Martin Janzen
Senior Software Architect

AN 104
Monitoring of Proprietary System Information

The work of standards bodies such as DVB, ATSC, ISO has improved the interoperability of complex video systems by establishing a common set of rules and protocols for differing devices to communicate with each other. However, the rapid advance of technology and new features introduced by vendors to establish competitive advantage often mean development of standards lag behind real world equipment – resulting in incompatibilities between different vendors' equipment.

In particular, numerous proprietary additions to MPEG Systems Information (DVB-SI or ATSC PSIP) have resulted in incompatibilities. This paper presents strategies for the preventative monitoring of those proprietary systems.

Background

DVStation provides monitoring for RF, MPEG-2 transport stream, and content within an easy-to-use and integrated environment. However, while international standards bodies are working on forming a unified set of operating protocols for digital broadcasting, still national and vendor-specific variations pose problems to consistent operational monitoring.

The MPEG transport stream contains information about the structure of programs and services, and their characteristics, in a series of tables which are contained within specific, assigned PIDs.

ISO, within the core MPEG standard, defines the basic tables, while other regional or national bodies, such as DVB define additions to that core set. Additionally, vendors frequently implement their own, proprietary extensions to provide additional services such as interactive television, encryption, etc.

Monitoring versus Testing

Test equipment is designed to answer the question: "does this broadcast match the international standards?". On the other hand, a preventative monitoring tool must answer the question: "does this broadcast meet my broadcast standards?" – while taking into account regional and vendor-specific differences with the international standards.

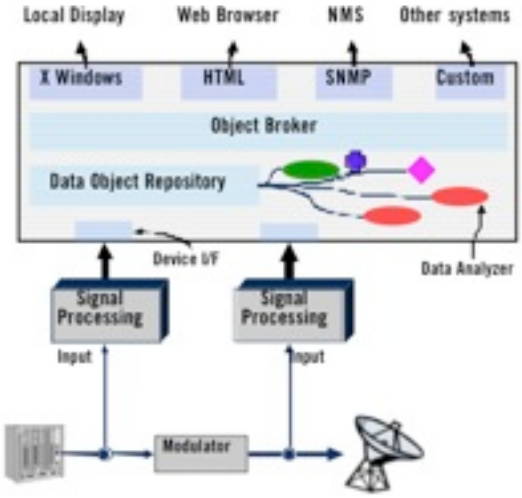
To support this end, DVStation's modular architecture has been designed to quickly support proprietary standards. System information tables, specifically, are accommodated through Pixelmetrix's unique, data-driven table decode and parsing engine.

Using a text file in XML notation, the system can quickly be customized to support different table structure variants – all while maintaining concurrent support of the core protocol standards: MPEG, DVB-SI, ATSC, and ISDB.

This application note explains the table decode engine of DVStation and provides examples on creating customized table decodes.

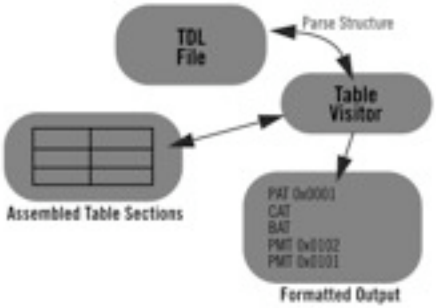
DVStation Software Architecture

DVStation employs a multi-process architecture, dividing tasks between user/program interfacing and data collection, correlation, and storage.



Internal system communication is based on CORBA, allowing additional software processes and/or control interfaces to be added – either locally or via a remotely located computer attached via LAN or WAN.

Individual table sections are identified by the hardware processor cards and forwarded to the host software for assembly, parsing, and processing.

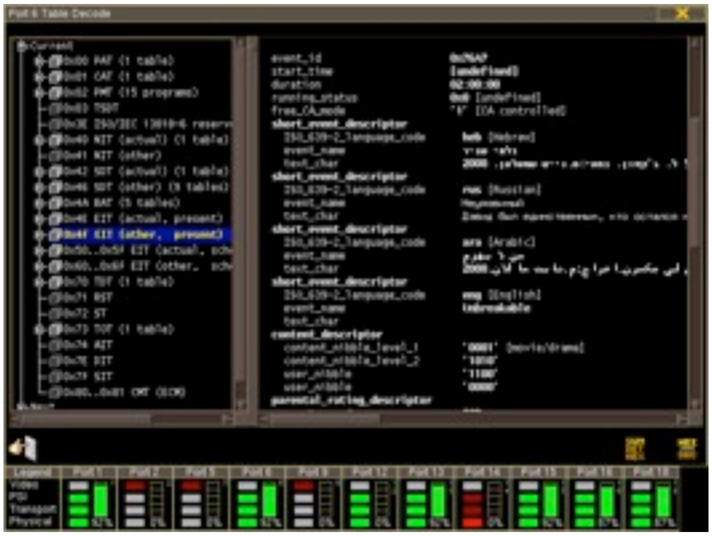


Pixelmetrix corporation

TDL Overview

What is the Table Definition Language, anyway?

