The driver API's are not documented.

As a result, TSDuck provides no support for DVB tuners on macOS.

6.1.3 Device Naming

All TSDuck modules using DVB receivers (`tslsdvb`, `tsscan`, `dvb` plugin) use a "*device name*" to designate a DVB receiver device. The syntax of the device name depends on the operating system.

- On Linux, a receiver device is named as `/dev/dvb/adapter`*A*`[:`*F*`[:`*M*`[:`*V*`]]]` where:

  A = adapter number

  F = frontend number (default: `0`).

  M = demux number (default: `0`).

  V = dvr number (default: `0`).

  Only the adapter number is important if there is more than one DVB receiver device in the system. There is usually no good reason to specify non-zero frontend, demux and dvr.

- On Windows, a receiver device name is the name of a DirectShow tuner filter. Since these names are usually complicated, with spaces and mixed cases ("*Nova-T Stick DVB-T Tuner (Dev1 Path0)*" for instance), the specified name is not case sensitive and spaces are ignored. As an alternative, the name "`:`*N*" can be used to designate the *N*th receiver device in the system, the first index being zero.

Use the `tslsdvb` utility to list all available DVB receiver devices. By default, when no device name is specified, the "*first*" DVB receiver device is used, that is the say the device which appears first when the command "`tslsdvb`" is invoked.

In all cases (`tslsdvb`, `tsscan`, `dvb` plugin), the option `--adapter` (or `-a`) can be used to simply designate the *N*th receiver device in the system, the first index being zero. When the system has several receivers devices, `tslsdvb` also displays the corresponding device index.

6.1.4 Tested Devices

On Linux, TSDuck works indifferently with any supported DVB device. If a driver exists (with optional firmware) for a given DVB receiver, it should work with TSDuck.

On Windows, TSDuck should work with any DVB receiver coming with a BDA driver but the integration is less straightforward than on Linux and additional testing should be performed. Typically, if the device comes with a "DVB Network Tuner" DirectShow filter and an optional "BDA Receiver Component" DirectShow filter, it should work with TSDuck. At least one device (one from TechniSat) has exhibited different software architecture and could not be used by TSDuck.

The following table summarizes the DVB receiver devices which have been tested with TSDuck.

Please note that this table is informational only. It was built from various users' feedback at some point in time. There is no exhaustive test suite using all these devices. Probably no one, neither the author of TSDuck nor any of its users, have all these devices. So, keep in mind that these devices are not tested for every new version of TSDuck.

Table 6: Tested DVB receiver devices

Brand	Model	DVB	# (1)	Bus	Linux	Windows
BlackGold	BGT3620	DVB-T2/C	6	PCIe	Not tested	Tested OK
DVBSky	S960	DVB-S/S2	1	USB	Tested OK	Tested OK
DVBSky	S960C (2)	DVB-S/S2	1	USB	Tested OK	Tested OK
GoTView	MasterHD3	DVB-T2/C	2 (3)	USB	Tested OK (4)	Tested OK
Hauppauge	WinTV Nova-T-500 (5)	DVB-T	2	PCI	Tested OK (6)	Not tested
Hauppauge	WinTV Nova-TD-500 (7)	DVB-T	2	PCI	Tested OK (6, 8)	Not tested
Hauppauge	WinTV Nova-T-Stick (9)	DVB-T	1	USB	Tested OK (6, 10)	Tested OK (11)
Hauppauge	WinTV Nova-T-Stick SE	DVB-T	1	USB	Tested OK (6, 12)	Tested OK (12)
Hauppauge	WinTV Nova-S	DVB-S	1	PCI	Tested OK	Not tested
Hauppauge	WinTV Nova-HD-S2 (13)	DVB-S/S2	1	PCI	Tested OK (14)	Tested OK
Hauppauge	WinTV-soloHD	DVB-T2/C	2	USB	Not tested	Tested OK (15)
MaxMedia	HU 372 (23)	DVB-T2/C	2 (3)	USB	Tested OK (4)	Tested OK
Pinnacle	PCTV DVB-T Stick 72e	DVB-T	1	USB	Tested OK (6)	Tested OK
Pinnacle	PCTV nanoStick T2 290e	DVB-T2/C	2	USB	Tested OK (16)	Tested OK
Pinnacle	PCTV DVB-S2 Stick 461e	DVB-S/S2	1	USB	Not working (17, 18)	Tested OK (19)
TBS	TBS 6284	DVB-T/T2	4	PCIe	Not tested	Tested OK
TBS	TBS 6903	DVB-S/S2	2	PCIe	Not tested	Tested OK
TBS	TBS 5922	DVB-S/S2	1	USB	Not tested	Tested OK
TBS	TBS 5925	DVB-S/S2	1	USB	Not tested	Tested OK
TBS	TBS 6704	ATSC	4	PCIe	Tested OK	Not tested
TBS	TBS 6904	DVB-S/S2	4	PCIe	Tested OK	Not tested
TechniSat	SkyStar USB HD	DVB-S/S2	1	USB	Not tested	Not working (20)
TechnoTrend	TT-connect S2-3600	DVB-S/S2	1	USB	Not tested	Tested OK (21)
TechnoTrend	TT-connect S2-4600	DVB-S/S2	1	USB	Not tested	Tested OK
TechnoTrend	TT-budget S2-4100	DVB-S/S2	1	PCIe	Not tested	Tested OK
Terratec	Cinergy T USB XE Rev 2 (22)	DVB-T	1	USB	Tested OK (24)	Tested OK
TeVii	H640 (23)	DVB-T2/C	2 (3)	USB	Tested OK (4)	Tested OK
TeVii	S482 DVB-S2	DVB-S/S2	2	PCIe	Not tested	Tested OK

Notes from the table:

1. Number of tuners. When more than one is present, they usually appear as different receiver devices in the operating system.
2. The DVBSky S960C has a DVB-CI CAM slot (not CI+).
3. The GoTView MasterHD3 has two demodulators, one for DVB-T and one for DVB-T2/C. On Windows, they appear as one single DVB-T tuner. On Linux, they appear as two frontends, one for DVB-T and one for DVB-T2/C.
4. With Linux kernels 4.2 up to 4.7, two frontends are available: frontend0 is DVB-T, frontend1 is DVB-T2/DVB-C. The support in kernels after version 4.7 is partial, something was broken. The device starts but only with the one (DVB-T) frontend. The second frontend (Si2168 demodulator for DVB-T2 and DVB-C) doesn't start due to i2c error.
5. The Hauppauge WinTV Nova-T-500 is a PCI board which embeds two USB tuners and a USB-to-PCI bridge.

6. Need the firmware file revision 1.20 for DiBcom-based DVB receiver devices on Linux, http://www.wi-bw.tfh-wildau.de/~pboettch/home/files/dvb-usb-dib0700-1.20.fw
7. The Nova-TD-500 is similar to the Nova-T-500 but has two aerial inputs instead of one.
8. Do not plug antenna cables in both aerial inputs, this leads to garbage reception. Use only the top aerial input and this feeds the two tuners. The bottom aerial input is not used. Also specify the following options in `/etc/modprobe.d/options`:

   ```
   options dvb_usb_dib0700 force_lna_activation=1
   options dvb_usb disable_rc_polling=1
   ```

9. Two different revisions exist: 70001 and 70009 (read the sticker).
10. Revision 70001 tested, works OK. Revision 70009 not tested.
11. Revision 70001 tested, works OK with the Hauppauge driver CD version 2.5E but does not work with recent drivers versions 3.x and 4.x. Revision 70009 not tested (requires drivers CD version 4.x).
12. Model 203, revision D1F4 70019 tested.
13. This is a “lite” version of the Hauppauge HVR-4000.
14. Need the `dvb-fe-cx24116.fw` firmware file. Known limitation: Some PCI DMA transfers are aborted without known reason, resulting in packet loss. The problem appears only on some hardware systems and may be related to PCI bus configuration. The problem is characterized by the following error messages from `dmesg`:

    ```
    cx88[0]: irq mpeg  [0x80000] pci_abort*
    cx88[0]/2-mpeg: general errors: 0x00080000
    ```

15. On Windows, the Hauppauge software installation is incomplete. After installing the drivers, the WinTV-soloHD initially appears as one single DVB-T tuner. DVB-C is not accessible. The bundled application WinTV must be run at least once and tuned to a DVB-C transport. Afterwards, a second tuner is installed for DVB-C. This tuner is persistent after reboots.
16. Need the firmware file `dvb-demod-si2168-b40-01.fw`.
17. Need the firmware file for Montage M88DS3103-based DVB receiver devices on Linux from the OpenELEC dvb-firmware package.
    https://github.com/OpenELEC/dvb-firmware/blob/master/firmware/dvb-demod-m88ds3103.fw
18. Documented to work on Linux. But the experience demonstrates that it is mostly unreliable. The first tuning operation after insertion of the USB device works. Subsequent tuning operations fail.
19. On Windows, it has been observed that the PCTV 461e discards all null packets (PID 0x1FFF). As a consequence, transport stream analyses are incorrect, bitrates are incorrect and all *tsp* plugins which use stuffing to insert new packets do not work correctly.
20. The TechniSat drivers for Windows have a proprietary and unusual interface. They cannot be integrated in a DirectShow reception graph and, consequently, cannot be used by TSDuck.
21. DVB tuners drivers for Windows: http://www.tt-pc.com/2959/PC_Products.html
22. Two different revisions exist: Rev 1 and Rev 2. They use different chipsets and need different drivers. Only the Rev 2 has been tested with TSDuck.
23. Reported as identical to GoTView Master HD3.
24. Need the firmware file for Afatech-based DVB receiver devices on Linux, http://www.otit.fi/~crope/v4l-dvb/af9015/af9015_firmware_cutter/firmware_files/4.95.0/dvb-usb-af9015.fw

6.2 Dektec Devices

6.2.1 Overview

The Dektec devices include a wide range of professional MPEG/DVB devices: ASI input or output, modulators (QPSK, QAM, OFDM, ATSC, DMB, ISDB, etc) and IP multicasting. The PCI devices are named DTA-1xx and the USB devices are named DTU-2xx. The ASI devices can perform either input, output or both. Modulators are output-only, obviously. See [22] for more details.

The tsp plugin named `dektec` can perform input or output on any Dektec device, provided that the appropriate drivers are installed on the system. Dektec provides drivers and API for their devices on Windows and Linux (see [23]). For each operating system, there are two Dektec drivers: one for all PCI devices and one for all USB devices.

6.2.2 Linux Platforms

The Dektec drivers are provided in source format. They must be compiled for each specific version of the Linux kernel.

For a better integration with the various distros, an independent project has been setup to create DKMS packages for Dektec drivers (see [24]). This project provides a script to build packages for Red Hat, CentOS, Fedora and Ubuntu distros, using the source code from the Dektec site. Pre-built packages are also available from the *releases* section in [24].

