DVB 标准深入分析和实际应用

版 本 历 史

版本/状态	作者	起止日期	备注
1.0/初稿	许怡洋	2006-10-08	创建文档

前　言

一直想对自己五年的数字电视机顶盒开发做个总结。一方面回顾一下几年的历程，把自己所了解的 DVB 机顶盒相关的知识归纳一下，对自己是个交代，另一方面希望对新入行的同行兄弟们有些帮助，毕竟开始这个行业的标准众多不是很好入手。

想写的很多，开始架子也搭的过大。结果过了这么久才写了一个章节，其他的不知道什么时候能写完了。马上年底了，干脆先先发出来吧，别又是虎头蛇尾的折腾一气。

最后希望行内的高手有暇看过后，帮忙做下勘误。免得错误太多误导大家，谢谢！

我的信箱地址是rainbowbox@163.com，也可以在我的论坛技术板块里头留言，再次感谢！

涉及的标准概述

机顶盒涉及的标准主要来自于 ETSI DVB 组织标准和国家标准。ETSI 的 DVB 标准在条件接收、传输、多路复用、MHP 等方面有很多的标准规范。对于机顶盒软件设计开发我们要熟悉以下的标准：

Standard Ref: EN 300 468
Edition: 1.4.1
Specification for Service Information (SI) in DVB systems

Standard Ref: TR 101 211
Edition: 1.5.1
Guidelines on implementation and usage of Service Information (SI)

GB/T 17975.1-2000
Idt ISO/IEC 13818-1:1996
运动图像及其伴音信息的通用编码 第 1 部分：系统

GB/T 17975.2-2000
信息技术 运动图像及其伴音信号的通用编码 第 2 部分 视频
适用范围：

本标准规定了数字存贮媒体和数字视频通信用的图像信息的编码表示，并且规定了解码过程。这种表示支持恒定码率传输、可变码率传输、随机存取、信道跳换、可分级解码、比特流编辑、以及诸如快速正放、快速倒放、慢放、暂停和静像等特殊功能。本标准与 GB/T 17191.2-1997 前向兼容，并与 EDTV、HDTV、SDTV 格式上兼容或下兼容。本标准主要应用于数字存贮媒体、视频广播与通信。存贮媒体可以与解码器直接联接，或是通过诸如总线、LAN 或电信链路等通信设施与解码器联接。

GB/T 17975.3-2002
信息技术 运动图像及其伴音信号的通用编码 第 3 部分：音频
适用范围：

本标准在 GB/T 17191.3 基础上规定了以下扩展：1、更低取样频率；2、多声道和多语种的高质量音频信号的编码表述和解码方法。编码器的输入和解码器的输出是和 PCM 标准兼容的。本标准适用于广播、传输和存储媒体。

GY/Z 175-2001
数字电视广播条件接收系统规范

如果要做机顶盒中间件，需要再了解下面三个标准：
Standard Ref: TS 101 812
Edition: 1.2.1

Digital Video Broadcasting (DVB) Multimedia Home Platform (MHP)

Standard Ref: TS 102 812
Edition: 1.1.1
Digital Video Broadcasting (DVB) Multimedia Home Platform

Standard Ref: TS 102 819
Edition: 1.1.1
"Digital Video Broadcasting (DVB); Globally Executable MHP (GEM)"

这些标准奠定了机顶盒的基础，只有掌握并熟悉运用它们才能做出高质量的机顶盒软件。

这个章节里我着重介绍三个标准的重点和在实际中的运用,这是机顶盒软件中最重要也是最常用的部分：

- 运动图像及其伴音信息的通用编码 第 1 部分：系统
  这部分重要的部分是 PSI 表的组织结构和运用，包括 PAT(节目关联表)、PMT（节目映射表)、NIT（网络信息表)、CAT（条件存取表）

- Specification for Service Information (SI) in DVB systems
  重点是 SI 子表的组织结构和子表之间互相的逻辑关系，这是对 MPEG2 系统的 PSI 子表的扩展和补充

- Guidelines on implementation and usage of Service Information (SI)
  这个 SI 的实施指导非常重要，建议全文通读认真理解

PSI/SI 子表详解及实际运用

在 Specification for Service Information (SI) in DVB systems 标准中有业务信息的 PID 分配和 Table_id 分配的两个表格。在实际运用中中有些子表是很少用到的，我把常用的罗列出来，合并 PID 和 Table_id 为一个表来讲解，这样比较容易理解。

业务信息的 PID 和 Table_id 分配

表	PID 值	Table_id 值
PAT(节目关联表)	0x0000	0x00
CAT(条件接收表)	0x0001	0x01
PMT(节目映射表)		0x02
NIT(网络信息表)	0x0010	现行网络信息段 0x40
		其它网络信息段 0x41
SDT(业务描述表)	0x0011	现行传输流业务描述段 0x42
		其它传输流业务描述段 0x46
BAT(业务关联表)	0x0011	0x4A
EIT(事件信息表)	0x0012	现行传输流事件信息段，当前/后续 0x4E
		其它传输流事件信息段，当前/后续 0X4F
		现行传输流事件信息段，时间表 0x50-0x5F
		其它传输流事件信息段，时间表 0x60-0x6F
TDT(时间－日期表)	0x0014	0x70
TOT(时间偏移表)	0x0014	0x72
预留使用	0x0016 至 0x001B 0x0003 至 0x000F	0x04 至 0x3F 0x43 至 0x45 0x4B 至 0x4D 0x74 至 0x7D 0xFF
用户定义		0x80 至 0xFE