The purpose of the TDL is to tell the DVStation's table parser how to interpret table sections received over an MPEG2 transport stream. Currently this information is used by the DVStation's graphical user interface, which presents the details of a table section in the right hand side of the Table Decode window.



Also, it is used internally by the DVStation's host software to determine which PIDs carry packet elementary streams, system information (i.e. table sections), and PCR values. It can also be used to extract information about networks, services, etc.

Why is it needed?

The functions described above could be carried out by specialized code handwritten for each kind of table section, with no need for a table definition language of any kind. However, there are many advantages to be gained by separating the description of the table sections from the DVStation code:

- Writing TDL files is easy and fast. This makes it easier to keep up to date with changes to existing standards, and to add new support for functionality such as data broadcasting.
- Writing TDL files is less error-prone than writing specialized codes. The table parser code can be written, debugged, and tested once; addition of new kinds of table sections does not require additional system or regression testing.
- TDL enables easy customization. For instance, TDL files can be written to describe vendor specific user private table sections. Customized TDL files can augment the set of standard table sections understood by the DVStation's table parser, or they can modify or replace the standard definitions entirely.
- One doesn't have to be a C++ programmer to use TDL. Therefore, Pixelmetrix field personnel, and DVStation customers themselves, can create TDL files and install them without requiring assistance from the Pixelmetrix R&D group. This allows for a much faster response to customers' requirements.

TDL Syntax

What does TDL look like?

TDL is based on the Extensible Markup Language, or XML. This is a subset of SGML, the Standard Generalized Markup Language, and is intended to be used for defining new languages using a standardized syntax. In practice, XML looks very much like the Hypertext Markup Language (HTML), which will be familiar to anyone who has worked on web page design.

TDL files are encoded using UTF8, a Unicode transformation format which encodes standard English characters exactly as in ASCII, while allowing "escaped" versions of Unicode characters to be used for languages other than English. This means that a standard text editor can be used to create and modify TDL files.

This flexibility makes for a truly international tool – since the text descriptions of the various elements need not be restricted to English, but could be displayed in any roman or non-roman alphabet and characters such as Japanese, Chinese, Arabic, Greek.

TDL consists of a set of elements, including data elements such as numeric fields, character strings, and time values, as well as control elements such as loops and conditional operations.

Each element consists of a beginning <tag> and an ending </tag>. If no other elements are required between the beginning and ending tags, a single <tag/> will do.

Each tag may also contain attributes, some of which are required and some optional. Attributes are specified within the beginning tag of an element, as, for instance, <tag attr1="xxx">, and are always enclosed in double quotes, even if they represent a numeric value.

Finally, a TDL file may contain comments, consisting of text enclosed by <!-- and > marks. Comments may span multiple lines.

Whitespace, such as spaces, tab characters, and newlines, is not significant except between quotes or within the text description of an ident element (see below).

The set of data element tags provided by TDL has been chosen so as to correspond as closely as possible to the format used in the core MPEG-2 specification [1] and related documents. This makes it relatively straightforward to create TDL files describing new table sections, additional descriptors, enumerated type values, and so on.

Installing Custom Table Decodes

TDL files for all supported standards – MPEG, DVB, ATSC, and ISDB – are installed on the built-in hard disk of each DVStation.

Modifying these files, or adding new files, means the DVStation immediately supports the new standard.

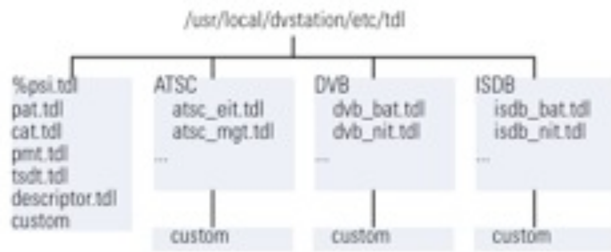
TDL files describing the supported protocols are contained within the /usr/local/dvstation/etc/tdl directory on every DVStation.

A TDL Example

The easiest way to understand the syntax of a TDL file is by looking at a typical example. The illustration at right shows the MPEG-2 Program Attribute Table section, as described in ITU-T Recommendation H.222.0, Section 2.4.4.3 [1].

This table section contains a set of initial header fields, followed by a repeating group of fields beginning with program_number, and finally a single CRC_32 field. Within the repeating group of fields, either a network_PID or a program_map_PID field, but not both, may be present, depending on the value of program_number.

