BBC

R&D White Paper

WHP 037

August 2002

TPEG Library v2.0 overview

T. Ferne

Research & Development
BRITISH BROADCASTING CORPORATION

BBC Research & Development
White Paper WHP 037

TPEG Library v2.0 Overview

Tristan Ferne

Abstract

This document gives an overview of the BBC's TPEG C++ Library version 2.0. TPEG (Transport Protocol Experts Group) is a byte-stream protocol that can carry a range of data – traffic and travel information in particular. The BBC TPEG library is designed to provide platform independent C++ classes that can be used to implement a complete TPEG system from the service providers' play-out software to TPEG receivers. The library is freely available from the BBC for development purposes.

This document is included in the BBC TPEG C++ Library distribution.

BBC Research & Development
White Paper WHP 037

TPEG Library v2.0 Overview

Tristan Ferne

Contents

1 Introduction 1
1.1 TPEG – Travel Protocol Experts Group 1
2 The BBC TPEG C++ Library 1
2.1 Use of the library 2
2.2 Application source 2
2.3 Frame multiplexer 2
2.4 Decoder 3
3 BBC TPEG C++ Library API 4
3.1 Introduction 4
3.1.1 General library information 4
3.2 TPEG Frames 4
3.2.1 TPEG_Frame 5
3.2.2 TPEG_TransportFrame extends TPEG_Frame 5
3.3 TPEG_ServiceFrame extends TPEG_Frame 5
3.4 TPEG_ServiceFrameNormal 5
3.4.1 TPEG_ServiceFrameStreamDirectory 5
3.4.2 TPEG_ComponentFrame extends TPEG_Frame 5
3.4.3 TPEG_RTMComponentFrame extends TPEG_ComponentFrame 5
3.4.4 TPEG_SNIComponentFrame extends TPEG_ComponentFrame 5
3.5 Road Traffic Messages (RTM) 7
3.5.1 TPEG_RTM 7
3.5.2 TPEG_RTMComponent 8
3.5.3 TPEG_RTMTopLevelComponent 8
3.5.4 Top-level components extends TPEG_RTMTopLevelComponent 8
3.5.5 RTM components extends TPEG_RTMComponent 9
3.5.6 LOC components extends TPEG_RTMComponents 10
3.5.7 TPEG_RTMXMLParser 10
3.5.8 TPEG_RTMDatabase 10
3.5.9 TPEG_RTMList 10
3.5.10 Data tables 10
3.6 Service and Network Information (SNI) 12

3.6.1 TPEG_SNIComponent 12
3.6.2 TPEG_SNIContentDescriptionComponent extends TPEG_SNIComponent 12
3.6.3 TPEG_SNIContentDescriptionComponentEntry 12
3.6.4 TPEG_SNICurrentServiceInformationComponent extends TPEG_SNIComponent 13
3.6.5 TPEG_SNIFastTuningComponent extends TPEG_SNIComponent 13
3.6.6 TPEG_SNIFastTuningComponentEntry 13
3.6.7 TPEG_SNIFreeTextComponent extends TPEG_SNIComponent 13
3.6.8 TPEG_SNIGeographicalCoverageComponent extends TPEG_SNIComponent 13
3.6.9 TPEG_SNIGeographicalCoverageComponentEntry 13
3.6.10 TPEG_SNIHelpInformationComponent extends TPEG_SNIComponent 13
3.6.11 TPEG_SNIServiceTableAccleratorComponent extends TPEG_SNIComponent 13
3.6.12 TPEG_SNITimeScheduleComponent extends TPEG_SNIComponent 13
3.6.13 TPEG_SNITimeScheduleComponentEntry 13
3.6.14 TPEG_SNITimeSlot 14
3.6.15 TPEG_SNIComponentList 14
3.6.16 TPEG_SNIDatabase 14
3.6.17 TPEG_SNIServiceComponentSI 14
3.7 General 15
3.7.1 TPEG_CRC 15
3.7.2 TPEG_DateTime 15
3.7.3 TPEG_Exception 15
3.7.4 TPEG_RTMDecodeException extends TPEG_Exception 15
3.7.5 TPEG_ParseException extends TPEG_Exception 15
3.7.6 TPEG_OutOfRangeException extends TPEG_Exception 15
3.7.7 TPEG_EncodeException extends TPEG_Exception 15
3.7.8 TPEG_OperatingTime 15
3.7.9 Data types 15
3.7.10 Conversion functions 15
4 Examples of use 16
4.1 RTM Generator and Multiplexer 16
4.2 RTM Receiver 17
5 References 19
6 Contacts 19
7 Appendix 1 - Known problems and limitations 20
7.1 Memory leaks 20
7.2 Unsupported parts of the specification 20
7.2.1 SNI 20
7.2.2 RTM 20
7.2.3 LOC 20
7.2.4 PTI 20
8 Appendix 2 – Installation 21

8.1 Archive structure and contents 21
8.2 Linux installation 21
8.3 Windows installation 21
8.3.1 libxml 22
8.3.2 Bison and Lex 22

White Papers are distributed freely on request.

Authorisation of the Chief Scientist is required for publication.

BBC Research & Development
White Paper WHP 037

TPEG Library v2.0 Overview

Tristan Ferne

1 Introduction

1.1 TPEG – Travel Protocol Experts Group

TPEG is an open specification [1, 2, 3 & 4] for a byte-stream protocol that can carry a range of data – traffic and travel information in particular. The basic framing structure is defined as a stream of data containing transport frames. Each transport frame then contains several component frames. Component frames store data for a particular application. The specification includes three types of application/component frame; Service and Network Information (SNI), Road Traffic Message (RTM) and Public Transport Information (PTI).

For road traffic information the RTM application has been defined. A single RTM (Road Traffic Message) contains various fixed data fields, such as event severity and message generation time, together with any number of RTM Components. RTM Components define things such as accidents, locations (i.e. longitude and latitude) or roadworks and can contain fixed data fields and other RTM Components giving more detailed information (e.g. vehicle information). All non-numerical data fields (e.g. vehicle type) are indices into defined tables of possible values. Because of this language independence, receivers can easily decode messages into text, speech or graphics.