表一：

这张表格涵盖了在应用中常用到的 PSI/SI 子表，如果做机顶盒的开发就要把这些每个子表的 PID、Table_id、组成结构、描述符、用途、关键字段含义烂熟与心。这是构成机顶盒应用软件的基础。后面我会剖析子表中的一些关键描述符、关键字段。

我引用下面这张图并修改了一下用来讲解在 DVB 传输系统中，网络、传输流、业务和事件的关系。子表的用途我标注到了相应的层次中便于理解。如果这个逻辑关系搞的很清楚，那么祝贺你，你已经理解了最容易概念混淆的部分。请直接跳过本节往下看。

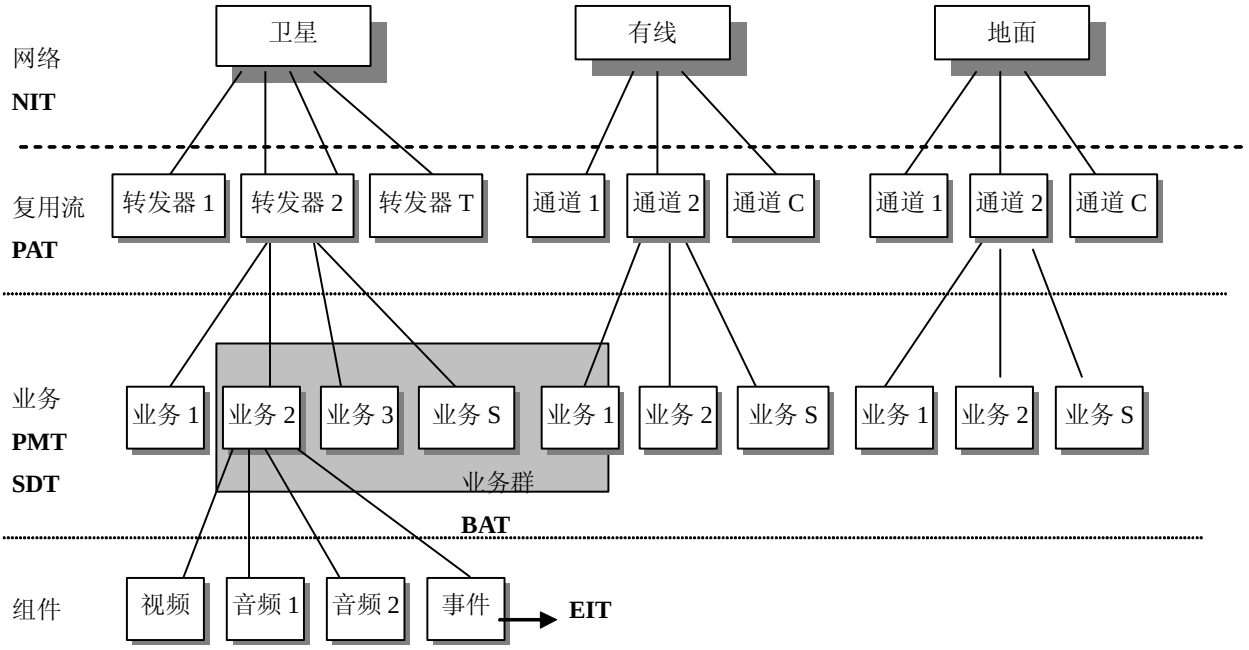


图 1　数字广播、业务传送模式

我把子表的作用范围对照这个树形结构标了出来，通过揣摩这张图我们能获得 NIT、PAT、PMT、SDT、BAT、EIT 这几个子表之间的关系，每个子表的语法大家可以参考标准，下面我把标准中的内容提炼出来结合实际应用分成几个部分供大家深入理解。