6.2.3 Microsoft Windows Platforms

The Dektec drivers are provided in binary format and can be directly installed. An installation guide is included in the zip file of each driver. See [23].

6.2.4 MacOS Platforms

Dektec provides no support for macOS. All Dektec features of TSDuck are disabled on macOS.

6.2.5 Tested Devices

The following Dektec devices have been successfully tested with TSDuck:

- DTA-140 : PCI ASI input and output.
- DTU-245 : USB ASI input and output.
- DTA-107 : PCI DVB-S modulator.
- DTA-107S2 : PCI DVB-S2 modulator.
- DTA-110T : PCI DVB-T modulator.
- DTA-115 : PCI multi-standard modulator (some modulation types are subject to optional licences) with an additional bidirectional ASI port.
- DTU-315 : USB-3 multi-standard modulator (subject to optional licences).
- DTA-2137C : PCIe DVB-S/S2 demodulator with ASI outputs.
- DTA-2138B : PCIe DVB-T/T2, DVB-C/C2, ISDB-T demodulator.

Any other Dektec device should work with TSDuck.

6.3 HiDes Devices

6.3.1 Overview

HiDes is a company from Taiwan, a manufacturer of cheap DVB-T devices (see [25]). These devices are based on chips from ITE Technologies Inc., also from Taiwan.

The UT-100C model is a USB DVB-T modulator adaptor (transmission). This device is probably the cheapest modulator on Earth for Digital TV.

Other models from HiDes include reception, ISDB-T support or PCIe interface. Currently, only USB DVB-T modulators are supported by TSDuck.

The tsp plugin named `hides` can perform output on HiDes devices, provided that the appropriate drivers are installed on the system. These drivers are available at [26].

6.3.2 Linux Platforms

The drivers for HiDes devices are provided in source format. It is unclear if these drivers were provided by HiDes or ITE. They must be compiled for each specific version of the Linux kernel.

For a better integration with the various distros, an independent project has been setup to create DKMS packages for HiDes drivers (see [26]). This project provides a script to build packages for Red Hat, CentOS, Fedora and Ubuntu distros. Pre-built packages are also available from the *releases* section in [26].

The name of a HiDes device is illustrated below:

```
$ tshides -v
Found 1 HiDes device

Index ........... 0
Name ............ "usb-it950x0"
Device .......... /dev/usb-it950x0
Chip type ....... 0x9507
Device type ..... 11
Driver version .. v16.11.10.1w
API version ..... 1.3.20160929.0
Link firmware ... 255.39.2.0
OFDM firmware ... 255.9.11.0
Company ......... ITEtech
Hardware info ... Eagle DVBT
```

Note the 'w' at the end of the driver version. This indicates a modified "waiting" version of the driver as provided in [26].

The original driver from HiDes or ITE has a "polling" design which is much less efficient. If you have a driver version without trailing 'w', this is probably an original version of the driver. TSDuck will work but in a very inefficient way : each time packets shall be sent to the modulator, the *tsp* application has to actively wait (looping on very short timers) for the modulator to be ready, unnecessarily consuming CPU and lacking accuracy. With the modified 'w' version, the output thread of the *tsp* application is simply suspended until the very precise moment where the modulator is ready.

6.3.3 Microsoft Windows Platforms

The HiDes driver is provided in binary format and can be directly installed. The installer is in a zip file.

There is no known fixed reference URL for the latest version of the Windows driver. To make sure that TSDuck users can always find a working version of this driver, it is also available from the *releases* section in [26].

Unlike the Linux driver, the Windows driver has not been modified for TSDuck. The original driver is anyway delivered in binary form and cannot be easily modified. Note that the original Windows driver has a standard "waiting" design and does not suffer from the "polling" design of the original Linux driver.

The name of a HiDes device is a DirectShow filter name, as illustrated below:

```
C:\> tshides -v
Found 1 HiDes device

Index ........... 0
Name ............ "IT9507 TX Filter"
Device .......... \\?\usb#vid_048d&pid_9507#ut100cv4201504240422#{fbf6f530-07b9-11d2-
a71e-0000f8004788}\{9963cc0e-ee70-11e0-ba8f-92d34824019b}
USB mode ........ 0x0200
Vendor id ....... 0x048D
Product id ...... 0x9507
Chip type ....... 0x9507
```

```
Device type ..... 11
Driver version .. 21.17.39.1
Link firmware ... 255.39.2.0
OFDM firmware ... 255.9.11.0
```

Identical devices use the same DirectShow filter and have probably identical names. The device path is unique but is a complicated Windows device reference and is barely usable. So, when we have several identical HiDes devices on the same machine, it is probably easier to reference them by adapter index (0, 1, 2, etc.) using option `--adapter`.

Note that the verbose display (option `-v`) is different between Windows and Linux. This is due to the distinct API's of the HiDes drivers on distinct operating system. The command *tshides* displays what is available for the platform it is running on.

6.3.4 MacOS Platforms

HiDes provides no support for macOS. All HiDes features of TSDuck are disabled on macOS.

6.3.5 Tested Devices

The following HiDes devices have been successfully tested with TSDuck:

- UT-100C : USB DVB-T modulator.
- UT-100A : USB DVB-T receptor and modulator. Only the modulator is supported with TSDuck.

6.3.6 Power Constraints

The HiDes devices have no external power. They are exclusively powered through the USB port. It has been reported that some USB ports did not provide sufficient power to the device, resulting in random corruptions in the output stream.

In case of problem, try to connect the HiDes device to a powered USB 3.0 hub.

In [25], the HiDes documentation states that the maximum required power is 390 mA. But it is currently unclear if the HiDes device requires more than the normalized maximum of 500 mA from the USB port or if some USB ports fail to provide the required 500 mA.

Appendix A TSDuck Configuration File

The TSDuck configuration file is used to specify default command line options or alternate options for all or selected TSDuck commands. This configuration file is specific per user.

The location of the user's TSDuck configuration file depends on the operating system.

- Unix : `$HOME/.tsduck`
- Windows : `%APPDATA%\tsduck\tsduck.ini`

The format of this file resembles the old ".INI" files on Windows systems. There is one main section, followed by several command-specific sections.

Here are the main rules for the configuration files:

- The main section comes first.
- A section starts with the section name enclosed in square brackets.
- The name of a section is the name of a TSDuck command (e.g. `[tsp]`, `[tsswitch]`, etc.)
- In a section, an entry has the syntax "*name = value*".
- An entry can be specified several times in a section when multiple values are allowed.
- When a TSDuck command searches for an entry in the configuration file, it searches first in the section with the name of the command. If the entry is not found here, it is searched in the top main section.
- Lines starting with a dieresis (#) are comments and are ignored.
- Lines ending with a back-slash are continued on the next text line.
- Quotes can be used to group command line arguments when necessary.
- Back-slashes in the middle of a line are used to escape characters.

The following table lists the supported entries in the configuration file.

Table 7: Configuration file entries

Entry name	Description	Apply to
`default.options`	Used as command line options when none are specified.	All commands
`prepend.options`	Options to prepend before the actual options.	All commands
`append.options`	Options to append after the actual options.	All commands
`default.input`	Default input plugin (with options) when none are specified.	`tsp`, `tsswitch`
`default.plugin`	Default packet processing plugin (with options) when none are specified.	`tsp`
`default.output`	Default output plugin (with options) when none are specified.	`tsp`, `tsswitch`
`default.region`	Default region for UHF / VHF frequency layout [8].	All commands

The following example configuration file illustrates most entries.

```
# Sample configuration file
prepend.options = --verbose

[tsversion]
default.options = --all

[tsp]
default.input  = file '/home/john doe/name with spaces \' and quotes\'.ts'
default.plugin = until --packet 1,000,000
default.plugin = analyze
default.output = drop
```

[8] A list of all supported regions can be found in the file named *tsduck.hfbands.xml* in the same directory as all TSDuck binaries, commands and plugins.

In this case, when the command *tsp* is used alone without arguments, the actual command will be:

```
tsp --verbose \
    -I file '/home/john doe/name with spaces \' and quotes\'.ts' \
    -P until --packet 1,000,000 \
    -P analyze \
    -O drop
```

If one type of plugin is specified, the defaults no longer apply. For instance, the command:

```
tsp -P regulate
```

will become:

```
tsp --verbose \
    -I file '/home/john doe/name with spaces \' and quotes\'.ts' \
    -P regulate \
    -O drop
```

Appendix B Channel Configuration XML Reference Model

This appendix describes the XML reference format for the channel configuration file.

B.1 File usage

A channel configuration file is an XML file containing the description of TV channels (networks, transport streams, services). This file is typically used to simplify the specification of tuning parameters on the command line. Instead of specifying all tuning parameters, use the option `--channel-transponder` followed by a service name. The TSDuck command retrieves the characteristics of the transport stream which contains the specified service and automatically uses its tuning parameters.

If the same service appears in networks of different types (for instance a DVB-S and a DVB-T network), the TSDuck command will use the one in the same type of network as the hardware tuner in use.

There is one default file per user but any other file can be specified using the option `--tuning-file`. The location of the user's default file depends on the operating system.

- Unix : `$HOME/.tsduck.channels.xml`
- Windows : `%APPDATA%\tsduck\channels.xml`

B.2 Channel configuration file format

B.2.1 Conventions

The format which is used here is informal. It shall be considered as a template. It does not conform to any formal specification such as XML-Schema.

All allowed nodes and attributes are present in the template. The contents of attributes in this template are comments describing the expected content of the corresponding attribute in real XML files. The values of these attributes in the template are descriptive only; they would be invalid if directly used in input XML files for TSDuck.

Notes on types and formats:

- Tags and attributes are not case-sensitive.
- Integer values can be represented in decimal or hexadecimal (`0x` prefix).
- Booleans are "`true`" or "`false`".
- Some attributes accept symbols in addition to plain numerical values. The names of accepted symbols are listed in the attribute. Example:

  ```
  type="ATSC|DVB-C|DVB-S|DVB-T"
  ```

B.2.2 XML file structure

An XML file describing channels for TSDuck uses `<tsduck>` as root node. The root node contains any number of networks which, in turn contain transport streams and services.

All XML files shall be encoded in UTF-8 format to allow international character sets in service names for instance. The initial declaration line *"<?xml version="1.0" encoding="UTF-8"?>"* is optional but recommended.