The Service and Network Information (SNI) application carries information about a particular TPEG service and the service components carried in that service.

The BBC currently has a TPEG service containing RTM data for traffic and travel information in the UK. The TPEG stream is broadcast over the BBC's DAB digital radio network and provided on the Internet.

2 The BBC TPEG C++ Library

This document describes the API of the BBC TPEG C++ library (version 2.0), it assumes you have the corresponding version of the TPEG specification [1, 2, 3 & 4]. The current version of the library supports the following TPEG specifications:

BPN 027-2	TPEG Part 2 - Syntax, Semantics and Framing Structure	TPEG - SSF_3.0/001
BPN 027-3	TPEG Part 3 - Service and Network Information	TPEG - SNI_3.0/001
BPN 027-4	TPEG Part 4 - Road Traffic Message Application	TPEG - RTM_3.0/003
BPN 027-6	TPEG Part 6 – Location referencing for applications	TPEG – LOC_1.0/003

The library provides platform-independent classes needed for the basic Transport, Service and Component Frames, for supporting the SNI (Service and Network Information), for the TPEG RTM

(Road Traffic Message) application and for the TPEG LOC (Location referencing) application. The library also supports the parsing and generation of tpegML – the XML representation of TPEG [5, 6 & 7]. There are also some utility classes for operations such as converting data types. See 7.2 for unsupported parts of the specifications.

2.1 Use of the library

The library is designed to be used to create three main modules, an application source, a frame multiplexer and a decoder. Figure 1 shows the TPEG chain; Application sources provide component frames for specific applications (currently only the RTM application is supported) to the multiplexer. The multiplexer then creates a complete TPEG stream from these. The decoder receives the TPEG stream, demultiplexes the frames and decodes the application data.

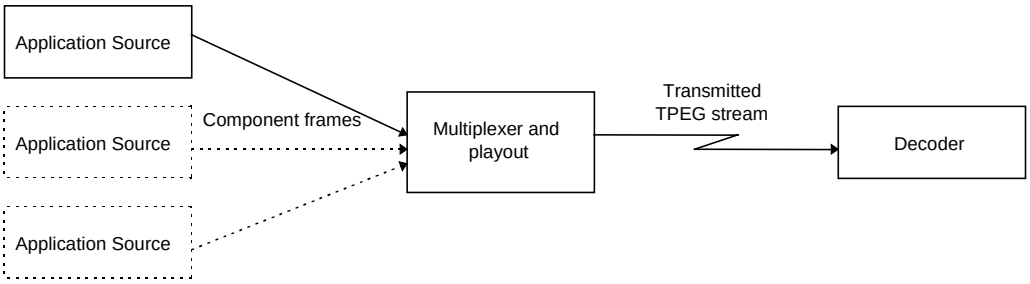

Figure 1 - TPEG system chain

2.2 Application source

This module creates component frames containing application data. It can be separate from, or integrated into the frame multiplexer. The library can be used to create an RTM application source that creates RTMComponentFrames by parsing tpeg-rtmML (an XML description of road traffic messages) files containing RTM messages.

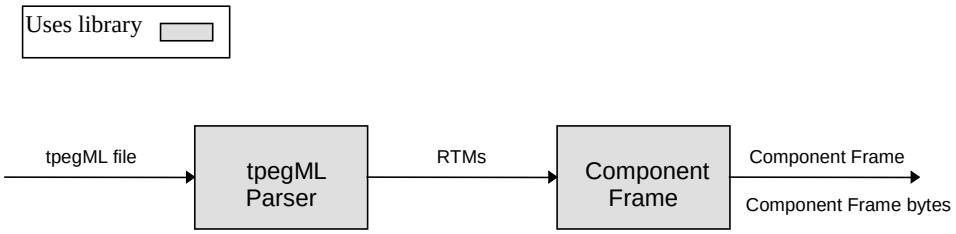

Figure 2 - RTM application source

Figure 2 shows an RTM application source. The RTM file parser parses tpegML (an XML description of road traffic messages) files containing RTM messages and creates a list of RTM objects. This list is then stored in the RTMComponentFrame. The ComponentFrame object can be passed to another part of the program, or it can be encoded into its byte stream representation.

2.3 Frame multiplexer

This module multiplexes component frames into TPEG transport frames. These are then sent to a playout application that plays out the final TPEG stream. It also creates the necessary SNI (Service and Network Information) component frames.

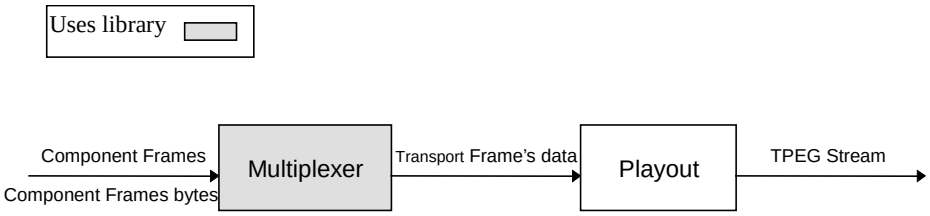

Figure 3 - Multiplexer and playout

The multiplexer takes in component frame data or ComponentFrame objects, possibly from multiple sources. These are then added to a TransportFrame object. SNIComponentFrames are created, to provide service information about the service and its components, and added to the TransportFrame. The TransportFrame then provides the encoded TransportFrame byte data to the playout software that cyclically plays out this data to create the TPEG stream.

2.4 Decoder

This module receives a TPEG stream, demultiplexes it and decodes and displays the application. The library provides the frame demultiplexing, and for the RTM application there are RTM decoding functions as well as data structures for holding decoded RTMs. The library also includes a simple RTM message database and methods to render messages as text strings.