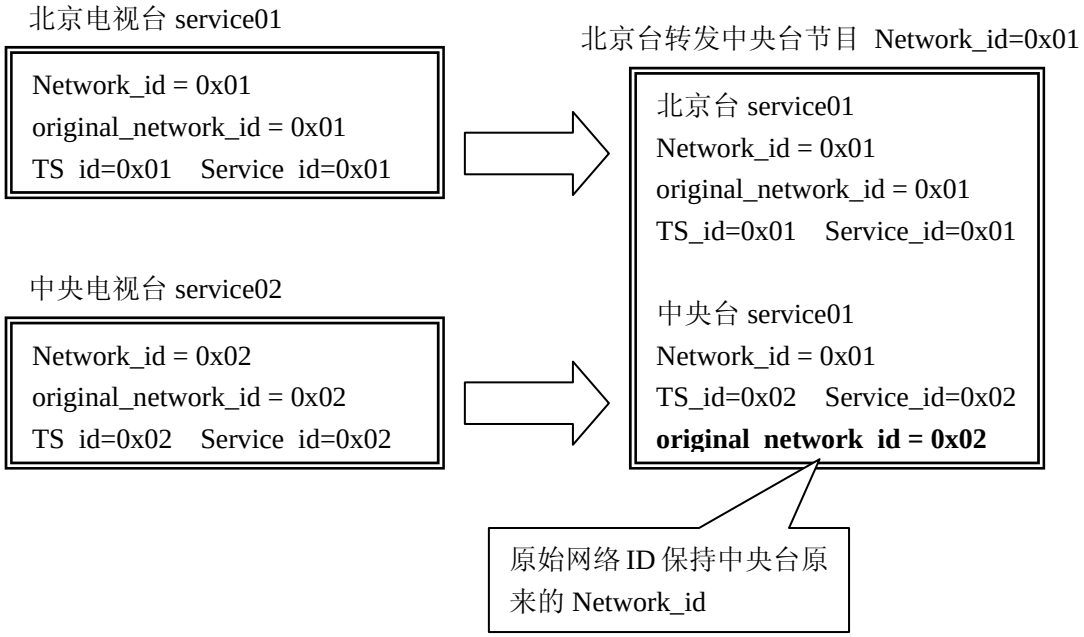
- **Network、TS、Service、Event 的概念以及之间的关系**

从上面图一能看出来 Network 、TS、 Service、 Event 是一个由上至下的层次逻辑结构。

Network 直译叫做网络，用来区别不同的传送系统和与运营商，例如卫星、地面、有线（网络标示符可以参见 SI 标准的附录 D）。实际上可以也看作是一个网络的运营商，例如中央电视台。每个运营商都需要注册唯一的 network_id, 假设中央台的 network_id = 0x01, 北京电视台的 network_id=0x02。

这里顺便解释一下 original_network_id 和 network_id 的用途。original_network_id 表示的是最初的网络 ID,这个 ID 是不随着 Network_id 变化而变化的。当本台的节目进行播发时，network_id = = original_network_id,但是当本台的节目被其他台播发时，network_id != original_network_id,

例如：由北京台播发的节目包括本台的 service01，还转发了中央台的节目 service02,那么在北京台发送的两个 service 中 network_id 都是 0x02,但是中央台 service02 的 original_network_id 依然是 0x02。也就是说 original_network_id 始终会保持节目所在的原来的网络 ID 不变。示意图如下：



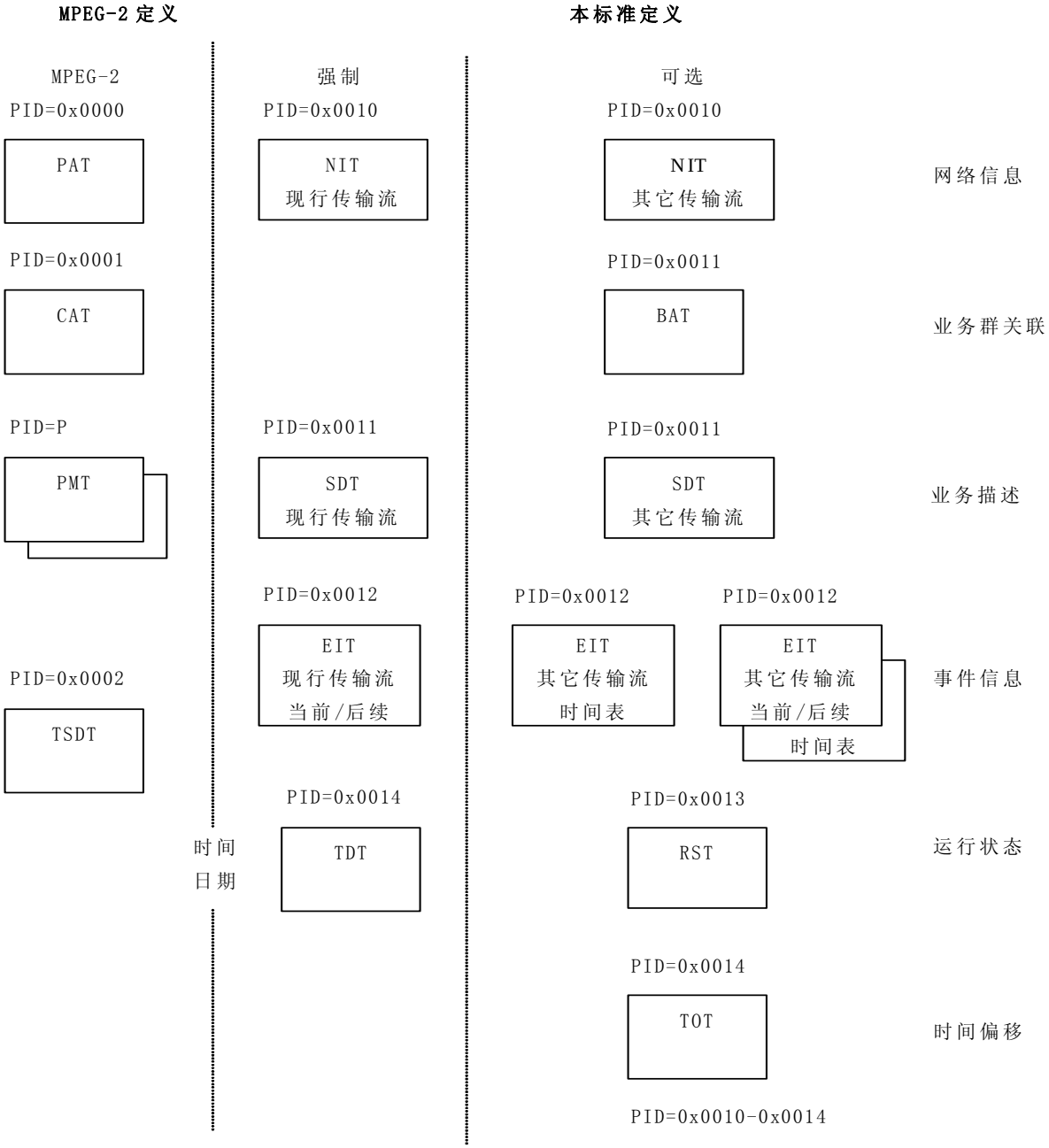
TS（传输流）就是一个物理传送通道，只是在卫星、有线、地面这三种传输方式中描述的物理参数不尽相同。例如在数字电视有线网络（DVB-C）中用频率、符号率、调制方式来描述这个物理通道。