The global structure of the XML channel configuration file is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<tsduck>

  <!-- Several networks in an XML file -->
  <!-- The type attribute must match the tuner type to locate a service -->
  <network id="uint16, required" type="ATSC|DVB-C|DVB-S|DVB-T, required">

    <!-- Several transport streams in a network -->
    <ts id="uint16, required" onid="uint16, optional">
```

```
        <!-- Tuning information: exactly one of atsc, dvbc, dvbs, dvbt -->
        <!-- Must match the network type -->

        <!-- Several services in the TS -->
        <service id="uint16, required"
                 name="string, optional"
                 provider="string, optional"
                 LCN="uint16, optional"
                 PMTPID="uint13, optional"
                 type="uint8, optional"
                 cas="bool, optional"
                 atsc_type="uint6, optional"
                 atsc_major_id="uint10, optional"
                 atsc_minor_id="uint10, optional"/>
    </ts>
  </network>

</tsduck>
```

B.3 Tuning parameters

There must be exactly one tuning parameter structure per transport stream description (`<ts>` structure).

B.3.1 ATSC

To be used for ATSC tuners:

```
<atsc frequency="uint64, required"
      modulation="8-VSB|16-VSB, default=8-VSB"
      inversion="on|off|auto, default=auto"/>
```

B.3.2 DVB-C

To be used for DVB-C/C2 tuners:

```
<dvbc frequency="uint64, required"
      symbolrate="uint32, default=6,900,000"
      modulation="16-QAM|32-QAM|64-QAM|128-QAM|256-QAM|QAM, default=64-QAM"
      FEC="1/2|2/3|3/4|4/5|5/6|6/7|7/8|8/9|9/10|3/5|1/3|1/4|2/5|5/11|auto|none,
          default=auto"
      inversion="on|off|auto, default=auto"/>
```

B.3.3 DVB-S

To be used for DVB-S/S2 tuners:

```
<dvbs satellite="uint2, default=0"
      orbital="string, optional"
      frequency="uint64, required"
      symbolrate="uint32, default=27,500,000"
      modulation="QPSK|8-PSK|16-APSK|32-APSK, default=QPSK"
      system="DVB-S|DVB-S2, default=DVB-S"
      polarity="horizontal|vertical|left|right|auto|none, default=auto"
      inversion="on|off|auto, default=auto"
      FEC="1/2|2/3|3/4|4/5|5/6|6/7|7/8|8/9|9/10|3/5|1/3|1/4|2/5|5/11|auto|none,
           default=auto"
      pilots="on|off|auto, default=auto"
      rolloff="0.20|0.25|0.35|auto, default=auto"
      ISI="uint8, optional"
      PLS_code="uint18, optional"
      PLS_mode="ROOT|GOLD|COMBO, default=ROOT"/>
```

B.3.4 DVB-T

To be used for DVB-T/T2 tuners:

```
<dvbt frequency="uint64, required"
```

```
modulation="QPSK|16-QAM|64-QAM|256-QAM, default=64-QAM"
inversion="on|off|auto, default=auto"
HPFEC="1/2|2/3|3/4|4/5|5/6|6/7|7/8|8/9|9/10|3/5|1/3|1/4|2/5|5/11|auto|none,
       default=auto"
LPFEC="1/2|2/3|3/4|4/5|5/6|6/7|7/8|8/9|9/10|3/5|1/3|1/4|2/5|5/11|auto|none,
       default=auto"
bandwidth="1.712-MHz|5-MHz|6-MHz|7-MHz|8-MHz|10-MHz|auto, default=auto"
transmission="1K|2K|4K|8K|16K|32K|auto, default=auto"
guard="1/4|1/8|1/16|1/32|1/128|19/128|19/256|auto, default=auto"
hierarchy="1|2|4|auto|none, default=auto"
PLP="uint8, optional"/>
```

Appendix C PSI/SI XML Reference Model

This appendix describes the XML reference format for all tables and descriptors.

C.1 PSI/SI file format

C.1.1 Conventions

The format which is used here is informal. It shall be considered as a template. It does not conform to any formal specification such as XML-Schema.

All allowed nodes and attributes are present in the template. The contents of attributes in this template are comments describing the expected content of the corresponding attribute in real XML files. The values of these attributes in the template are descriptive only; they would be invalid if directly used in input XML files for TSDuck.

Notes on types and formats:

- Tags and attributes are not case-sensitive.
- The names of tags and attributes are copied from ISO or DVB standards.
- Integer values can be represented in decimal or hexadecimal (`0x` prefix).
- Booleans are "`true`" or "`false`".
- Some attributes accept symbols in addition to plain numerical values. The names of accepted symbols are listed in the attribute. Example:

  ```
  running_status="undefined|not-running|starting|pausing|running|off-air"
  ```

- Hexadecimal content is a suite of hexadecimal digits. Spaces are ignored. Note that the name *hexadecimal content* is used for data blocks, usually private ones, of arbitrary length. This is different from integer values in attributes which can be represented as hexadecimal using the prefix `0x`. In *hexadecimal content* blocks, there is no `0x` prefix, everything is hexadecimal.
- The pseudo-node *<DESCRIPTOR_LIST>* is a place-holder for a sequence a descriptor nodes.
- Unsupported tables and descriptors can still be used. Their payloads must be specified as hexadecimal content. See tags `<generic_short_table>`, `<generic_long_table>` and `<generic_descriptor>` in section C.11, page 325.

C.1.2 XML file structure

An XML file containing PSI/SI tables for TSDuck uses `<tsduck>` as root node. The root node contains any number of tables.

All XML files shall be encoded in UTF-8 format to allow international character sets in service names or event descriptions for instance. The initial declaration line "*<?xml version="1.0" encoding="UTF-8"?>*" is optional but recommended.

```
<?xml version="1.0" encoding="UTF-8"?>
<tsduck>
  <!-- any number of table structures -->
</tsduck>
```

C.2 MPEG-defined tables

C.2.1 Conditional Access Table (CAT)

Defined by MPEG in [1].

```
<CAT version="uint5, default=0" current="bool, default=true">
  <DESCRIPTOR_LIST>
</CAT>
```

C.2.2 DSM-CC Stream Descriptors Table

Defined by MPEG in [3].

```
<DSMCC_stream_descriptors_table
    version="uint5, default=0"
    current="bool, default=true"
    table_id_extension="uint16, default=0xFFFF">
  <DESCRIPTOR_LIST>
</DSMCC_stream_descriptors_table>
```

C.2.3 Program Association Table (PAT)

Defined by MPEG in [1].

```
<PAT version="uint5, default=0"
     current="bool, default=true"
     transport_stream_id="uint16, required"
     network_PID="uint13, optional">

  <!-- One per service -->
  <service service_id="uint16, required" program_map_PID="uint13, required"/>

</PAT>
```

C.2.4 Program Map Table (PMT)

Defined by MPEG in [1].

```
<PMT version="uint5, default=0"
     current="bool, default=true"
     service_id="uint16, required"
     PCR_PID="uint13, default=0x1FFF">

  <!-- Program-level descriptors -->
  <DESCRIPTOR_LIST>

  <!-- One per elementary stream -->
  <component stream_type="uint8, required" elementary_PID="uint13, required">
    <DESCRIPTOR_LIST>
  </component>

</PMT>
```

C.2.5 Transport Stream Description Table (TSDT)

Defined by MPEG in [1].

```
<TSDT version="uint5, default=0" current="bool, default=true">
  <DESCRIPTOR_LIST>
</TSDT>
```

C.3 DVB-defined tables

C.3.1 Application Information Table (AIT)

Defined by DVB in [12] and [15].

```
<AIT version="uint5, default=0"
     current="bool, default=true"
     test_application_flag="bool, default=true"
     application_type="uint15, required">

  <!-- Common descriptors loop -->
  <DESCRIPTOR_LIST>

  <!-- One per application -->
```

```
  <application control_code="uint8, required">
    <application_identifier
         organization_id="uint32, required"
         application_id="uint16, required"/>
    <DESCRIPTOR_LIST>
  </application>

</AIT>
```

C.3.2 Bouquet Association Table (BAT)

Defined by DVB in [6].

The optional attribute `preferred_section` indicates in which section the description of a transport stream should be preferably serialized. When unspecified for a TS, the corresponding TS description is serialized in an arbitrary section.

```
<BAT version="uint5, default=0"
     current="bool, default=true"
     bouquet_id="uint16, required">

  <!-- Bouquet-level descriptors -->
  <DESCRIPTOR_LIST>

  <!-- One per transport stream -->
  <transport_stream transport_stream_id="uint16, required"
                    original_network_id="uint16, required"
                    preferred_section="uint8, optional">
    <DESCRIPTOR_LIST>
  </transport_stream>

</BAT>
```

C.3.3 Discontinuity Information Table

Defined by DVB in [6].

```
<discontinuity_information_table transition="bool, required"/>
```

C.3.4 Event Information Table (EIT)

Defined by DVB in [6].

If `type="pf"`, this is an EITp/f (present/following).

If `type` is a 4-bit integer, this is an EITs (schedule) with TID `0x50` + type (EITs Actual) or `0x60` + type (EITs Other), depending on the `actual` attribute.

When an EIT is compiled by TSDuck (serialized as binary sections), the events are sorted in ascending order of start time and spread over sections as described in [7].

The attribute `last_table_id` is optional. By default, it is set to the same table id as the table. Upon serialization, the DVB rules are enforced to bind its value within the DVB-specified limits.

```
<EIT type="pf|uint4, default=pf"
     version="uint5, default=0"
     current="bool, default=true"
     actual="bool, default=true"
     service_id="uint16, required"
     transport_stream_id="uint16, required"
     original_network_id="uint16, required"
     last_table_id="uint8, default=same as table id">

  <!-- One per event -->
  <event event_id="uint16, required"
         start_time="YYYY-MM-DD hh:mm:ss, required"
         duration="hh:mm:ss, required"
         running_status="undefined|not-running|starting|pausing|running|off-air,
```

```
                          default=undefined"
          CA_mode="bool, default=false">
    <DESCRIPTOR_LIST>
  </event>

</EIT>
```

C.3.5 IP/MAC Notification Table (INT)

Defined by DVB in [14].

```
<INT version="uint5, default=0"
     current="bool, default=true"
     action_type="uint8, default=0x01"
     processing_order="uint8, default=0x00"
     platform_id="uint24, required">

  <!-- Plaform-level descriptors -->
  <DESCRIPTOR_LIST>

  <!-- One per device -->
  <device>
    <target>
      <DESCRIPTOR_LIST>
    </target>
    <operational>
      <DESCRIPTOR_LIST>
    </operational>
  </device>

</INT>
```

C.3.6 Network Information Table (NIT)

Defined by DVB in [6].

The optional attribute `preferred_section` indicates in which section the description of a transport stream should be preferably serialized. When unspecified for a TS, the corresponding TS description is serialized in an arbitrary section.