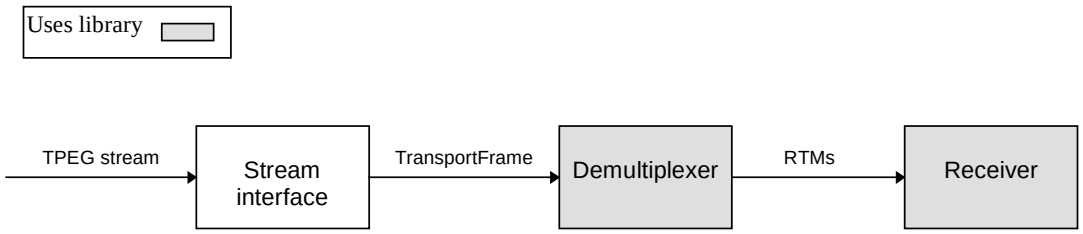

Figure 4 - RTM Decoder

The stream interface receives a TPEG stream, synchronises with it and extracts raw transport frame data. This data is passed to a TransportFrame object that demultiplexes the data and creates ComponentFrame objects. An RTMComponentFrame parses the data to create RTM objects which are then passed to a receiver module that manages the received RTMs and provides the user interface.

3 BBC TPEG C++ Library API

3.1 Introduction

This section describes the contents of the BBC TPEG C++ library. It is grouped into sections of related classes. Detailed documentation of each class and its methods and fields are provided separately in HTML form.

3.1.1 General library information

General rules used in the library are:

All classes in the library are named with the prefix ‘TPEG_’

Header files have the extension ‘.h’ and implementation files have ‘.cpp’

Any classes that use ‘new’ define a copy constructor and an assignment operator. Any internal objects may be copied by reference or by value depending on the class (also see the next point).

Classes that use 'new' to create other objects internally should also handle their destruction. Note that some classes may or may not create objects internally depending on how the class is being used. These classes therefore keep a flag indicating how their internal objects were allocated and will destroy internal objects when appropriate.

3.2 TPEG Frames

These classes represent the framing structures used in TPEG.

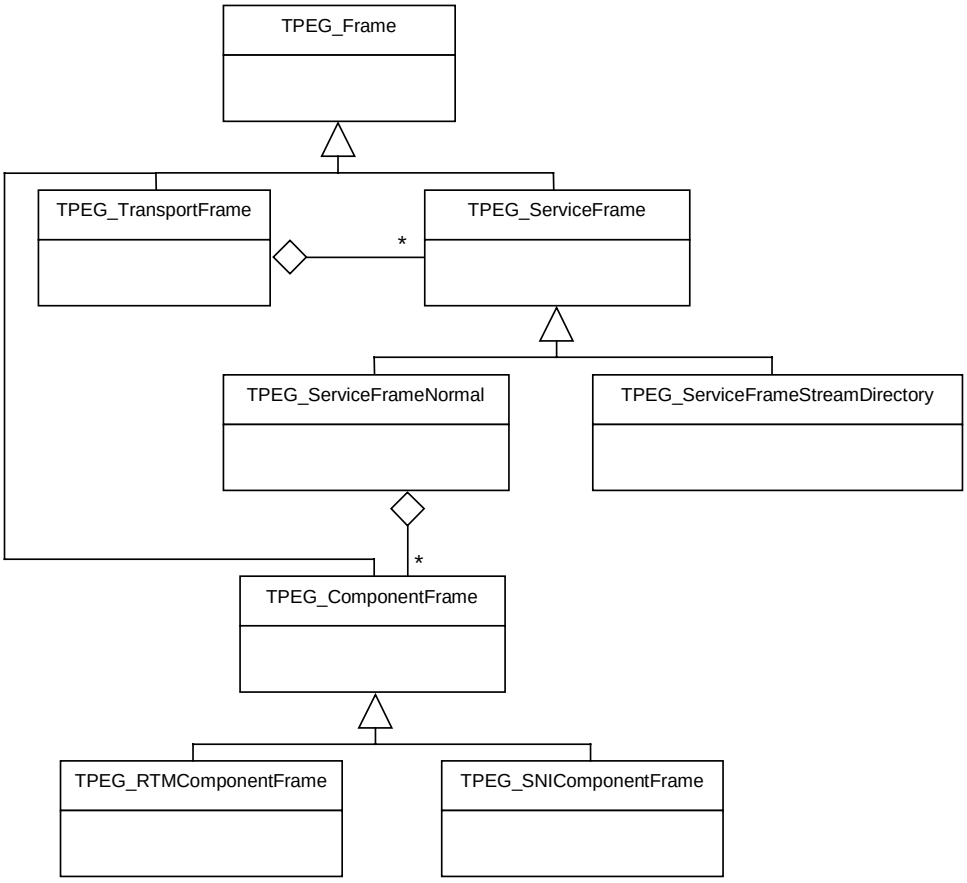

Figure 5 - TPEG Frames

3.2.1 TPEG_Frame

This is an abstract base class for all TPEG frames. Virtual methods that should be overridden by subclasses include:

parse(): Parse byte data to create internal data and objects (e.g. in a decoder).

encode(): Encode internal data and objects to create byte data (e.g. in the playout software).

3.2.2 TPEG_TransportFrame extends TPEG_Frame

This represents a TPEG transport and service frame. It overrides parse() and encode() from TPEG_Frame. It contains either a TPEG_ServiceFrameNormal or a TPEG_ServiceFrameStreamDirectory. The TPEG_ServiceFrames that are referenced in this will:

- Be included in the transport frame data when encoded.
- Have their data parsed when decoding a transport frame's data. Any service frames that are in the data for the transport frame but are not referenced will not be decoded.

3.3 TPEG_ServiceFrame extends TPEG_Frame

This is an abstract base class for TPEG_ServiceFrameNormal and TPEG_ServiceFrameStreamDirectory.

3.4 TPEG_ServiceFrameNormal

This represents a service frame containing conventional data. It contains a list of TPEG_ComponentFrames. TPEG_ComponentFrames that are referenced in this list will:

- Be included in the service frame data when encoded.
- Have their data parsed when decoding a service frame's data. Any component frames that are in the data for the service frame but are not referenced will not be decoded.

3.4.1 TPEG_ServiceFrameStreamDirectory

This represents a stream directory which contains information about what TPEG services are present in the current stream.

3.4.2 TPEG_ComponentFrame extends TPEG_Frame

This is an abstract base class for TPEG_ComponentFrames.