Service（业务）就是在通道上传输的节目。例如中央一台。通过复用，可以在一个物理通道上传输多个节目（6－8 套）。我认为 GB/T 17975.1-2000 国家标准称之为业务总感觉不好理解，所以称之为节目。

Event 是描述的是节目中某个时间段的事件。例如半个小时的新闻联播就是一个 Event。

- **PSI 与 SI 的关联**

总的来说 SI 是对 PSI 的扩展。PSI 范围中子表的主要作用是索引定位数字音视频业务，加以 CA 对节目进行加密，没有那么多复杂的信息需要通过多种多样的子表和描述符传递。SI 丰富了数字网络中的内容，通过传送的信息可以组织成 EPG(电子节目指南)、NVOD(准视频点播)、马赛克电视等多种应用。



注：在“强制”规定的 NIT 表中，“现行传输流”应理解为“现行传送系统”。
在“可选”规定的 NIT 表中，“其他传输流”应理解为“其他传送系统”。

图 2 业务信息（SI）总体结构

通过上表我们可以看出 PSI 和 SI 定义的范围。需要记住以下几点：

1. SI 中定义了一些强制发送的子表，在网络中必须发送这几个子表。可选是根据运营商开展的业务有选择的发送。其中 NIT 必须全网发送，即在网络中任何一个 TS 中都要插入 NIT。

2. SDT、EIT 分为现行传输流和其他传输流。在当前传输流和其他传输流中这两个子表分别对应的 PID 相同，但 Table_id 是不一样的。参见前面的业务信息的 PID 和 Table_id 分配。

3. 在考虑随机存取的系统中每个子表根据前端情况有自己的重复率，按照标准规定在传输码率为 100 兆比特/秒的系统中，对于标有同一个 PID、table_id 及 table_id_extension 值的业务信息段，其段的最后一个字节与下一个段的首字节发送的最小时间间隔为 25 毫秒。

4. 除 EIT 表外，每个表中的段限长为 1024 字节，但 EIT 中的段限长 4096 字节

- **EIT present/following 和 EIT Schedule 释疑**

EIT 是 SI 中的关键子表，也是比较复杂的一个子表，这里着重介绍 EIT 的相关知识。

1. 什么是 event(事件)？通俗的说事件就是一个节目中的一个时间片断。它有起始时间、持续时间、事件名称和事件的简单描述。也可以说一个 service(节目)就是多个连续 event 的集合。例如：19：00---19：30 的新闻联播就是一个 event，它的起始时间是 19：00；持续时间是 30 分钟；节目名称是新闻联播。由此看出 event 可以任意定义，只要具备了它的几个属性就是一个 event。

2. EIT present/following 表示的是两个事件。EIT present 是正在播出的事件，EIT following 是即将播出的事件。在 EIT 语法字段结构中用 section_number 区分 present/following。
当 section_number=0，表示是当前节目信息，section_number=1,表示是后续节目信息。
当 table_id=0x4E，表示这是一个现行传送流的当前/后续事件信息
当 table_id=0x4F，表示这是一个其他传送流的当前/后续事件信息

   举个例子：
   假如现在时间是 19：20，那么我们会收到下面这样的当前/后续事件：
   19：00----19：30　新闻联播　event_id=0x01　　（当前事件）
   19：31----20：00　动画片　　event_id=0x02　　（后续事件）