```
<NIT version="uint5, default=0"
     current="bool, default=true"
     network_id="uint16, required"
     actual="bool, default=true">

  <!-- Network-level descriptors -->
  <DESCRIPTOR_LIST>

  <!-- One per transport stream -->
  <transport_stream transport_stream_id="uint16, required"
                    original_network_id="uint16, required"
                    preferred_section="uint8, optional">
    <DESCRIPTOR_LIST>
  </transport_stream>

</NIT>
```

C.3.7 Running Status Table (RST)

Defined by DVB in [6].

```
<RST>
  <!-- One per event -->
  <event transport_stream_id="uint16, required"
         original_network_id="uint16, required"
         service_id="uint16, required"
         event_id="uint16, required"
```

```
          running_status="undefined|not-running|starting|pausing|running|off-air,
                          required"/>
</RST>
```

C.3.8 Selection Information Table

Defined by DVB in [6].

```
<selection_information_table version="uint5, default=0" current="bool, default=true">

  <!-- Common descriptors loop, for transmission parameters -->
  <DESCRIPTOR_LIST>

  <!-- One per service -->
  <service service_id="uint16, required"
           running_status="undefined|not-running|starting|pausing|running|off-air,
                           required">
    <DESCRIPTOR_LIST>
  </service>

</selection_information_table>
```

C.3.9 Service Description Table (SDT)

Defined by DVB in [6].

```
<SDT version="uint5, default=0"
     current="bool, default=true"
     transport_stream_id="uint16, required"
     original_network_id="uint16, required"
     actual="bool, default=true">

  <!-- One per service -->
  <service service_id="uint16, required"
           EIT_schedule="bool, default=false"
           EIT_present_following="bool, default=false"
           running_status="undefined|not-running|starting|pausing|running|
                           off-air, default=undefined"
           CA_mode="bool, default=false">
    <DESCRIPTOR_LIST>
  </service>

</SDT>
```

C.3.10 Time and Date Table (TDT)

Defined by DVB in [6].

```
<TDT UTC_time="YYYY-MM-DD hh:mm:ss, required"/>
```

C.3.11 Time Offset Table (TOT)

Defined by DVB in [6].

```
<TOT UTC_time="YYYY-MM-DD hh:mm:ss, required">
  <DESCRIPTOR_LIST>
</TOT>
```

C.4 SCTE-defined tables

C.4.1 Splice Information Table (SCTE 35)

Defined by ANSI/SCTE in [17].

Exactly one of the following tags is allowed. This is the splice command in this table.

```
<splice_null>
<splice_schedule>
```

```
<splice_insert>
<time_signal>
<bandwidth_reservation>
<private_command>
```

Table definition:

```
<splice_information_table
    protocol_version="uint8, default=0"
    pts_adjustment="uint33, default=0"
    tier="uint12, default=0xFFF">

  <!-- Splice commands, only one of them is allowed -->

  <splice_null/>

  <splice_schedule>
    <!-- One per splice event -->
    <splice_event
        splice_event_id="uint32, required"
        splice_event_cancel="bool, default=false"
        out_of_network="bool, required when splice_event_cancel is false"
        utc_splice_time="uint32, required when splice_event_cancel is false and
                         program_splice_flag is to be set"
        unique_program_id="uint16, required when splice_event_cancel is false"
        avail_num="uint8, default=0"
        avails_expected="uint8, default=0">
      <!-- Optional -->
      <break_duration
          auto_return="bool, required"
          duration="uint33, required"/>
      <!-- One per component when splice_event_cancel is false and
           utc_splice_time is not specified -->
      <component
          component_tag="uint8, required"
          utc_splice_time="uint32, required"/>
    </splice_event>
  </splice_schedule>

  <splice_insert
      splice_event_id="uint32, required"
      splice_event_cancel="bool, default=false"
      out_of_network="bool, required when splice_event_cancel is false"
      splice_immediate="bool, default=false"
      pts_time="uint33, required when splice_event_cancel is false and
                splice_immediate is false and program_splice_flag is to be set"
      unique_program_id="uint16, required when splice_event_cancel is false"
      avail_num="uint8, default=0"
      avails_expected="uint8, default=0">
    <!-- Optional -->
    <break_duration
        auto_return="bool, required"
        duration="uint33, required"/>
    <!-- One per component when splice_event_cancel is false and
         pts_time is not specified -->
    <component
         component_tag="uint8, required"
         pts_time="uint33, required when splice_immediate is false"/>
  </splice_insert>

  <time_signal pts_time="uint33, optional"/>

  <bandwidth_reservation/>

  <private_command identifier="uint32, required">
    Hexadecimal digits.
```

```
    </private_command>

    <DESCRIPTOR_LIST>

  </splice_information_table>
```

C.5 ATSC-defined tables

C.5.1 Cable Virtual Channel Table (CVCT)

Defined by ATSC in [18].

```
  <CVCT version="uint5, default=0"
        current="bool, default=true"
        protocol_version="uint8, default=0"
        transport_stream_id="uint16, required">

    <!-- Common descriptors loop -->
    <DESCRIPTOR_LIST>

    <!-- One per channel -->
    <channel short_name="string, required"
             major_channel_number="uint10, required"
             minor_channel_number="uint10, required"
             modulation_mode="analog|64-QAM|256-QAM|8-VSB|16-VSB|uint8, required"
             carrier_frequency="uint32, default=0"
             channel_TSID="uint16, required"
             program_number="uint16, required"
             ETM_location="uint2, default=0"
             access_controlled="bool, default=false"
             hidden="bool, default=false"
             path_select="uint1, default=0"
             out_of_band="bool, default=0"
             hide_guide="bool, default=false"
             service_type="analog|dtv|audio|data|software|uint6, default=dtv"
             source_id="uint16, required">
      <DESCRIPTOR_LIST>
    </channel>

  </CVCT>
```

C.5.2 Master Guide Table (MGT)

Defined by ATSC in [18].

```
  <MGT version="uint5, default=0" protocol_version="uint8, default=0">

    <!-- Common descriptors loop -->
    <DESCRIPTOR_LIST>

    <!-- One per table type -->
    <table type="TVCT-current|TVCT-next|CVCT-current|CVCT-next|ETT|DCCSCT|
                 EIT-0..EIT-127|ETT-0..ETT-127|RRT-1..RRT-255|DCCT-0DCCT-255|
                 uint16, required"
           PID="uint31, required"
           version_number="uint5, required"
           number_bytes="uint32, required">
      <DESCRIPTOR_LIST>
    </table>

  </MGT>
```

C.5.3 System Time Table (STT)

Defined by ATSC in [18].

```
<STT protocol_version="uint8, default=0"
     system_time="uint32, required"
     GPS_UTC_offset="uint8, required"
     DS_status="bool, required"
     DS_day_of_month="uint5, default=0"
     DS_hour="uint8, default=0">

  <DESCRIPTOR_LIST>

</STT>
```

C.5.4 Terrestrial Virtual Channel Table (TVCT)

Defined by ATSC in [18].

```
<TVCT version="uint5, default=0"
      current="bool, default=true"
      protocol_version="uint8, default=0"
      transport_stream_id="uint16, required">

  <!-- Common descriptors loop -->
  <DESCRIPTOR_LIST>

  <!-- One per channel -->
  <channel short_name="string, required"
           major_channel_number="uint10, required"
           minor_channel_number="uint10, required"
           modulation_mode="analog|64-QAM|256-QAM|8-VSB|16-VSB|uint8, required"
           carrier_frequency="uint32, default=0"
           channel_TSID="uint16, required"
           program_number="uint16, required"
           ETM_location="uint2, default=0"
           access_controlled="bool, default=false"
           hidden="bool, default=false"
           hide_guide="bool, default=false"
           service_type="analog|dtv|audio|data|software|uint6, default=dtv"
           source_id="uint16, required">
    <DESCRIPTOR_LIST>
  </channel>

</TVCT>
```

C.6 MPEG-defined descriptors

C.6.1 association_tag_descriptor

Defined by MPEG in [3].

```
<association_tag_descriptor
    association_tag="uint16, required"
    use="uint16, required">
  <selector_bytes>
    Hexadecimal content
  </selector_bytes>
  <private_data>
    Hexadecimal content
  </private_data>
</association_tag_descriptor>
```

C.6.2 audio_stream_descriptor

Defined by MPEG in [1].

```
<audio_stream_descriptor
    free_format="bool, required"
    ID="uint1, required"
    layer="uint2, required"
```

```
    variable_rate_audio="bool, required"/>
```

C.6.3 AVC_timing_and_HRD_descriptor

Defined by MPEG in [1].

```
<AVC_timing_and_HRD_descriptor
    hrd_management_valid="bool, required"
    N_90khz="uint32, optional"
    K_90khz="uint32, optional"
    num_units_in_tick="uint32, optional"
    fixed_frame_rate="bool, required"
    temporal_poc="bool, required"
    picture_to_display_conversion="bool, required"/>
```

C.6.4 AVC_video_descriptor

Defined by MPEG in [1].

```
<AVC_video_descriptor
    profile_idc="uint8, required"
    constraint_set0="bool, required"
    constraint_set1="bool, required"
    constraint_set2="bool, required"
    AVC_compatible_flags="uint5, required"
    level_idc="uint8, required"
    AVC_still_present="bool, required"
    AVC_24_hour_picture="bool, required"/>
```

C.6.5 CA_descriptor

Defined by MPEG in [1].

```
<CA_descriptor CA_system_id="uint16, required" CA_PID="uint13, required">
  <private_data>
    Hexadecimal content
  </private_data>
</CA_descriptor>
```

C.6.6 carousel_identifier_descriptor

Defined by MPEG in [3].

```
<carousel_identifier_descriptor carousel_id="uint32, required">
  <private_data>
    Hexadecimal content
  </private_data>
</carousel_identifier_descriptor>
```

C.6.7 copyright_descriptor

Defined by MPEG in [1].

```
<copyright_descriptor copyright_identifier="uint32, required">
  <additional_copyright_info>
    Hexadecimal content (optional element)
  </additional_copyright_info>
</copyright_descriptor>
```

C.6.8 data_stream_alignment_descriptor

Defined by MPEG in [1].

```
<data_stream_alignment_descriptor alignment_type="uint8, required"/>
```

C.6.9 deferred_association_tags_descriptor

Defined by MPEG in [3].

```
<deferred_association_tags_descriptor
    transport_stream_id="uint16, required"
    program_number="uint16, required">
  <!-- One per association tag -->
  <association tag="uint16, required"/>
  <private_data>
    Hexadecimal content
  </private_data>
</deferred_association_tags_descriptor>
```

C.6.10 external_ES_ID_descriptor

Defined by MPEG in [1].

```
<external_ES_ID_descriptor external_ES_ID="uint16, required"/>
```

C.6.11 HEVC_timing_and_HRD_descriptor

Defined by MPEG in [1].