3.4.3 TPEG_RTMComponentFrame extends TPEG_ComponentFrame

This represents a TPEG RTM (Road Traffic Messages) component frame. It contains a list of TPEG_RTMs that may be created internally (in parse()) or set externally (e.g. in a content generator). The parse() method sets the frame's data fields and internally creates any RTMs carried in the frame. parse() is then called on these RTMs. Note that these internally created RTMs are appended to the internal RTM list.

If the TPEG_RTMs have been created internally then the destructor will destroy them. If the TPEG_RTMs have been added from outside then they should be destroyed externally.

3.4.4 TPEG_SNIComponentFrame extends TPEG_ComponentFrame

This represents a TPEG SNI (Service and Network Information) component frame. It contains a list of TPEG_SNIComponents that can be created (in parse()) or set externally (e.g. in a content generator). The parse() method sets the frame's data fields and internally creates any SNI components carried in the frame. Parse() is then called on these SNI components. Note that these internally created SNI components are appended to the internal list.

If the TPEG_SNIComponents have been created internally then the destructor will destroy them. If the TPEG_SNIComponents have been added from outside then they should be destroyed externally.

3.5 Road Traffic Messages (RTM)

These classes support the TPEG Road Traffic Message application. This application is intended to provide road users with information related to events on and status of the road network.

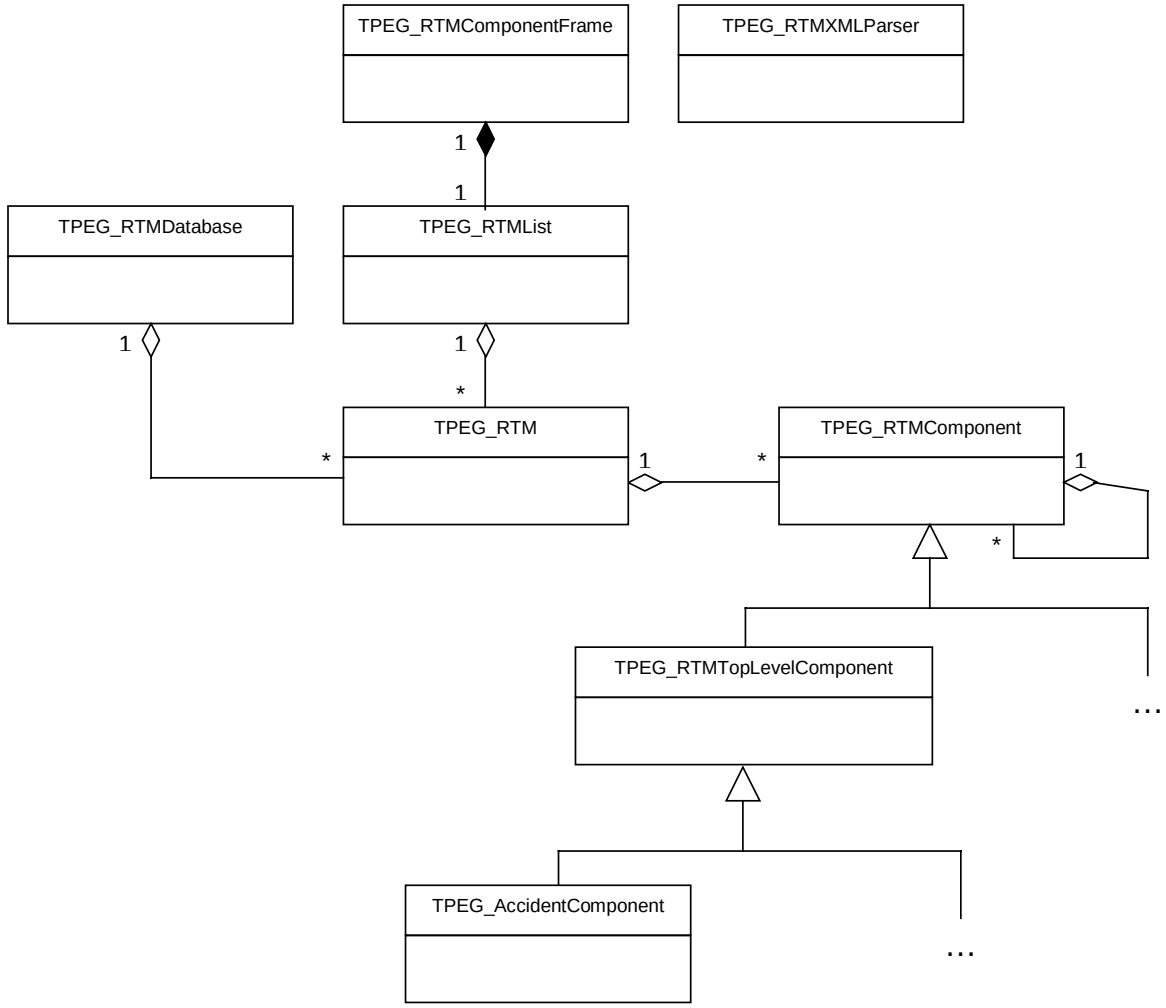

Figure 6 - RTM Classes

3.5.1 TPEG_RTM

This represents a TPEG Road Traffic Message. It contains several message information fields and a list of top-level TPEG_RTMComponents. A TPEG_RTM can also represent a delete message. Methods include:

parse(): Parse byte data to fill data fields and create RTM components.

encode(): Encode data fields and RTM components to create byte data.

decode(): Decodes raw data of RTM components after RTM file parser.

toString(): Returns string representing this RTM.

toXML(): Returns string representing this RTM in tpeg-rtmML (an XML description of road traffic messages).

getDescription(): Returns string presenting a limited English language version of the RTM.

The TPEG_RTMComponents may be created internally (in parse()) or set externally (e.g. in a content generator). If the TPEG_RTMComponents have been created internally then the destructor

will destroy them. If the TPEG_RTMComponents have been added from outside then they should be destroyed externally.