   如果时间到了 19：35，那么我们将会收到下面这样更新的当前/后续事件：
   19：31----20：00　动画片　　event_id=0x02　　（当前事件）
   20：01----22：00　黑客帝国　event_id=0x03　　（后续事件）

3. EIT Schedule 用来发送大量的 event 信息，也就是 EPG 的节目单。例如中央 1 台一周的电视节目预告就需要用到 EIT Schedule 传送。

   EITSchedule 被分成 16 个 table_id 传送。当 table_id=0x50－0x5F,表示是现行传送流的表；当 table_id=0x60－0x6F,表示是其他传送流的节目表。也就是说一个节目最多可以用 16 个子表用来发送节目预告。这 16 个 table_id 的 EIT 是按照时间先后顺序排列的。

这里要注意的是 EIT 子表中引入节（segment）的概念。这是个很让人迷惑的东西，我在这里解释一下。

在普通子表中分段使用段（section），每个子表语法结构中都有个 8bit 字段 last_section_number，用来表示一个子表最多由多少个段组成。由字长我们可以看出，一个子表最多有 256 个段。而 EIT 中在子表和段中插入了一个节的层次。关于节有如下的约定：

- 一个 EIT 子表被分成 32 个节；
- 每个节最多有 8 个段；
- 一个 EIT 子表最多有 256 个段；
- 每个节所包含的事件信息最长不能超过 3 个小时；
- 如果在节中小于 8 个段在使用， 就需要靠字段 segment_last_section_number 标识节中有哪些有效的段。EIT 中传送的字段 segment_last_section_number 值算法为 s0 + n – 1。这里 s0 的值是节中的第一个段号，n 表示该节中段的个数。
  举个例子：如果某个 EIT 子表中第二个节只包含 2 个段，那么在这 2 个段中字段 segment_last_section_number 的值就是：8 + 2 - 1 = 9。大家想一想为什么这里的 s0＝8？因为每个节最多包含 8 个段，第一个节的段号是 0－7，那么第二个节段号就是 8－15，由此得出第二个节的 s0 为 8。这一段约定有些难以理解，大家可以分析一些 EIT sechudle 的码流再结合上述就能明白了；

- 节中包含所有的段，也就是有 8 个段。那么字段 segment_last_section_number 值算法为 s0 +7；

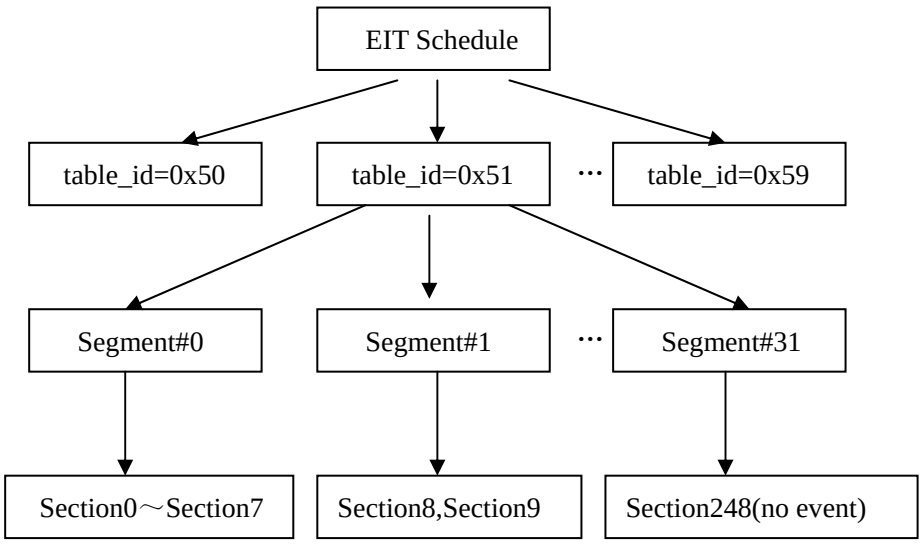
- 如果节中包含所有的段是空段（段中没有任何事件信息），那么字段 segment_last_section_number 值算法为 s0 +0；

- EIT Table_id=0x50(或者0x60)的第一个节包含着今天00:00－02:59:59 UTC时间的三个小时的事件信息，第二个节包含了03:00:00 and 05:59:59 UTC时间的三个小时的事件信息，依次类推下去，一个子表最多能包含4天的事件信息；

  这样我们可以计算出一个 EIT Schedule 最多能存放多长时间的节目简介：
  3h(单个节最长时间)*32（最大的节个数）*16(最大的子表个数)＝1536h=64day

  一个 service 的 EIT Schedule 子表最多可以传送 64 天的 EPG，一个子表最多可以传送 4 天的 EPG。在实际运营中一般会只传一周的节目指南，也就是说只要两个子表就足够了。所以我们在运用中用到的 EIT Schedule 的 table_id 范围是：？？？
  这个问题就留给大家当个简单的作业吧：）