```
<HEVC_timing_and_HRD_descriptor
    hrd_management_valid="bool, required"
    N_90khz="uint32, optional"
    K_90khz="uint32, optional"
    num_units_in_tick="uint32, optional"/>
```

C.6.12 HEVC_video_descriptor

Defined by MPEG in [1].

```
<HEVC_video_descriptor
    profile_space="uint2, required"
    tier_flag="bool, required"
    profile_idc="uint5, required"
    profile_compatibility_indication="uint32, required"
    progressive_source_flag="bool, required"
    interlaced_source_flag="bool, required"
    non_packed_constraint_flag="bool, required"
    frame_only_constraint_flag="bool, required"
    reserved_zero_44bits="uint44, default=0"
    level_idc="uint8, required"
    HEVC_still_present_flag="bool, required"
    HEVC_24hr_picture_present_flag="bool, required"
    temporal_id_min="uint3, optional, specify both min and max or none"
    temporal_id_max="uint3, optional, specify both min and max or none"/>
```

C.6.13 hierarchy_descriptor

Defined by MPEG in [1].

```
<hierarchy_descriptor
    temporal_scalability="bool, required"
    spatial_scalability="bool, required"
    quality_scalability="bool, required"
    hierarchy_type="uint4, required"
    hierarchy_layer_index="uint6, required"
    tref_present="bool, required"
    hierarchy_embedded_layer_index="uint6, required"
    hierarchy_channel="uint6, required"/>
```

C.6.14 IBP_descriptor

Defined by MPEG in [1].

```
<IBP_descriptor
    closed_gop="bool, required"
    identical_gop="bool, required"
    max_gop_length="uint14, required"/>
```

C.6.15 ISO_639_language_descriptor

Defined by MPEG in [1].

```
<ISO_639_language_descriptor>
  <!-- One per language -->
  <language code="char3, required" audio_type="uint8, required"/>
</ISO_639_language_descriptor>
```

C.6.16 maximum_bitrate_descriptor

Defined by MPEG in [1].

```
<maximum_bitrate_descriptor maximum_bitrate="uint32, in bits/second, required"/>
```

C.6.17 MPEG4_audio_descriptor

Defined by MPEG in [1].

```
<MPEG4_audio_descriptor MPEG4_audio_profile_and_level="uint8, required"/>
```

C.6.18 MPEG4_video_descriptor

Defined by MPEG in [1].

```
<MPEG4_video_descriptor MPEG4_visual_profile_and_level="uint8, required"/>
```

C.6.19 multiplex_buffer_utilization_descriptor

Defined by MPEG in [1]. The two attributes must be present both or absent both.

```
<multiplex_buffer_utilization_descriptor
    LTW_offset_lower_bound="uint15, optional"
    LTW_offset_upper_bound="uint15, optional"/>
```

C.6.20 NPT_endpoint_descriptor

Defined by MPEG in [3].

```
<NPT_endpoint_descriptor
    start_NPT="uint33, required"
    stop_NPT="uint33, required"/>
```

C.6.21 NPT_reference_descriptor

Defined by MPEG in [3].

```
<NPT_reference_descriptor
    post_discontinuity="bool, default=false"
    content_id="uint7, default=0x7F"
    STC_reference="uint33, required"
    NPT_reference="uint33, required"
    scale_numerator="uint16, required"
    scale_denominator="uint16, required"/>
```

C.6.22 private_data_indicator_descriptor

Defined by MPEG in [1].

```
<private_data_indicator_descriptor private_data_indicator="uint32, required"/>
```

C.6.23 registration_descriptor

Defined by MPEG in [1].

```
<registration_descriptor format_identifier="uint32, required">
  <additional_identification_info>
    Hexadecimal content (optional element)
  </additional_identification_info>
</registration_descriptor>
```

C.6.24 SL_descriptor

Defined by MPEG in [1].

```
<SL_descriptor ES_ID="uint16, required"/>
```

C.6.25 smoothing_buffer_descriptor

Defined by MPEG in [1].

```
<smoothing_buffer_descriptor
    sb_leak_rate="uint22, required"
    sb_size="uint22, required"/>
```

C.6.26 STD_descriptor

Defined by MPEG in [1].

```
<STD_descriptor leak_valid="bool, required"/>
```

C.6.27 stream_event_descriptor

Defined by MPEG in [3]. Note: <private_data> and <private_text> are mutually exclusive. They both define the same private data part, the former using hexadecimal format and the latter ASCII text.

```
<stream_event_descriptor
    event_id="uint16, required"
    event_NPT="uint33, required">
  <private_data>
    Hexadecimal content
  </private_data>
  <private_text>
    ASCII string to be used instead of private_data
  </private_text>
</stream_event_descriptor>
```

C.6.28 stream_mode_descriptor

Defined by MPEG in [3].

```
<stream_mode_descriptor stream_mode="uint8, required"/>
```

C.6.29 system_clock_descriptor

Defined by MPEG in [1].

```
<system_clock_descriptor
    external_clock_reference="bool required"
    clock_accuracy_integer="uint6, required"
    clock_accuracy_exponent="uint3, required"/>
```

C.6.30 target_background_grid_descriptor

Defined by MPEG in [1].

```
<target_background_grid_descriptor
    horizontal_size="uint14, required"
    vertical_size="uint14, required"
    aspect_ratio_information="uint4, required"/>
```

C.6.31 video_stream_descriptor

Defined by MPEG in [1].

```
<video_stream_descriptor
    multiple_frame_rate="bool, required"
    frame_rate_code="uint4, required"
    MPEG_1_only="bool, required"
    constrained_parameter="bool, required"
    still_picture="bool, required"
```

```
    profile_and_level_indication="uint8, required when MPEG_1_only='false'"
    chroma_format="uint2, required when MPEG_1_only='false'"
    frame_rate_extension="bool, required when MPEG_1_only='false'"/>
```

C.6.32 video_window_descriptor

Defined by MPEG in [1].

```
<video_window_descriptor
    horizontal_offset="uint14, required"
    vertical_offset="uint14, required"
    window_priority="uint4, required"/>
```

C.7 DVB-defined descriptors

Note that a few descriptors are allowed in specific tables only since they reuse tag values which are otherwise MPEG-reserved [1]. They cannot be used elsewhere. These restrictions, when applicable, are documented in XML comments for the table-specific descriptor. Such descriptors exist for the AIT [12], the UNT [13] and the INT [14].

C.7.1 AAC_descriptor

Defined by DVB in [6].

```
<AAC_descriptor
    profile_and_level="uint8, required"
    SAOC_DE="bool, default=false"
    AAC_type="uint8, optional">
  <additional_info>
    Hexadecimal content, optional
  </additional_info>
</AAC_descriptor>
```

C.7.2 AC3_descriptor

Defined by DVB in [6].

```
<AC3_descriptor
    component_type="uint8, optional"
    bsid="uint8, optional"
    mainid="uint8, optional"
    asvc="uint8, optional">
  <additional_info>
    Hexadecimal content, optional
  </additional_info>
</AC3_descriptor>
```

C.7.3 AC4_descriptor

Defined by DVB in [6].

```
<AC4_descriptor
    ac4_dialog_enhancement_enabled="bool, optional"
    ac4_channel_mode="uint2, optional">
  <ac4_dsi_toc>
    Hexadecimal content, optional
  </ac4_dsi_toc>
  <additional_info>
    Hexadecimal content, optional
  </additional_info>
</AC4_descriptor>
```

C.7.4 adaptation_field_data_descriptor

Defined by DVB in [6].

```
<adaptation_field_data_descriptor adaptation_field_data_identifier="uint8, required"/>
```

C.7.5 ancillary_data_descriptor

Defined by DVB in [6].

```
<ancillary_data_descriptor ancillary_data_identifier="uint8, required"/>
```

C.7.6 application_descriptor

Defined by DVB in [12]. Must be in an AIT (table id 0x74).

```
<application_descriptor
    service_bound="bool, required"
    visibility="uint2, required"
    application_priority="uint8, required">
  <!-- One per profile -->
  <profile
      application_profile="uint16, required"
      version="string 'major.minor.micro', required"/>
  <!-- One per transport_protocol_label -->
  <transport_protocol label="uint8, required"/>
</application_descriptor>
```

C.7.7 application_icons_descriptor

Defined by DVB in [12]. Must be in an AIT (table id 0x74).

```
<application_icons_descriptor
    icon_locator="string, required"
    icon_flags="uint16, required">
  <reserved_future_use>
    Hexadecimal content
  </reserved_future_use>
</application_icons_descriptor>
```

C.7.8 application_name_descriptor

Defined by DVB in [12] and [15]. Must be in an AIT (table id 0x74).

```
<application_name_descriptor>
  <!-- One per language -->
  <language code="char3, required" application_name="string, required"/>
</application_name_descriptor>
```

C.7.9 application_recording_descriptor

Defined by DVB in [12]. Must be in an AIT (table id 0x74).

```
<application_recording_descriptor
    scheduled_recording="bool, required"
    trick_mode_aware="bool, required"
    time_shift="bool, required"
    dynamic="bool, required"
    av_synced="bool, required"
    initiating_replay="bool, required">
  <!-- One per label -->
  <label label="string, required" storage_properties="uint2, required"/>
  <!-- One per component tag -->
  <component tag="uint8, required"/>
  <private>
    Hexadecimal content
  </private>
  <reserved_future_use>
    Hexadecimal content
  </reserved_future_use>
</application_recording_descriptor>
```

C.7.10 application_signalling_descriptor

Defined by DVB in [12].

```
<application_signalling_descriptor>
  <!-- One per application -->
  <application application_type="uint15, required"
               AIT_version_number="uint5, required"/>
</application_signalling_descriptor>
```

C.7.11 application_storage_descriptor

Defined by DVB in [12]. Must be in an AIT (table id `0x74`).

```
<application_storage_descriptor
    storage_property="uint8, required"
    not_launchable_from_broadcast="bool, required"
    launchable_completely_from_cache="bool, required"
    is_launchable_with_older_version="bool, required"
    version="uint31, required"
    priority="uint8, required"/>
```

C.7.12 application_usage_descriptor

Defined by DVB in [12]. Must be in an AIT (table id `0x74`).

```
<application_usage_descriptor usage_type="uint8, required"/>
```

C.7.13 audio_preselection_descriptor

Defined by DVB in [6].

```
<audio_preselection_descriptor>
  <!-- One entry per preselection, up to 31 preselections -->
  <preselection
      preselection_id="uint5, required"
      audio_rendering_indication="uint3, required"
      audio_description="bool, default=false"
      spoken_subtitles="bool, default=false"
      dialogue_enhancement="bool, default=false"
      interactivity_enabled="bool, default=false"
      ISO_639_language_code="char3, optional"
      message_id="uint8, optional">
    <multi_stream_info>
      <!-- One per auxiliary component, up to 7 components -->
      <component tag="uint8, required"/>
    </multi_stream_info>
    <future_extension>
      Hexadecimal content
    </future_extension>
  </preselection>
</audio_preselection_descriptor>
```

C.7.14 bouquet_name_descriptor

Defined by DVB in [6].