3.5.2 TPEG_RTMComponent

This is the base class for all the RTM components. It contains a list of TPEG_RTMComponents (for sub-components). Virtual methods include:

parse(): Parse byte data to create internal data and sub-components.

encode(): Parse internal data and sub-components to create byte data.

decode(): RTM file parser sets raw bytes - this method decodes these into actual data fields.

toString(): Returns string representing this RTM component.

toXML():Returns string presenting a limited English language version of the RTM.

toDescription(): Returns string representing this RTM component in a format used by the BBC Digital Travel Assistant and TPEG email alerts.

There are sub-classes of this class for all the RTM components defined in the TPEG specification (see below) and these all implement the methods shown above.

The sub-components may be created internally (in parse()) or set externally (e.g. in a content generator). If the TPEG_RTMComponents have been created internally then the destructor will destroy them. If the TPEG_RTMComponents have been added from outside then they should be destroyed externally.

3.5.3 TPEG_RTMTopLevelComponent

This class extends TPEG_RTMComponent to represent top-level components (i.e. those that are directly carried by an RTM message).

3.5.4 Top-level components extends TPEG_RTMTopLevelComponent

The following classes are top-level components (i.e. those components that an RTM can hold):

TPEG_AccidentComponent

TPEG_ActivitiesComponent

TPEG_DiversionAdviceComponent <unsupported>

TPEG_FacilitiesPerformanceComponent

TPEG_LocationContainerComponent

TPEG_MovingHazardComponent

TPEG_NetworkPerformanceComponent

TPEG_NetworkConditionsComponent

TPEG_NonRepetitiveTimeComponent

TPEG_ObstructionsComponent

TPEG_PublicTransportInformationComponent

TPEG_RepetitiveTimeComponent

TPEG_RoadConditionsComponent

TPEG_SecurityAlertComponent

TPEG_VisibilityComponent

TPEG_WeatherComponent

3.5.5 RTM components extends TPEG_RTMComponent

The following classes are components used within other components only (i.e. not top-level components) to describe events:

TPEG_ActivityComponent

TPEG_AdhesionComponent

TPEG_AdviceComponent

TPEG_AnimalComponent

TPEG_AnimalInformationComponent

TPEG_AnimalProblemComponent

TPEG_ConditionStatusComponent

TPEG_DelayComponent

TPEG_LengthComponent

TPEG_LightingComponent

TPEG_MarkingComponent

TPEG_ObjectComponent

TPEG_ObjectProblemComponent

TPEG_ObscurityComponent

TPEG_PeopleComponent

TPEG_PerformanceComponent

TPEG_PositionComponent

TPEG_PrecipitationComponent

TPEG_RegulationComponent

TPEG_RestrictionComponent

TPEG_RoadConditionsComponent

TPEG_RoadsideAssistanceComponent

TPEG_RoadsideServicesComponent

TPEG_RoadworksComponent

TPEG_SpeedComponent

TPEG_SurfaceComponent

TPEG_TemperatureComponent

TPEG_TrafficControlComponent

TPEG_TravelTimeComponent

TPEG_VehicleComponent

TPEG_VehicleInformationComponent

TPEG_VehicleProblemComponent

TPEG_VisualAcuityComponent

TPEG_WindComponent

3.5.6 LOC components extends TPEG_RTMComponents

LOC components describe the location of an event. These are used inside the TPEG_LocationContainer component.

Note that the library only supports the location co-ordinates container and not the location descriptions container [4].

TPEG_DescriptorComponent

TPEG_DirectionTypeComponent

TPEG_ExpansionComponent

TPEG_HeightComponent

TPEG_LanguageComponent

TPEG_LocationCoordinatesComponent

TPEG_ModeOfTransportComponent

TPEG_ModeTypeListComponent

TPEG_WGS84Component

3.5.7 TPEG_RTMXMLParser

This class parses tpegML files (the XML representation of TPEG [5, 6 & 7]) to create a list of RTMs. It is used to generate RTM content. It parses the XML file using "libxml" – the XML C library for Gnome [8]. A validating parser is used to check the validity of the XML files against the appropriate DTDs. Any errors that are present in the file but pass the validity check (e.g. out of range values) will cause an error to be thrown. The parsing of the affected RTM will be halted but parsing of subsequent messages will be unaffected.

3.5.8 TPEG_RTMDatabase

This class provides a simple hashtable/map that stores RTM messages. When passed a message it will add, update or delete it from the database. A method is also provided that will delete any messages that have passed their expiry time.

3.5.9 TPEG_RTMList

This class provides an expandable list that holds RTM messages.

3.5.10 Data tables

The header file 'TPEG_RTMTables.h' contains the tables of text values for RTM data fields. Each table contains an array of strings, a table length and a default string.

The header file ‘TPEG_LOCTables.h’ contains the tables of text values for LOC data fields. Each table contains an array of strings, a table length and a default string.

3.6 Service and Network Information (SNI)

These classes support the provision of Service and Network Information (SNI) for a TPEG service.