最后我用一张示意图表示 EIT 中子表、节(segment)、段(section)之间的关系，对上述概念做一个回顾并出几道题给大家便于加深理解。EIT 子表是 EPG 的关键，这部分一定要熟练掌握。



从这张图上我们能得到什么信息？我出几个问题，大家最好先自己独立思考一下，答案在后面公布。

Q1：这个 EIT Schedule 最多能传送几天的节目预告？

Q2：Segment#0、Segment#1、Segment#31 这 3 个节中的 section 字段里 segment_last_section_number 分别是多少？

Q3：假设第一个子表 0x50 起始 event 时间是从 2006 年 1 月 1 日 00:00 开始的，那么图中 Section0 的起始时间是什么时间？

我认为 EIT 中让人迷惑的难点疑点基本就是上述内容了，需要再详细的细节可以查看标准。如果你能把这些都理解掌握，那么 EIT 的运用也就不成问题了。

【答案】

A1：因为 table_id 从 0x50~0x59,一共 10 个子表，一张子表最多包含 4 天事件信息，所以这个 EIT Schedule 最多能传送 40 天的节目信息。

A2：Segment#0 中有 8 个 section,所以 segment_last_section_number＝0＋8＝8

Segment#1 中只有 2 个 section,所以 segment_last_section_number＝8＋2－1＝9

Segment#31 中有 1 个 section 但没有包含 event 内容,所以 segment_last_section_number＝248＋0＝248

A3：因为 0x50 最多包含 4 天信息，所以 0x51 的 Section0 起始时间为 2006 年 1 月 5 日 00:00~02:59:59

- **重要的几个子表用途和关联**

PSI/SI 定义的子表很多，必须要熟知重要的子表。这一节的目的就是明晰用途、理清互相的关系。

NIT 用途：用来描述网络信息、网络传输的复用流/TS 流的物理结构信息

PAT 用途：描述当前传输流中节目的 PID 信息

PMT 用途：描述节目的音视频 PID、PCR_PID、音视频 ECM_PID 信息

SDT 用途：描述节目名称、节目提供者名称等补充信息

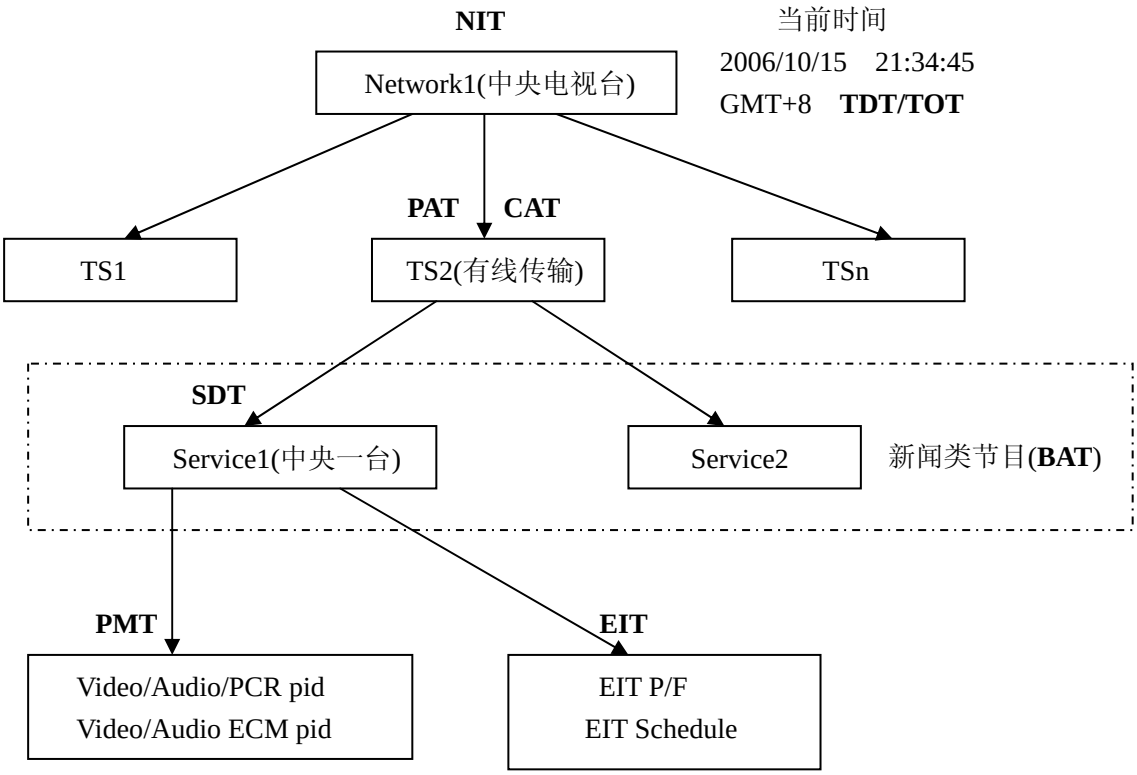
EIT 用途：描述事件信息，包括事件名称、事件起始和持续时间、事件内容简介等信息