*Note that the *for*, *if*, and *else* lines as well as the opening and closing braces look very much like C or C++ code. However, the value of N is not specified anywhere, nor is it defined. In fact, the ITU-T descriptions simply use this notation to indicate that the group of fields is to be repeated until the end of the received table section (minus the length of the ending CRC_32 field). It should be considered as a sort of pseudocode, rather than being taken completely literally.




```
Syntax                                  bits   Mnemonic

program_association_section()
{
   table_id                             8      uimsbf
   section_syntax_indicator             1      bslbf
   '0 '                                 1      bslbf
   reserved                             2      bslbf
   section_length1                      2      uimsbf
   transport_stream_id                  16     uimsbf
   reserved                             2      bslbf
   version_number                       5      uimsbf
   current_next_indicator               1      bslbf
   section_number                       8      uimsbf
   last_section_number                  8      uimsbf
   for (i = 0; i < N; i++)
   {
      program_number                    16     uimsbf
      reserved                          3      bslbf
      if (program_number == '0')
      {
      network_PID                       13     uimsbf
      }
      else
      {
         program_map_PID                13     uimsbf
      }
   }
   CRC_32                               32     rpchof
}
```

TDL Format

Now, here is how this same table section would be described using the Table Definition Language (for clarity, property elements have been removed from this example).

The first three lines appear at the beginning of each TDL file. They indicate, first, that we are using XML version 1.0 and that the characters in this TDL file use the Unicode UTF-8 encoding. They also state that this file follows the TDL syntax as specified in the dvtdl.dtd Document Type Definition file. This enables the DVStation host software to check TDL files for invalid elements or attributes, and various other syntactic errors.

```
<?xml version="1.0" encoding="UTF8" standalone="no"?>
<!DOCTYPE dvtdl SYSTEM "dvtdl.dtd">
<dvtdl xmlns="http://www.pixelmetrix.com/dtd/dvtdl.dtd">
```

This is followed by some comment lines which tell the reader exactly what is described by this TDL file. It is very important to include the name of the standard or the vendor documentation which contains the original definition, as well as the version of that document which was used when creating the TDL file.

```
<!- MPEG2 Program Association Table -->
<!- Section 2.4.4.3, Table 2-25 >
```

The section element tells the table parser when to use this definition. In this case, it indicates that this is a definition of a table section which is applicable when the key field (the table_id) has a value of 0x00. Furthermore, this table section should only be carried on PID 0.

This is followed by the header fields. The correspondence between the ITU-T Recommendation and the TDL file should be clear.

Next, we have a loop element, which indicates that the elements between the <loop> and </loop> tags are to be repeated until the end of the table section is reached, less an offset of 32 bits (which is the length of the final CRC_32 field).

Within the loop, the switch, case, and default elements enable us to tell the table parser to interpret the next part of the table section as one of two different groups of fields, depending on the value of the program_number field. The meaning of these elements is similar to that of the C or C++ constructs of the same names: The fields in the first case element whose value attribute matches the value of the field specified in the switch element are used. If no case element matches the field's value, the default element is used instead.

Finally, after the loop body we have the CRC_32 field, followed by the remaining end tags.

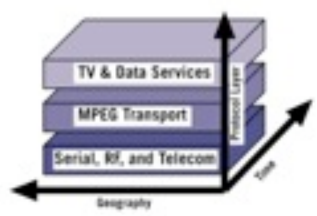
```
<section group="table_section" key="0x00" allowedPIDs="0x0000"
name="program_association_section" abbrev="PAT">
   <field  length="8" type="uimsbf" name="table_id" enum="table_id_assignments"/>
   <field  length="1" type="bslbf" name="section_syntax_indicator"/>
   <constant length="1" type="bslbf" value="0"/>
   <reserved length="2"/>
   <field  length="12" type="uimsbf" name="section_length"/>
   <field  length="16" type="uimsbf" name="transport_stream_id"/>
   <reserved length="2"/>
   <field  length="5" type="uimsbf" name="version_number"/>
   <field  length="1" type="bslbf" name="current_next_indicator"/>
   <field  length="8" type="uimsbf" name="section_number"/>
   <field  length="8" type="uimsbf" name="last_section_number"/>
   <loop endOffset="32">
      <field  length="16" type="uimsbf" name="program_number"/>

      <switch field="program_number">
         <case value="0">
            <reserved length="3"/>
            <pid   length="13"    name="network_PID"/>
         </case>
         <default>
            <reserved length="3"/>
            <pid   length="13"    name="program_map_PID"/>
         </default>
      </switch>
   </loop>

   <field length="32" type="rpchof" name="CRC_32"/>
</section>
</dvtdl>
```

About DVStation

Pixelmetrix has focused on creating a single self-contained monitoring station that can analyze thousands of parameters within hundreds of digital television signals. Through the use of plug-in modules and parallel processing, we monitor all these parameters in real time, simultaneously and continuously. We've targeted our development efforts to insure the quality of the signal, the integrity of the program service and the delivery of essential technical information to the right people in a timely and meaningful manner.