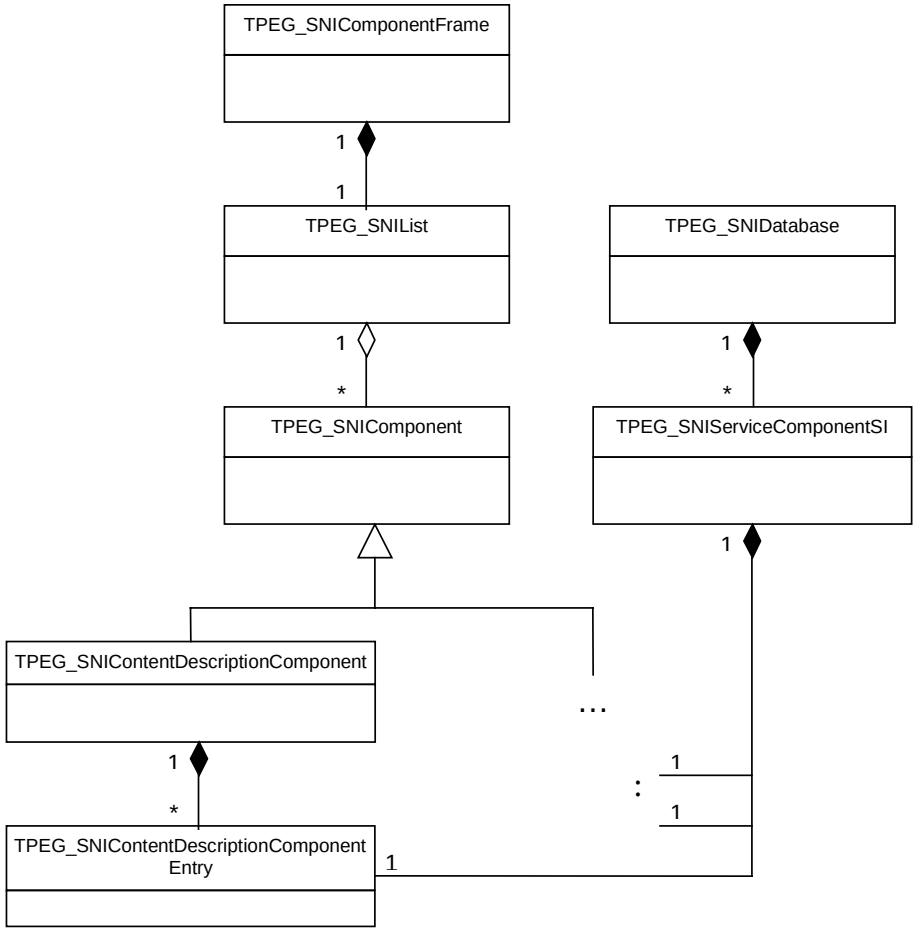

Figure 7 - SNI Classes

Note: The following SNI components are unsupported: service component reset, service logo, linkage information and subscriber information.

3.6.1 TPEG_SNIComponent

This is the base class for all Service and Network Information components. These contain information about the current service and the service components. Virtual methods include:

parse(): Parse byte data to fill data fields and create objects.

encode(): Encode data fields and objects to create byte data.

3.6.2 TPEG_SNIContentDescriptionComponent extends TPEG_SNIComponent

This class represents the Content Description GST (Guide to the Service Table). It contains a list of TPEG_SNIContentDescriptionComponentEntrys, each of which holds a description of a service component.

3.6.3 TPEG_SNIContentDescriptionComponentEntry

This class represents a single entry for a service component in the Content Description GST and contains a text description.

3.6.4 TPEG_SNICurrentServiceInformationComponent extends TPEG_SNIComponent

This class represents a Current Service Information component. It contains the name and description of the current TPEG service.

3.6.5 TPEG_SNIFastTuningComponent extends TPEG_SNIComponent

This class represents the Fast Tuning GST (Guide to the Service Table). It contains a list of TPEG_SNIFastTuningComponentEntrys, each of which holds service information for a service component.

3.6.6 TPEG_SNIFastTuningComponentEntry

This class represents a single entry for a service component in the Fast Tuning GST and may contain a service provider ID, service IDs, content ID, application ID, operating time and encryption indicator. This is used to associate a service component with a particular application type (e.g. RTM).

3.6.7 TPEG_SNIFreeTextComponent extends TPEG_SNIComponent

This class represents additional textual information about a service.

3.6.8 TPEG_SNIGeographicalCoverageComponent extends TPEG_SNIComponent

This class represents the Geographical Coverage Service Table. It contains a list of TPEG_SNIGeographicalCoverageComponentEntrys, each of which holds coverage information for a service component.

3.6.9 TPEG_SNIGeographicalCoverageComponentEntry

This class represents a single entry for a service component in the Geographical Coverage Service Table and contains the co-ordinates for the North-West and South-East corners of the coverage rectangle.

3.6.10 TPEG_SNIHelpInformationComponent extends TPEG_SNIComponent

This class provides help addresses about a service.

3.6.11 TPEG_SNIServiceTableAccleratorComponent extends TPEG_SNIComponent

This class represents the Service Table Accelerator component. It just contains the current version number of the SNI.

3.6.12 TPEG_SNITimeScheduleComponent extends TPEG_SNIComponent

This class represents the Time Schedule GST (Guide to the Service Table). It contains a list of TPEG_SNITimeScheduleComponentEntrys, each of which holds schedule information for a service component.

3.6.13 TPEG_SNITimeScheduleComponentEntry

This class represents a single entry for a service component in the Time Schedule GST and contains a TPEG_SNITimeSlot object.

3.6.14 TPEG_SNITimeSlot

This class represents a time slot for an application in a service component - it stores the start time, repetition details and duration of the application.

3.6.15 TPEG_SNIComponentList

This class provides an expandable list that holds SNI components.

3.6.16 TPEG_SNIDatabase

This class provides a simple store of service and network information for a single TPEG service. It contains SI (service information) about the service and for each service component with SI it stores a TPEG_SNIServiceComponentSI object. The database is filled by passing it TPEG_SNIComponents from which it extracts the SI for individual service components.

3.6.17 TPEG_SNIServiceComponentSI

This class represents the service information of a service component. An instance of this class holds up to four different SNI component entries for this service component: TPEG_SNIFastTuningComponentEntry, TPEG_SNITimeScheduleComponentEntry, TPEG_SNIGeographicalCoverageComponentEntry and TPEG_SNIContentDescriptionComponentEntry.

3.7 General

3.7.1 TPEG_CRC

This class represents and implements a 16-bit CRC for TPEG (CCITT CRC-16).

3.7.2 TPEG_DateTime

This class represents TPEG dates/times (stored as a 32-bit unsigned integer number of seconds).

3.7.3 TPEG_Exception

This is the base class for all TPEG exceptions.

3.7.4 TPEG_RTMDecodeException extends TPEG_Exception

This exception may be thrown when parsing XML.

3.7.5 TPEG_ParseException extends TPEG_Exception

This exception may be thrown when parsing byte data to create objects.

3.7.6 TPEG_OutOfRangeException extends TPEG_Exception

This exception may be thrown when setting data fields with out of range data.

3.7.7 TPEG_EncodeException extends TPEG_Exception

This exception may be thrown when encoding objects to byte data.

3.7.8 TPEG_OperatingTime

This class contains a start and a stop time.

3.7.9 Data types

The file ‘TPEG_DataTypes.h’ contains typedef’s for all the fundamental TPEG data types (e.g. intunti).