```
<bouquet_name_descriptor bouquet_name="string, required"/>
```

C.7.15 CA_identifier_descriptor

Defined by DVB in [6].

```
<CA_identifier_descriptor>
  <!-- One per CAS -->
  <CA_system_id value="uint16, required"/>
</CA_identifier_descriptor>
```

C.7.16 cable_delivery_system_descriptor

Defined by DVB in [6].

```
<cable_delivery_system_descriptor
    frequency="FrequencyHz, required"
    FEC_outer="undefined|none|RS, default=RS"
    modulation="auto|16-QAM|32-QAM|64-QAM|128-QAM|256-QAM, default=16-QAM"
    symbol_rate="SymbolsPerSecond, required"
    FEC_inner="undefined|1/2|2/3|3/4|5/6|7/8|8/9|3/5|4/5|9/10|none, required"/>
```

C.7.17 CI_ancillary_data_descriptor

Defined by DVB in [6].

```
<CI_ancillary_data_descriptor>
  <ancillary_data>
    Hexadecimal content
  </ancillary_data>
</CI_ancillary_data_descriptor>
```

C.7.18 component_descriptor

Defined by DVB in [6].

```
<component_descriptor
    stream_content="uint4, required"
    stream_content_ext="uint4, default=0xF"
    component_type="uint8, required"
    component_tag="uint8, default=0"
    language_code="char3, required"
    text="string, optional"/>
```

C.7.19 content_descriptor

Defined by DVB in [6].

```
<content_descriptor>
  <!-- One per classification -->
  <content content_nibble_level_1="uint4, required"
           content_nibble_level_2="uint4, required"
           user_byte="uint8, required"/>
</content_descriptor>
```

C.7.20 country_availability_descriptor

Defined by DVB in [6].

```
<country_availability_descriptor country_availability="bool, required">
  <!-- One per country -->
  <country country_code="char3, required"/>
</country_availability_descriptor>
```

C.7.21 CP_descriptor

Defined by DVB in [6].

```
<CP_descriptor CP_system_id="uint16, required" CP_PID="uint13, required">
  <private_data>
    Hexadecimal content
  </private_data>
</CP_descriptor>
```

C.7.22 CP_identifier_descriptor

Defined by DVB in [6].

```
<CP_identifier_descriptor>
  <!-- One per CP system -->
```

```
  <CP_system_id value="uint16, required"/>
</CP_identifier_descriptor>
```

C.7.23 data_broadcast_descriptor

Defined by DVB in [6].

```
<data_broadcast_descriptor
    data_broadcast_id="uint16, required"
    component_tag="uint8, required"
    language_code="char3, required">
  <selector_bytes>Hexadecimal content</selector_bytes>
  <text>String</text>
</data_broadcast_descriptor>
```

C.7.24 data_broadcast_id_descriptor

Defined by DVB in [6].

```
<data_broadcast_id_descriptor data_broadcast_id="uint16, required">
  <selector_bytes>Hexadecimal content</selector_bytes>
</data_broadcast_id_descriptor>
```

C.7.25 DII_location_descriptor

Defined by DVB in [15]. Must be in an AIT (table id `0x74`).

```
<DII_location_descriptor transport_protocol_label="uint8, required">
  <!-- One per module -->
  <module DII_identification="uint15, required" association_tag="uint16, required"/>
</DII_location_descriptor>
```

C.7.26 DTS_descriptor

Defined by DVB in [6].

```
<DTS_descriptor
    sample_rate_code="uint4, required"
    bit_rate_code="uint6, required"
    nblks="uint7, 0x05 to 0x1F, required"
    fsize="uint14, 0x005F to 0x2000, required"
    surround_mode="uint6, required"
    lfe="bool, default=false"
    extended_surround="uint2, default=0">
  <additional_info>
    Hexadecimal content
  </additional_info>
</DTS_descriptor>
```

C.7.27 DTS_neural_descriptor

Defined by DVB in [6].

```
<DTS_neural_descriptor config_id="uint8, required">
  <additional_info>
    Hexadecimal content
  </additional_info>
</DTS_neural_descriptor>
```

C.7.28 dvb_html_application_boundary_descriptor

Defined by DVB in [15]. Must be in an AIT (table id `0x74`).

```
<dvb_html_application_boundary_descriptor
    label="string, required"
    regular_expression="string, required"/>
```

C.7.29 dvb_html_application_descriptor

Defined by DVB in [15]. Must be in an AIT (table id `0x74`).

```
<dvb_html_application_descriptor parameter="string, optional">
  <!-- One per application id: -->
  <application id="uint16, required"/>
</dvb_html_application_descriptor>
```

C.7.30 dvb_html_application_location_descriptor

Defined by DVB in [15]. Must be in an AIT (table id `0x74`).

```
<dvb_html_application_location_descriptor
    physical_root="string, required"
    initial_path="string, required"/>
```

C.7.31 dvb_j_application_descriptor

Defined by DVB in [15]. Must be in an AIT (table id `0x74`).

```
<dvb_j_application_descriptor>
  <!-- One per parameter: -->
  <parameter value="string, required"/>
</dvb_j_application_descriptor>
```

C.7.32 dvb_j_application_location_descriptor

Defined by DVB in [15]. Must be in an AIT (table id `0x74`).

```
<dvb_j_application_location_descriptor
    base_directory="string, required"
    classpath_extension="string, required"
    initial_class="string, required"/>
```

C.7.33 ECM_repetition_rate_descriptor

Defined by DVB in [14].

```
<ECM_repetition_rate_descriptor
    CA_system_id="uint16, required"
    ECM_repetition_rate="uint16, required">
  <private_data>
    Hexadecimal content
  </private_data>
</ECM_repetition_rate_descriptor>
```

C.7.34 enhanced_AC3_descriptor

Defined by DVB in [6].

```
<enhanced_AC3_descriptor
    mixinfoexists="bool, required"
    component_type="uint8, optional"
    bsid="uint8, optional"
    mainid="uint8, optional"
    asvc="uint8, optional"
    substream1="uint8, optional"
    substream2="uint8, optional"
    substream3="uint8, optional">
  <additional_info>
    Hexadecimal content
  </additional_info>
</enhanced_AC3_descriptor>
```

C.7.35 extended_event_descriptor

Defined by DVB in [6].

```
<extended_event_descriptor
    descriptor_number="uint8, required"
    last_descriptor_number="uint8, required"
    language_code="char3, required">
  <text>String</text>
  <!-- One per item -->
  <item>
    <description>String</description>
    <name>String</name>
  </item>
</extended_event_descriptor>
```

C.7.36 external_application_authorization_descriptor

Defined by DVB in [12]. Must be in an AIT (table id 0x74).

```
<external_application_authorization_descriptor>
  <!-- One per application -->
  <application
      organization_id="uint32, required"
      application_id="uint16, required"
      application_priority="uint8, required"/>
</external_application_authorization_descriptor>
```

C.7.37 graphics_constraints_descriptor

Defined by DVB in [12]. Must be in an AIT (table id 0x74).

```
<graphics_constraints_descriptor
    can_run_without_visible_ui="bool, required"
    handles_configuration_changed="bool, required"
    handles_externally_controlled_video="bool, required">
  <graphics_configuration>
    Hexadecimal content
  </graphics_configuration>
</graphics_constraints_descriptor>
```

C.7.38 IPMAC_generic_stream_location_descriptor

Defined by DVB in [14]. Must be in an INT (table id 0x4C).

```
<IPMAC_generic_stream_location_descriptor
    interactive_network_id="uint16, required"
    modulation_system_type="DVB-S2|DVB-T2|DVB-C2|DVB-NGH|uint8, required"
    modulation_system_id="uint16, default=0"
    PHY_stream_id="uint16, default=0">
  <selector_bytes>Hexadecimal content</selector_bytes>
</IPMAC_generic_stream_location_descriptor>
```

C.7.39 IPMAC_platform_name_descriptor

Defined by DVB in [14]. Must be in an INT (table id 0x4C).

```
<IPMAC_platform_name_descriptor
    language_code="char3, required"
    text="string, required"/>
```

C.7.40 IPMAC_platform_provider_name_descriptor

Defined by DVB in [14]. Must be in an INT (table id 0x4C).

```
<IPMAC_platform_provider_name_descriptor
    language_code="char3, required"
    text="string, required"/>
```

C.7.41 IPMAC_stream_location_descriptor

Defined by DVB in [14]. Must be in an INT (table id 0x4C).

```
<IPMAC_stream_location_descriptor
    network_id="uint16, required"
    original_network_id="uint16, required"
    transport_stream_id="uint16, required"
    service_id="uint16, required"
    component_tag="uint8, required"/>
```

C.7.42 ip_signalling_descriptor

Defined by DVB in [15]. Must be in an INT (table id 0x4C).

```
<ip_signalling_descriptor platform_id="uint24, required"/>
```

C.7.43 ISP_access_mode_descriptor

Defined by DVB in [14]. Must be in an INT (table id 0x4C).

```
<ISP_access_mode_descriptor access_mode="unused|dialup|uint8, required"/>
```

C.7.44 linkage_descriptor

Defined by DVB in [6].

```
<linkage_descriptor
    transport_stream_id="uint16, required"
    original_network_id="uint16, required"
    service_id="uint16, required"
    linkage_type="uint8, required">
  <!-- if linkage_type == 0x08 -->
  <mobile_handover_info
      handover_type="uint4, required"
      origin_type="NIT|SDT, required"
      network_id="uint16, required if hand-over_type is 0x01, 0x02, 0x03"
      initial_service_id="uint16, required if origin_type is NIT"/>
  <!-- else if linkage_type == 0x0D -->
  <event_linkage_info
      target_event_id="uint16, required"
      target_listed="bool, required"
      event_simulcast="bool, required"/>
  <!-- else if linkage_type >= 0x0E && linkage_type <= 0x1F -->
  <extended_event_linkage_info>
    <!-- For each event -->
    <event
        target_event_id="uint16, required"
        target_listed="bool, required"
        event_simulcast="bool, required"
        link_type="uint2, required"
        target_id_type="uint2, required"
        user_defined_id="uint16, required if target_id_type == 3"
        target_transport_stream_id="uint16, required if target_id_type == 1"
        target_original_network_id="uint16, optional"
        target_service_id="uint16, optional"/>
  </extended_event_linkage_info>
  <private_data>
    Hexadecimal content
  </private_data>
</linkage_descriptor>
```

C.7.45 local_time_offset_descriptor

Defined by DVB in [6].