BAT 用途：描述业务群的名称、业务群 ID,以及业务群包含的节目信息。

CAT 用途：描述了 CA 相关信息

TDT/TOD 用途：描述当前的格林威治时间信息和本地偏移时间信息

把这些子表结合起来我们就可以得出下面这张示意图，它表示了每个子表在网络中的用途和描述的信息：



理解了这张示意图也就理解了 PSI/SI 主要子表之间的逻辑关系，进而就掌握了 SI 标准的核心内容。

- **描述符用途概要分析**

描述符是存放在子表中用来描述该表中特殊意义的字段，描述符插入到段中打包传送。每个子表都有自己的用途，因此插入到各个子表中的描述符也就多种多样了。

还是引用SI标准中的描述符表，我把常用的筛选了出来，为了便于理解，我基于一些功能用途对描述符进行了归类，以便我下面对分类的描述符统一讲解，对一些无法分类的重要的描述符做详细剖析。这张表描述了描述符出现在子表中的位置。也就是说子表会根据需求插入可能出现位置的描述符。

DVB 标准中描述符的数量可谓是铺天盖地来形容，刚开始根本无法记住这么多描述符的用途和在哪个子表中会被调用。等做了一段应用开发后才能理清头绪。我要做的就是把我的经验共享出来，让开始迷惑头痛的时间缩短些。我定义的是基本分类能帮助你快速进入开发状态，有些并不常用的描述符被我过滤掉了，那些等你在实际用到时再查标准就容易多了。

描述符的可能位置

描述符	标签值	NIT	BAT	SDT	EIT	TOT	PMT
描述名称的							
network_name_descriptor	0x40	*	-	-	-	-	-
bouquet_name_descriptor	0x47	-	*	*	-	-	-
multilingual_network_name_descriptor	0x5B	*	-	-	-	-	-
multilingual_bouquet_name_descriptor	0x5C	-	*	-	-	-	-
multilingual_service_name_descriptor	0x5D	-	-	*	-	-	-
service_descriptor	0x48	-	-	*	-	-	-
描述传送方式的							
satellite_delivery_system_descriptor	0x43	*	-	-	-	-	-
cable_delivery_system_descriptor	0x44	*	-	-	-	-	-
terrestrial_delivery_system_descriptor	0x5A	*	-	-	-	-	-
Frequency_list_descriptor	0x62	*	-	-	-	-	-
描述VBI图文电视及字幕							
VBI_teletext_descriptor	0x45	-	-	-	-	-	*
VBI_teletext_descriptor	0x46	-	-	-	-	-	*
teletext_descriptor	0x56	-	-	-	-	-	*
subtitling_descriptor	0x59	-	-	-	-	-	*
描述NVOD功能的							
NVOD_reference_descriptor	0x4B	-	-	*	-	-	-
time_shifted_service_descriptor	0x4C	-	-	*	-	-	-
time_shifted_event_descriptor	0x4F	-	-	-	*	-	-
描述EPG内容信息及内容分类							
short_event_descriptor	0x4D	-	-	-	*	-	-
extended_event_descriptor	0x4E	-	-	-	*	-	-
content_descriptor	0x54	-	-	-	*	-	-
描述马赛克业务							
mosaic_descriptor	0x51	-	-	*	-	-	*
描述CA业务相关信息的							
CA_identifier_descriptor	0x53	-	*	*	*	-	-
parental_rating_descriptor	0x55	-	-	-	*	-	-
描述本地偏移时钟							
local_time_offset_descriptor	0x58	-	-	-	-	*	-
私有数据描述符							
private_data_specifier_descriptor	0x5F	*	*	*	*	-	*
无法按照功能归类的							
service_list_descriptor	0x41	*	*	-	-	-	-
linkage_descriptor	0x4A	*	*	*	*	-	-
multilingual_component_descriptor	0x5E	-	-	-	*	-	-

component_descriptor	0x50	-	-	-	*	-	-
预留使用	0x6F 至0x7F						
用户定义	0x80 至0xFE						
禁止	0xFF						

下面对归类的描述符做一个简介：

1. 描述名称的
   从描述符的字面上也能看出，这类描述符用来描述所属子表需要传送的名称。包括 NIT 中的网络名称、BAT 中的业务群名称、SDT 的节目名称和节目提供者名称。前缀有 multilingual_的用来添加多语种字符。

2. 描述传送方式的
   用来描述 DVB 传送方式的，包括 DVB-C(有线传输)、DVB-T(地面传输)、DVB-S(卫星传输)三种传输方式的物理参数描述

3. 描述 VBI 图文电视及字幕
   用来描述 teletext 图文电视信息和 subtitle 字幕信息的描述符。

4. 描述 NVOD 功能的
   这组描述符分别出现在 SDT 和 EIT 中，用来实现 NVOD 功能，在后面软件模块设计 NVOD 部分会详细讲解。

5. 描述 EPG 内容信息及内容分类
   这组描述符用来描述 EIT 的信息。包括事件名称、事件简介，事件内容分类等。在后面软件模块设计 EPG 部分会详细讲解。