3.7.10 Conversion functions

The file ‘TPEG_Conversion.h’ contains inline functions to do the following:

- Bytes to various TPEG integer types (e.g. intunli)
- Length and speed conversions
- Numag (TPEG numerical format) conversion

The file ‘TPEG_CoordinateConversion.h’ contains inline functions to convert between longitude/latitude and OS National Grid References (NGR).

4 Examples of use

Two example applications have been written using the library and are provided with it as example code. These were not incorporated into the library itself because they include platform-dependent code (for gcc under Linux). Extracts from these applications are shown below to illustrate the user of the library.

4.1 RTM Generator and Multiplexer

This application (‘simpleSpool.cpp’) acts as both an application source (RTM generator) and simple frame multiplexer/playout module. The RTM generator part uses a TPEG_RTMXMLParser to parse an RTM XML file to generate RTM objects. The multiplexer just uses this single application source and some fixed SNI data to create the transport frames.

The main steps in this are:

Initialise frame objects and create fixed SNI data (extract from main()):

```
// Set Transport Frame parameters
serviceFrameNormal.setServiceID_A(serviceID_A);
serviceFrameNormal.setServiceID_B(serviceID_B);
serviceFrameNormal.setServiceID_C(serviceID_C);

// Set stream directory
sid.serviceID_A(serviceID_A);
sid.serviceID_B(serviceID_B);
sid.serviceID_C(serviceID_C);
serviceFrameStreamDirectory.addSID(&sid);

// Create SI in fast tuning table for 1 service component
TPEG_SNIFastTuningComponent fastTuningSNI; // FastTuning table
fastTuningSNI.setVersion(1);
TPEG_SNIFastTuningComponentEntry tableEntry; // Entry in table
tableEntry.setServiceComponentID(1);
tableEntry.setServiceID_A(serviceID_A);
tableEntry.setServiceID_B(serviceID_B);
tableEntry.setServiceID_C(serviceID_C);
tableEntry.setApplicationID(1);
fastTuningSNI.addEntry(&tableEntry);

// Add FastTuningComponent to SNI component frame
sniComponentFrame.getSNIComponentList->add(&fastTuningSNI);
```

Parse XML content file to create list of RTM objects (from parseContentFile()):

```
try {
   parser->parseFile(filename,&anRTMList);
}
catch (TPEG_Exception e) {
   // Catch any exceptions
   cout<<e.getMessage()<<”\n\n”;
}
```

Create RTM component frames and distribute RTMs amongst them. Then add all component frames to the transport frame and encode transport frame to create byte data (from createTransportFrame()):

```
// Calculate number of RTM component frames needed and
// create array of component frames: rtmComponentFrames[]

rtm = anRTMList.getFirstRTM(); // Point to first RTM
for (each RTM component frame) {
   for (each RTM in this component frame) {
          // Add (pointer to) RTM to component frame list
          rtmComponentFrames[ct].getRTMList()->addRTM(rtm);
```

```
            rtm = anRTMList().getNextRTM(); // Move to next RTM
        }
        // Add this component frame to service frame
        serviceFrameNormal.addComponentFrame(&rtmComponentFrames[ct]);
    }
    // Add SNI frame to transport frame
    serviceFrameNormal.addComponent(&sniComponentFrame);

    // Add service frame to transport frame
    transportFrame.setServiceFrame(&serviceFrameNormal);
    // Add stream directory to second transport frame
    transportFrame2.setServiceFrame(&serviceFrameStreamDirectory);

    // TransportFrames now complete so create byte data from transport frames
    encodedFrame = transportFrame2.encode();
    list<byte> l = transportFrame.encode();
    encodedFrame.splice(encodedFrame.end(),l);
```

4. Stream out encoded data bytes to output socket (platform dependent bit). Also check for whether XML input file has changed. If new content available then create component and transport frames again.

4.2 RTM Receiver

This application (‘simplerx.cpp’) is a simple RTM receiver. It connects to a TPEG data stream and decodes this. SNI information is displayed to allow the user to select a service component carrying an RTM application. This RTM application is decoded and a database of RTMs is kept.

Initialise frame objects:

```
// Set sID’s on service frames that should be decoded
serviceFrameNormal.setServiceID_A(serviceID_A);
serviceFrameNormal.setServiceID_B(serviceID_B);
serviceFrameNormal.setServiceID_C(serviceID_C);
transportFrame.setServiceFrame(&serviceFrameNormal);
// Service frame will decode this RTM component frame…
serviceFrameNormal.addComponentFrame(&rtmComponentFrame);
// Service frame will decode this SNI component frame…
serviceFrameNormal.addComponentFrame(&sniComponentFrame);
```

Platform dependent code to connect to TPEG data stream (in demuxDecode()).

State machine algorithm used to parse TPEG stream (from decodeFrame()):

Look for ‘FF0F’ sync bytes

Read field length, CRC, service frame type, service ID’s and encryption indicator. Continue if want to decode this transport frame.

Read in transport frame data bytes

Check that byte following transport frame is OK (should be 00 or FF). If OK then decode:

```
// Initialise temporary lists of RTMs and SNI components
// RTMs/SNIComponents must be removed before each transport frame is parsed
rtmList = rtmComponentFrame.getRTMList();
rtmList.removeAllRTMs();
sniList = sniComponentFrame.getSNIComponentList();
sniList.removeAllSNIComponents();

try {
       // Parse transport frame data - parses components frames
       // RTMComponentFrame parse() adds RTMs to rtmlist
       // SNIComponentFrame parse() adds SNI components to sniList
       transportFrame.parse(frameDataArray);
}
catch (TPEG_ParseException e) {
```

```
            cout<<e.getMessage()<<"\n";
        }
        // Now either call sniDecode() or rtmDecode()
```