```
<local_time_offset_descriptor>
  <!-- One per region -->
  <region country_code="char3, required"
          country_region_id="uint6, required"
          local_time_offset="int, required"
          time_of_change="YYYY-MM-DD hh:mm:ss, required"
```

```
          next_time_offset="int, required"/>
  <!-- local_time_offset and next_time_offset: -->
  <!-- -780 to +780 minutes (-13 to +13 hours) -->
</local_time_offset_descriptor>
```

C.7.46 message_descriptor

Defined by DVB in [6].

```
<message_descriptor message_id="uint8, required" language_code="char3, required">
  <text>String</text>
</message_descriptor>
```

C.7.47 multilingual_bouquet_name_descriptor

Defined by DVB in [6].

```
<multilingual_bouquet_name_descriptor>
  <!-- One per language -->
  <language code="char3, required" bouquet_name="string, required"/>
</multilingual_bouquet_name_descriptor>
```

C.7.48 multilingual_component_descriptor

Defined by DVB in [6].

```
<multilingual_component_descriptor component_tag="uint8, required">
  <!-- One per language -->
  <language code="char3, required" description="string, required"/>
</multilingual_component_descriptor>
```

C.7.49 multilingual_network_name_descriptor

Defined by DVB in [6].

```
<multilingual_network_name_descriptor>
  <!-- One per language -->
  <language code="char3, required" network_name="string, required"/>
</multilingual_network_name_descriptor>
```

C.7.50 multilingual_service_name_descriptor

Defined by DVB in [6].

```
<multilingual_service_name_descriptor>
  <!-- One per language -->
  <language code="char3, required"
            service_provider_name="string, required"
            service_name="string, required"/>
</multilingual_service_name_descriptor>
```

C.7.51 network_name_descriptor

Defined by DVB in [6].

```
<network_name_descriptor network_name="string, required"/>
```

C.7.52 NVOD_reference_descriptor

Defined by DVB in [6].

```
<NVOD_reference_descriptor>
  <!-- One per service -->
  <service transport_stream_id="uint16, required"
           original_network_id="uint16, required"
           service_id="uint16, required"/>
</NVOD_reference_descriptor>
```

C.7.53 parental_rating_descriptor

Defined by DVB in [6].

```
<parental_rating_descriptor>
  <!-- One per country -->
  <country country_code="char3, required" rating="uint8, required"/>
</parental_rating_descriptor>
```

C.7.54 partial_transport_stream_descriptor

Defined by DVB in [6].

```
<partial_transport_stream_descriptor
    peak_rate="uint22, required"
    minimum_overall_smoothing_rate="uint22, default=0x3FFFFF"
    maximum_overall_smoothing_buffer="uint14, default=0x3FFF"/>
```

C.7.55 prefetch_descriptor

Defined by DVB in [15]. Must be in an AIT (table id `0x74`).

```
<prefetch_descriptor transport_protocol_label="uint8, required">
  <!-- One per module -->
  <module label="string, required" prefetch_priority="int, 1 to 100, required"/>
</prefetch_descriptor>
```

C.7.56 private_data_specifier_descriptor

Defined by DVB in [6].

```
<private_data_specifier_descriptor
    private_data_specifier="uint32|eacem|eutelsat, required"/>
```

C.7.57 protection_message_descriptor

Defined by DVB in [12].

```
<protection_message_descriptor>
  <!-- One per component, up to 15 components -->
  <component tag="uint8, required"/>
</protection_message_descriptor>
```

C.7.58 S2_satellite_delivery_system_descriptor

Defined by DVB in [6].

```
<S2_satellite_delivery_system_descriptor
    backwards_compatibility="bool, required"
    scrambling_sequence_index="uint18, optional"
    input_stream_identifier="uint8, optional"/>
```

C.7.59 satellite_delivery_system_descriptor

Defined by DVB in [6].

```
<satellite_delivery_system_descriptor
    frequency="SatelliteFrequencyHz, required"
    orbital_position="SatelliteOrbitalPosition, eg. 19.2, required"
    west_east_flag="east|west, required"
    polarization="horizontal|vertical|left|right, required"
    roll_off="0.35|0.25|0.20|reserved, default=0.35"
    modulation_system="DVB-S|DVB-S2, default=DVB-S"
    modulation_type="auto|QPSK|8PSK|16-QAM, default=QPSK"
    symbol_rate="SymbolsPerSecond, required"
    FEC_inner="undefined|1/2|2/3|3/4|5/6|7/8|8/9|3/5|4/5|9/10|none|, required"/>
```

C.7.60 scrambling_descriptor

Defined by DVB in [6].

```
<scrambling_descriptor scrambling_mode="uint8, required"/>
```

C.7.61 service_descriptor

Defined by DVB in [6].

```
<service_descriptor
    service_type="uint8, required"
    service_provider_name="string, required"
    service_name="string, required"/>
```

C.7.62 service_availability_descriptor

Defined by DVB in [6].

```
<service_availability_descriptor availability="bool, required">
  <!-- One per cell -->
  <cell id="uint16, required"/>
</service_availability_descriptor>
```

C.7.63 service_identifier_descriptor

Defined by DVB in [6].

```
<service_identifier_descriptor service_identifier="string, required"/>
```

C.7.64 service_list_descriptor

Defined by DVB in [6].

```
<service_list_descriptor>
  <!-- One per service -->
  <service service_id="uint16, required" service_type="uint8, required"/>
</service_list_descriptor>
```

C.7.65 service_move_descriptor

Defined by DVB in [6].

```
<service_move_descriptor
    new_original_network_id="uint16, required"
    new_transport_stream_id="uint16, required"
    new_service_id="uint16, required"/>
```

C.7.66 service_relocated_descriptor

Defined by DVB in [6].

```
<service_relocated_descriptor
    old_original_network_id="uint16, required"
    old_transport_stream_id="uint16, required"
    old_service_id="uint16, required"/>
```

C.7.67 short_event_descriptor

Defined by DVB in [6].

```
<short_event_descriptor language_code="char3, required">
  <event_name>String</event_name>
  <text>String</text>
</short_event_descriptor>
```

C.7.68 simple_application_boundary_descriptor

Defined by DVB in [12]. Must be in an AIT (table id `0x74`).

```
<simple_application_boundary_descriptor>
  <!-- One per prefix: -->
  <prefix boundary_extension="string, required"/>
</simple_application_boundary_descriptor>
```

C.7.69 simple_application_location_descriptor

Defined by DVB in [12]. Must be in an AIT (table id `0x74`).

```
<simple_application_location_descriptor initial_path="string, required"/>
```

C.7.70 stream_identifier_descriptor

Defined by DVB in [6].

```
<stream_identifier_descriptor component_tag="uint8, required"/>
```

C.7.71 stuffing_descriptor

Defined by DVB in [6].

```
<stuffing_descriptor>
  Hexadecimal content
</stuffing_descriptor>
```

C.7.72 subtitling_descriptor

Defined by DVB in [6].

```
<subtitling_descriptor>
  <!-- One per subtitle -->
  <subtitling language_code="char3, required"
              subtitling_type="uint8, required"
              composition_page_id="uint16, required"
              ancillary_page_id="uint16, required"/>
</subtitling_descriptor>
```

C.7.73 supplementary_audio_descriptor

Defined by DVB in [6].

```
<supplementary_audio_descriptor
    mix_type="uint1, required"
    editorial_classification="uint5, required"
    language_code="char3, optional">
  <private_data>
    Hexadecimal content
  </private_data>
</supplementary_audio_descriptor>
```

C.7.74 T2MI_descriptor

Defined by DVB in [6].

```
<T2MI_descriptor
    t2mi_stream_id="uint3, required"
    num_t2mi_streams_minus_one="uint3, default=0"
    pcr_iscr_common_clock_flag="bool, default=false">
  <reserved>
    Hexadecimal content
  </reserved>
</T2MI_descriptor>
```

C.7.75 target_IP_address_descriptor

Defined by DVB in [14] and [9]. Must be in a UNT (table id `0x4B`) or INT (table id `0x4C`).

```
<target_IP_address_descriptor IPv4_addr_mask="IPv4 address, required">
  <!-- One per IPv4 address: -->
```

```
  <address IPv4_addr="IPv4 address, required"/>
</target_IP_address_descriptor>
```

C.7.76 target_IP_slash_descriptor

Defined by DVB in [14]. Must be in an INT (table id 0x4C).

```
<target_IP_slash_descriptor>
  <!-- One per IPv4 address: -->
  <address
      IPv4_addr="IPv4 address, required"
      IPv4_slash_mask="uint8, required"/>
</target_IP_slash_descriptor>
```

C.7.77 target_IP_source_slash_descriptor

Defined by DVB in [14]. Must be in an INT (table id 0x4C).

```
<target_IP_source_slash_descriptor>
  <!-- One per pair of IPv4 address: -->
  <address
      IPv4_source_addr="IPv4 address, required"
      IPv4_source_slash_mask="uint8, required"
      IPv4_dest_addr="IPv4 address, required"
      IPv4_dest_slash_mask="uint8, required"/>
</target_IP_source_slash_descriptor>
```

C.7.78 target_IPv6_address_descriptor

Defined by DVB in [14] and [9]. Must be in a UNT (table id 0x4B) or INT (table id 0x4C).

```
<target_IPv6_address_descriptor IPv6_addr_mask="IPv6 address, required">
  <!-- One per IPv6 address: -->
  <address IPv6_addr="IPv6 address, required"/>
</target_IPv6_address_descriptor>
```

C.7.79 target_IPv6_slash_descriptor

Defined by DVB in [14]. Must be in an INT (table id 0x4C).

```
<target_IPv6_slash_descriptor>
  <!-- One per IPv6 address: -->
  <address
      IPv6_addr="IPv6 address, required"
      IPv6_slash_mask="uint8, required"/>
</target_IPv6_slash_descriptor>
```

C.7.80 target_IPv6_source_slash_descriptor

Defined by DVB in [14]. Must be in an INT (table id 0x4C).

```
<target_IPv6_source_slash_descriptor>
  <!-- One per pair of IPv6 address: -->
  <address
      IPv6_source_addr="IPv6 address, required"
      IPv6_source_slash_mask="uint8, required"
      IPv6_dest_addr="IPv6 address, required"
      IPv6_dest_slash_mask="uint8, required"/>
</target_IPv6_source_slash_descriptor>
```

C.7.81 target_MAC_address_descriptor

Defined by DVB in [14] and [9]. Must be in a UNT (table id 0x4B) or INT (table id 0x4C).