Our engineers began with a simple premise: effective monitoring of digital television networks - just as with telecom networks - requires the use of real-time, continuous and simultaneous evaluation of hundreds of points along the transmission chain. To receive this necessary network intelligence, adequate data collection, analysis and correlation is needed on three axis – time, layer and geography. Monitoring of all layers – physical, transport, coding, and quality – is essential for a complete maintenance picture.

Plug-in modules allow flexibility and accommodate changes in a fast evolving technical infrastructure. So far, we've focused on three categories of plug-in modules: physical line interfaces (ASI, SPI, RF, ATM etc.), a transport stream processor (TSP), and picture quality processors.

In our design, a line interface module extracts the MPEG-2 transport stream from the native RF or telecom signals and passes that data to a TSP – Transport Stream Processor. Line interface modules provide monitoring capability on the physical layer. For RF interfaces (QPSK, QAM, COFDM, 8VSB, etc.) monitoring means to check carrier level, C/N (carrier-to-noise ratio), bit error rate and EVM (Error Vector Magnitude), or other parameters that may be applicable. Additionally, a simple constellation diagram indicates overall modulation health.

Our ATM interface connects to a 155 Mb/s optical fiber and extracts MPEG transport streams from several VP/VCs (virtual path/virtual circuit). In addition to this basic functionality, the interface detects physical layer errors and parameters with the optical and Sonet/SDH signals.

Abbreviations

bslbf	bit string, leftmost bit first
DTD	Document Type Definition
GUI	Graphical User Interface
HTML	Hypertext Markup Language
PAT	Program Attribute Table
PCR	Program Clock Reference
PES	Packetized Elementary Stream
PID	Packet Identifier
PMT	Program Map Table
PSI	Program and System Information
rpchof	reverse polynomial coefficients, high order first
SI	System Information
tcimsbf	two's complement integer, most significant byte first
TDL	Table Definition Language
uimsbf	unsigned integer, most significant byte first
XML	Extensible Markup Language

References

"Measurement Guidelines for DVB Systems", Draft TR 101 290, DVB.

[1] ITU-T Recommendation H.222.0 (07/95) | ISO/IEC 13818-1, "Information Technology - Generic Coding of Moving Pictures and Associated Audio Information: Systems".

[2] World Wide Web Consortium XML Home Page, http://www.w3c.org/xml.

[4] Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB Systems, ETSI EN 300 468, V1.3.1 (1998-02).

[5] Program and System Information Protocol for Terrestrial Broadcast and Cable, ATSC Doc. A/65., 23 Dec 1997.

[6] Service Information for Digital Broadcasting System, ARIB Std-B10, 1.3.

Key Customers

Pixelmetrix enjoys an equal distribution of customers among the world's geographic regions. We have product deployed on all seven continents, including Antarctica.

Key clients of Pixelmetrix include:

- Arqiva
- BSD
- Broadcast Australia
- Ceske Radiokomunikace
- Itelazpi
- KPN
- LA7
- Network TEN
- Raiway
- TDF
- Towercast
- TVNZ, and others

About Pixelmetrix

Pixelmetrix Corporation is the global expert in Preventive Monitoring for digital, cable and IPTV networks. The company provides equipment and network intelligence systems to television broadcasters for the management and monitoring of quality of service and quality of experience. Headquartered in Singapore, Pixelmetrix has offices in the United States and Europe.

Pixelmetrix has been conferred the Frost & Sullivan Industrial Technologies Award 2009, C+T Technology Development Award 2009, Engineering & Technology Emmy® Award 2007, Broadcast Engineering publication Pick Hit Award 2005 and 2008, TV Technology publication STAR Awards (Superior Technology Award Recipient) 2000, 2004 and 2007, BIRTV Product of the Year Award 2006, Cable-Satellite/Mediacast Product of the Year Awards 2003 and 2004, as well as the Peter Wayne Award 2000, for Best Design and Innovation.

For More Information

To learn more about the DVStation, request a demo, or learn how Pixelmetrix might help you optimize video network integrity, contact us today!

Pixelmetrix Corporation

31 Kaki Bukit Road 3
#07-03 Techlink
Singapore 417818

Tel: +65 6547 4935
Fax: +65 6547 4945

email: info@pixelmetrix.com

North American Sales & Support

10097 Cleary Boulevard
Suite #114 Ft. Lauderdale
Florida 33324, USA 33324

Tel: 954 472 5445
Fax: 954 472 6989

European Sales & Support

Affolternstrasse 47a
8913 Ottenbach
Switzerland

Tel: +41 56 641 0317
Fax: +41 56 500 0161