sniDecode(): Decode and display SNI data, then allow user to select RTM service component to decode.

```
// Check for new version of SNI data
// (Actual code is more complex because you need to decide when you have
// received all the SNI component frames).
if (newVersion) {
    sniDatabase.clear(); // Clear database of SNI ready for new version
    // Add all SNI components from SNIComponentFrame to database
    sni = sniList->getFirstSNIComponent();
    for (ct=0; ct<sniList->getNumSNIComponents(); ct++) {
            // Set SI data from this SNIComponent
            sniDatabase.set(sni); // Adds/updates objects in database
            sni = sniList->getNextSNIComponent();
    }
    // Assume you have got all SNI data so display it
    cout<<"SNI Database (version "<<(int)sniDatabase.getVersion()<<"):\n";
    cout<<sniDatabase.getServiceName()<<":";
    cout<<sniDatabase.getServiceDescription()<<"\n";
    // Go through all service components with SI in database
    for (int ct=0; ct<sniDatabase.getNumComponents(); ct++) {
            // Print out SI for this service component
            cout<<(ct+1)<<") "<<sniDatabase.get(ct)->toString()<<"\n";
    }
    // Get user to choose which service component to decode
    // and set appropriate service component ID in rtmComponentFrame…
}
```

rtmDecode(): Add messages to RTM database.

```
// Pass list of RTMs from rtmComponentFrame to database
// This adds/updates/deletes RTMs from database and
// logs any changes to database to the console
rtmDatabase.set(*rtmList,true);
// Note: Should occasionally call rtmDatabase.checkExpire(true);
```

6. Repeat from step 3

5 References

1. EBU - B/TPEG, ‘TPEG Specifications - Part 2: Syntax, Semantics and Framing Structure, TPEG-SSF_3.0/001’. November 2001.
2. EBU - B/TPEG ‘TPEG Specification - Part 3: Service and Network Information Application, TPEG - SNI_3.0/001’. November 2001.
3. EBU - B/TPEG ‘TPEG Specification - Part 4: Road Traffic Message Application, TPEG - RTM_3.0/003. July 2002.
4. EBU - B/TPEG ‘TPEG Specification - Part 6: Location Referencing for Applications, TPEG - LOC_1.0/003. July 2002.
5. EBU – B/TPEG ‘tpegML specifications – Part 1: Introduction, common data types and tpegML’. August 2002.
6. EBU – B/TPEG ‘tpegML specifications – Part 2: tpeg-locML’. August 2002.
7. EBU – B/TPEG ‘tpegML specifications – Part 3: tpeg-rtmML’. August 2002.
8. ‘libxml – The XML C Library for Gnome’. http://xmlsoft.org/

6 Contacts

TPEG specifications can be found on the B/TPEG web-site (http://www.ebu.ch/bmc_btpeg.htm).

For matters relating to the BBC TPEG C++ Library please see the BBC R&D TPEG web-site (http://www.bbc.co.uk/rd/projects/tpeg).

7 Appendix 1 - Known problems and limitations

7.1 Memory leaks

There have been problems in the string handling used in the library.

7.2 Unsupported parts of the specification

7.2.1 SNI

Service component reset

Service logo

Same service linkage information

Related service linkage information

Subscriber information

7.2.2 RTM

Diversion advice

7.2.3 LOC

Location descriptions container

7.2.4 PTI

All of the Public Transport Information (PTI) specification.

8 Appendix 2 – Installation

8.1 Archive structure and contents

Unzipping the downloaded archive should produce the following directory structure:

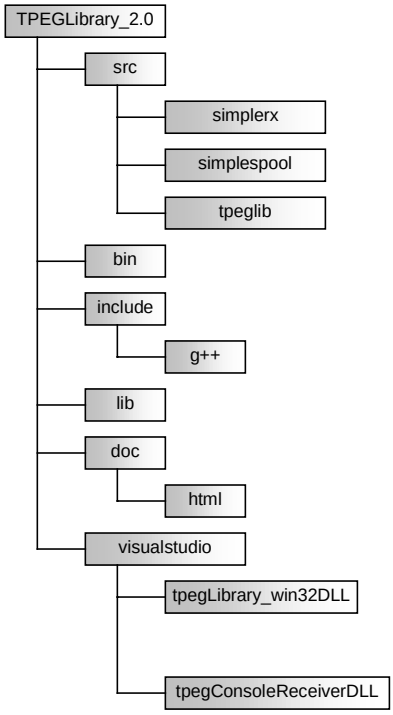

Figure 8 – TPEG Library Archive

Where:

simplerx: Source for a simple console-based TPEG receiver for Linux.

simplespool: Source for simple TPEG stream generator for Linux.

tpeglib: TPEG C++ library source sode.

html: Automatically generated HTML documentation for the TPEG C++ library.

tpegLibrary_win32DLL: Microsoft Visual Studio 6 files for building the TPEG C++ library as a DLL.

tpegConsoleReceiverDLL: Source for a simple console-based TPEG receiver for Win32 using the DLL.

8.2 Linux installation

The library was developed under gcc 2.8.1 and Linux 2.0.30

A makefile is included to build the library. `simplespool` is a simple TPEG stream generator that parses a given XML file and produces a TPEG stream on a given socket. `simplerx` is a simple TPEG receiver that takes a TPEG stream as input from a socket, file or stdin, keeps a database of messages and displays new messages to the console.

8.3 Windows installation

The library has been ported to Win32 using Windows NT 4.0 and Microsoft Visual C++ 6.0.

Project workspaces for Microsoft Visual Studio 6.0 are provided and they allow the building of a DLL and a simple console-based receiver. `tpegConsoleReceiver` is the Win32 equivalent of `simplerx` described above.

When using the TPEG Library DLL it should be copied to a directory where Windows can find it, i.e. one of; Current directory, executable's directory, System directory, Windows directory, other directory in path.

8.3.1 libxml

This has not yet been tested with the Windows port. For Linux it should be installed as described in [8].

8.3.2 Bison and Lex

I used precompiled binaries of GNU Bison and Flex for Win32 downloaded from:

http://agnes.dida.physik.uni-essen.de/~janjaap/mingw32/index.html

The settings needed for using Bison and Flex in Visual Studio 6:

1. Choose menu:

tools > options > directories > executable : Add bison and flex directories.

> options > directories >include : Add flex directory.

2. Copy bison.simple to project directory

3. Create empty unistd.h file in flex directory

Also see 'bison.txt' in the tpegLibrary_win32 directory.