```
<target_MAC_address_descriptor MAC_addr_mask="MAC address, required">
  <!-- One per MAC address: -->
  <address MAC_addr="MAC address, required"/>
</target_MAC_address_descriptor>
```

C.7.82 target_MAC_address_range_descriptor

Defined by DVB in [14]. Must be in an INT (table id 0x4C).

```
<target_MAC_address_range_descriptor>
  <!-- One per MAC address range: -->
  <range
      MAC_addr_low="MAC address, required"
      MAC_addr_high="MAC address, required"/>
</target_MAC_address_range_descriptor>
```

C.7.83 target_serial_number_descriptor

Defined by DVB in [14] and [9]. Must be in a UNT (table id 0x4B) or INT (table id 0x4C).

```
<target_serial_number_descriptor>
  <!-- Serial data bytes -->
  Hexadecimal content
</target_serial_number_descriptor>
```

C.7.84 target_smartcard_descriptor

Defined by DVB in [14] and [9]. Must be in a UNT (table id 0x4B) or INT (table id 0x4C).

```
<target_smartcard_descriptor super_CA_system_id="uint32, required">
  <!-- Private data bytes -->
  Hexadecimal content
</target_smartcard_descriptor>
```

C.7.85 teletext_descriptor

Defined by DVB in [6].

```
<teletext_descriptor>
  <!-- One per page -->
  <teletext language_code="char3, required"
            teletext_type="uint5, required"
            page_number="uint16, required"/>
</teletext_descriptor>
```

C.7.86 terrestrial_delivery_system_descriptor

Defined by DVB in [6].

```
<terrestrial_delivery_system_descriptor
    centre_frequency="FrequencyHz, required"
    bandwidth="8MHz|7MHz|6MHz|5MHz, required"
    priority="HP|LP, required"
    no_time_slicing="bool, required"
    no_MPE_FEC="bool, required"
    constellation="QPSK|16-QAM|64-QAM, required"
    hierarchy_information="uint3, required"
    code_rate_HP_stream="1/2|2/3|3/4|5/6|7/8, required"
    code_rate_LP_stream="1/2|2/3|3/4|5/6|7/8, required"
    guard_interval="1/32|1/16|1/8|1/4, required"
    transmission_mode="2k|8k|4k, required"
    other_frequency="bool, required"/>
```

C.7.87 time_shifted_event_descriptor

Defined by DVB in [6].

```
<time_shifted_event_descriptor
    reference_service_id="uint16, required"
    reference_event_id="uint16, required"/>
```

C.7.88 time_shifted_service_descriptor

Defined by DVB in [6].

```
<time_shifted_service_descriptor reference_service_id="uint16, required"/>
```

C.7.89 time_slice_fec_identifier_descriptor

Defined by DVB in [14].

```
<time_slice_fec_identifier_descriptor
    time_slicing="bool, required"
    mpe_fec="uint2, required"
    frame_size="uint3, required"
    max_burst_duration="uint8, required"
    max_average_rate="uint4, required"
    time_slice_fec_id="uint4, default=0">
  <id_selector_bytes>Hexadecimal content</id_selector_bytes>
</time_slice_fec_identifier_descriptor>
```

C.7.90 transport_protocol_descriptor

Defined by DVB in [15]. Must be in an AIT (table id `0x74`).

```
<transport_protocol_descriptor transport_protocol_label="uint8, required">
  <!-- Only one of the following shall be present -->
  <!-- For protocol id 1: -->
  <object_carousel
      original_network_id="uint16, optional"
      transport_stream_id="uint16, optional"
      service_id="uint16, optional"
      component_tag="uint8, required"/>
  <!-- For protocol id 2: -->
  <ip_mpe
      original_network_id="uint16, optional"
      transport_stream_id="uint16, optional"
      service_id="uint16, optional"
      alignment_indicator="bool, required">
    <!-- One per URL -->
    <url value="string, required"/>
  </ip_mpe>
  <!-- For protocol id 3: -->
  <http>
    <!-- One per URL -->
    <url base="string, required">
      <!-- One per URL extension -->
      <extension value="string, required"/>
    </url>
  </http>
  <!-- For other (unknown) protocol ids: -->
  <protocol id="uint16, required">
    Hexadecimal content.
  </protocol>
</transport_protocol_descriptor>
```

C.7.91 transport_stream_descriptor

Defined by DVB in [6].

```
<transport_stream_descriptor compliance="string, required"/>
```

C.7.92 VBI_data_descriptor

Defined by DVB in [6].

```
<VBI_data_descriptor>
  <!-- One per VBI data service -->
  <service data_service_id="uint8, required">
    <!-- One per field in the service -->
    <field field_parity="bool, default=false" line_offset="uint5, default=0"/>
    <!-- Valid only when data_service_id is not any of 1, 2, 4, 5, 6, 7 -->
    <reserved>
```

```
      Hexadecimal content
    </reserved>
  </service>
</VBI_data_descriptor>
```

C.7.93 VBI_teletext_descriptor

Defined by DVB in [6].

```
<VBI_teletext_descriptor>
  <!-- One per page -->
  <teletext language_code="char3, required"
            teletext_type="uint5, required"
            page_number="uint16, required"/>
</VBI_teletext_descriptor>
```

C.8 EACEM-defined descriptors (DVB private descriptors)

C.8.1 eacem_preferred_name_identifier_descriptor

Defined by EACEM in [20].

```
<eacem_preferred_name_identifier_descriptor name_id="uint8, required"/>
```

C.8.2 eacem_preferred_name_list_descriptor

Defined by EACEM in [20].

```
<eacem_preferred_name_list_descriptor>
  <!-- One per language -->
  <language code="char3, required">
    <!-- One per name -->
    <name name_id="uint8, required" name="string, required"/>
  </language>
</eacem_preferred_name_list_descriptor>
```

C.8.3 eacem_stream_identifier_descriptor

Defined by EACEM in [20].

```
<eacem_stream_identifier_descriptor version_byte="uint8, required"/>
```

C.8.4 HD_simulcast_logical_channel_descriptor

Defined by EACEM in [20].

```
<HD_simulcast_logical_channel_descriptor>
  <!-- One per service -->
  <service service_id="uint16, required"
           logical_channel_number="uint10, required"
           visible_service="bool, default=true"/>
</HD_simulcast_logical_channel_descriptor>
```

C.8.5 logical_channel_number_descriptor

Defined by EACEM in [20].

```
<logical_channel_number_descriptor>
  <!-- One per service -->
  <service service_id="uint16, required"
           logical_channel_number="uint10, required"
           visible_service="bool, default=true"/>
</logical_channel_number_descriptor>
```

C.9 Eutelsat-defined descriptors (DVB private descriptors)

C.9.1 eutelsat_channel_number_descriptor

Defined by Eutelsat in [21].

```
<eutelsat_channel_number_descriptor>
  <!-- One per service -->
  <service original_network_id="uint16, required"
           transport_stream_id="uint16, required"
           service_id="uint16, required"
           eutelsat_channel_number="uint10, required"/>
</eutelsat_channel_number_descriptor>
```

C.10 SCTE-defined descriptors

C.10.1 cue_identifier_descriptor

Defined by ANSI/SCTE in [17].

```
<cue_identifier_descriptor
  cue_stream_type="insert_null_schedule|all|segmentation|tiered_splicing|
                   tiered_segmentation|uint8, required">
  <!-- Defined by SCTE 35 for use in PMT -->
</cue_identifier_descriptor>
```

C.10.2 splice_avail_descriptor

Defined by ANSI/SCTE in [17]. Must be in a Splice Information Table (table id `0xFC`).

```
<splice_avail_descriptor
    identifier="uint32, default=0x43554549"
    provider_avail_id="uint32, required"/>
```

C.10.3 splice_DTMF_descriptor

Defined by ANSI/SCTE in [17]. Must be in a Splice Information Table (table id `0xFC`).

```
<splice_DTMF_descriptor
    identifier="uint32, default=0x43554549"
    preroll="uint8, required"
    DTMF="string, required"/>
```

C.10.4 splice_segmentation_descriptor

Defined by ANSI/SCTE in [17]. Must be in a Splice Information Table (table id `0xFC`).

```
<splice_segmentation_descriptor
    identifier="uint32, default=0x43554549"
    segmentation_event_id="uint32, required"
    segmentation_event_cancel="bool, default=false"
    web_delivery_allowed="bool, default=true"
    no_regional_blackout="bool, default=true"
    archive_allowed="bool, default=true"
    device_restrictions="uint2, default=3"
    segmentation_duration="uint40, optional"
    segmentation_type_id="uint8, required"
    segment_num="uint8, required"
    segments_expected="uint8, required"
    sub_segment_num="uint8, required when segmentation_type_id == 0x34 or 0x36"
    sub_segments_expected="uint8, required when segmentation_type_id == 0x34 or 0x36">
  <segmentation_upid type="uint8, required">
    Hexadecimal content
  </segmentation_upid>
  <!-- One per component when program_segmentation_flag is to be set to 0 -->
  <component component_tag="uint8, required" pts_offset="uint33, required"/>
</splice_segmentation_descriptor>
```

C.10.5 splice_time_descriptor

Defined by ANSI/SCTE in [17]. Must be in a Splice Information Table (table id `0xFC`).

```
<splice_time_descriptor
    identifier="uint32, default=0x43554549"
    TAI_seconds="uint48, required"
    TAI_ns="uint32, required"
    UTC_offset="uint16, required"/>
```

C.11 Generic format for unsupported tables and descriptors

Unsupported tables and descriptors can be represented using generic XML tags.

C.11.1 Generic short table

```
<generic_short_table table_id="uint8, required" private="bool, default=true">

  Generic table with binary payload of one short section, to be used when a
  specific table is not yet implemented. The body of this element shall contain
  an even number of hexadecimal digits, the payload of the short section.

  The private indicator shall be false on MPEG-defined sections and preferably
  true on DVB-defined and user-defined sections.

</generic_short_table>
```

C.11.2 Generic long table

```
<generic_long_table
    table_id="uint8, required"
    table_id_ext="uint16, default=0xFFFF"
    version="uint5, default=0"
    current="bool, default=true"
    private="bool, default=true">

  Generic table with binary payload of long sections, to be used when a specific
  table is not yet implemented.

  The private indicator shall be false on MPEG-defined sections and preferably
  true on DVB-defined and user-defined sections.

  <!-- One per section -->
  <section>
    The body of the section elements shall contain an even number of hexadecimal
    digits, the payload of the long section. The CRC32 field is not part of this
    payload, it will be recomputed.
  </section>

</generic_long_table>
```

C.11.3 Generic descriptor

```
<generic_descriptor tag="uint8, required">

  Generic descriptor with binary payload, to be used when a specific descriptor
  is not yet implemented. The body of this element shall contain an even number
  of hexadecimal digits.

</generic_descriptor>
```