6. 描述马赛克业务
   用来马赛克业务的实现，这个描述符要配合 linkage_descriptor 才能实现马赛克业务的定位。在后面软件模块设计马赛克部分会详细讲解。

7. 描述 CA 业务相关信息的-
   这组描述符用来引导机顶盒如何寻找 CA 中所需的 ECM_PID、EMM_PID；事件父母分级的实现等 CA 方面的功能

8. linkage_descriptor
   这是个非常灵活的描述符，从表中也可以看出它可以出现在很多子表中。常用来作链接定位作用。比如在马赛克中、在 loader 中，这个描述符会在后面的功能模块中详细讲解。

描述符概要就讲到这里，大家可以思考一下，为什么 DVB 标准要用这么繁多的描述符插入的方式去发送子表？开始我只是闷头去做，标准咋定义咱就咋实现呗，慢慢的我理解了。数字广播系统是个非常庞大的系统，在前面我已经讲过，为了最大限度的利用网络中有限的带宽资源，DVB 采用了 MPEG2 的编码压缩技术用于传送音视频数据。同样的，子表在传送中也需要带宽开支，为了最大限度减少带宽，用描述符的方式就可以根据需要插入到子表中传输数据。比如我发送 SDT 子表时，网络中的用户都是中国人，那我就可以不插入多语种描述符，这样节约了发送开销；而当我的网络用户中有了西班牙人、法国人、冰岛人时，我再插入多语种描述符应对不同的需求，这样就能灵活的调度有限的资源做更多的事。描述符数目繁多但能根据需要动态插入，原因就在于此。

- **子表中关键字段详解**

下面我用一个 EIT 信息段讲解段中的关键字段，这些关键字段具有通用性，在其他的子表中也具有相同的用途：

语　　法	位数
event_infotmation_section(){	
table_id	8
section_syntax_indicator	1
reserved_future_use	1
reserved	2
section_length	12
service_id	16
reserved	2
version_number	5
current_next_indicator	1
section_number	8
last_section_number	8
transport_stream_id	16
original_network_id	16
segment_last_section_number	8
last_table_id	8
for(i=0;i<N;i++){	
event_id	16
start_time	40
duration	24
running_status	3
free_CA_mode	1

```
            descriptors_loop_length                          12
            for(j=0;j<N;j++){
                  descriptor()
            }
      }
      CRC_32                                                 32
}
```

Table_id：子表的标识符，用来标明当前的段属于哪个子表

section_length：段的长度，这是一个 12bit 的字长，表示从该字段的下一个字节开始的本段的字节长度，并包含 CRC。这样整个段的最大长度为 4096 字节。 减去占用的 3 个字节，EIT 一个段最大长度为 4093 字节。这里要注意的是 EIT 和其他子表的区别，在其他子表这个 12bit 字段前两位被置为“00”，这样整个段的最大长度为 1024 字节，同样减去占用的 3 个字节，其他子表一个段的最大长度为 1021 字节。这就是 EIT 和其他子表在段长度的区别。

section_number：8bit 字段，给出了段号。子表中的第一个段的 section_number 标为“0x00”。子表可能被分成很多部分。在每个部分中，每增加一个段，section_number 就加 1。由字长可以看出 section_number 范围是 0～255。

last_section_number：8bit 字段，表示所属的子表的最后一个段（即段号最大的段）的段号。从这个字段可以算出子表中一共有多少个段组成。

version_number：5bit 字段。标识子表的版本号。当子表包含的信息发生变化时，version_number 加 1。当值增至 31 时，复位为 0。当 current_next_indicator 置“1”时，则 version_number 为当前使用的子表的版本号。当 current_next_indicator 置“0”时，则 version_number 为下一个使用的子表的版本号。在实际应用中 current_next_indicator 很少会用到，所以一般都是版本号依次递增，范围从 0～31。这个字段很重要，是应用中监控功能的实现基础，后面讲到应用监控模块时会详细讲解。

descriptors_loop_length：12bit 字段，指出从本字段的下一个字节开始的描述符的总字节长度。这个长度用来标识下面的整个描述符循环的长度，实际应用中插入的描述符不止一个，每个描述符内部都带有一个 8bit 的 descriptor_length 用来标识本描述符的长度，这样就需要把每个的描述符长度累加起来，从而算出总共有多少个描述符，也就是 N 的个数。

descriptor_tag：8bit 字段，用于标识不同的描述符。表 12 定义了 decsriptor_tag 的值。

descriptor_length：8bit 字段，给出描述符的总长度。表示描述符中，从该字段后开始的数据部分的字节数。由此看出一个描述符最大长度是 256 字节

描述符虽然多种多样，但每个描述符语法格式都类似下面这样，第一个 byte 用来标识描述符类别，第二个 byte 标识描述符的长度，后面则是各自的具体数据：

```
private_data_specifier_descriptor(){
        descriptor_tag                                  8
        descriptor_length                               8
        private_data_specifier                         32
}
```