ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р 55715— 2013

ТЕЛЕВИДЕНИЕ ВЕЩАТЕЛЬНОЕ ЦИФРОВОЕ

ОБОРУДОВАНИЕ ЦИФРОВОЙ ВСТАВКИ (СПЛАЙСИНГА) РЕГИОНАЛЬНЫХ ПРОГРАММ В ТРАНСПОРТНЫЙ ПОТОК MPEG-2 ВЕЩАТЕЛЬНОГО ТЕЛЕВИДЕНИЯ

Основные параметры

Издание официальное

Москва
Стандартинформ
2014

Сведения о стандарте

1 РАЗРАБОТАН Автономной некоммерческой организацией «Научно-технический центр информатики» (АНО «НТЦИ»)

2 ВНЕСЕН Управлением технического регулирования и стандартизации Федерального агентства по техническому регулированию и метрологии

УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 8 ноября 2013 г. № 1369-ст

3 Настоящий стандарт разработан с учетом основных нормативных положений Американского национального стандарта / Общества инженеров кабельных телекоммуникаций, подкомитет Цифрового Видео «Прикладной программный интерфейс сплайсинга вставки цифровых программ» (ANSI/SCTE 30 2009 Digital Program Insertion Splicing Application Program Interface)

4 ВВЕДЕН ВПЕРВЫЕ

Правила применения настоящего стандарта установлены в ГОСТ Р 1.0-2012 (раздел 8). Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок – в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования – на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (gost.ru).

Содержание

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ТЕЛЕВИДЕНИЕ ВЕЩАТЕЛЬНОЕ ЦИФРОВОЕ ОБОРУДОВАНИЕ ЦИФРОВОЙ ВСТАВКИ (СПЛАЙСИНГА) РЕГИОНАЛЬНЫХ ПРОГРАММ В ТРАНСПОРТНЫЙ ПОТОК MPEG-2 ВЕЩАТЕЛЬНОГО ТЕЛЕВИДЕНИЯ

Основные параметры

Digital Video Broadcasting (DVB).

Splicing equipment of regional programs in the transport stream of MPEG-2 broadcast television.

Basic parameters

Дата введения—2014—09—01

1 Область применения

Настоящий стандарт распространяется на оборудование цифровой вставки (сплайсинга) программ в транспортные потоки MPEG-2 вещательного телевидения.

Настоящий стандарт устанавливает основные параметры оборудования цифровой вставки региональных программ в транспортные потоки MPEG-2, размещаемого на головных станциях (центрах) формирования программ вещания. В состав оборудования цифровой вставки входят серверы и сплайсеры, которые используют для обмена данными прикладной программный интерфейс (Application Program Interface; API).

Прикладной программный интерфейс создает стандартизированный метод передачи между серверами и сплайсерами, обеспечивающий вставки контента в выходные данные мультиплекса MPEG-2 на выходе сплайсера. Этот API достаточно гибок, чтобы поддерживать один или более серверов, присоединенных к одному или более сплайсерам.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты:

ГОСТ Р 52210–2004 Телевидение вещательное цифровое. Термины и определения

ГОСТ Р 52591–2006 Система передачи данных пользователя в цифровом телевизионном формате. Основные положения

Примечание – При пользовании настоящим стандартом целесообразно проверить действие ссылочных стандартов в информационной системе общего пользования – на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет или по ежегодно издаваемому информационному указателю «Национальные стандарты», который опубликован по состоянию на 1 января текущего года, и по соответствующим ежемесячно издаваемым информационным указателям, опубликованным в текущем году. Если ссылочный стандарт заменен (изменен), то при пользовании настоящим стандартом следует руководствоваться заменяющим (измененным) стандартом. Если ссылочный стандарт отменен без замены, то положение, в котором дана ссылка на него, применяется в части, не затрагивающей эту ссылку.

Издание официальное

3 Термины, определения и сокращения

3.1 В настоящем стандарте применены термины по ГОСТ Р 52210, ГОСТ Р 52591, а также следующие термины с соответствующими определениями:

3.1.1 **ввод последовательный** (Back-To-Back Insertion): Ввод двух или более непрерывных (временно) сеансов без возврата к основной службе между сеансами.

3.1.2 **видео-черное поле-видео** (video-black-video; VBV): Режим монтажа видеопрограмм без использования видеовставки между кадрами.

3.1.3 **вставка (сплайсинг, сращивание)** (Splice): Процесс замещения основного канала каналом ввода; характеризуется точкой входа в вставку и точкой выхода из вставки.

3.1.4 **вход в вставку (точка входа в вставку)** (Splice-in): Точка начала вставки или время начала вставки, указанные в сообщении Splice_Request.

3.1.5 **выход из вставки (точка выхода из вставки)** (Splice-out): Точка окончания вставки или время окончания вставки. Ожидаемое время окончания вставки вычисляется добавлением к времени начала вставки продолжительности вставки, определенной в сообщении Splice_Request.

3.1.6 **выходной канал** (Output Channel): Канал, который формируется на выходе сплайсера.

3.1.7 **выходной мультиплекс** (Output Multiplex): Транспортный поток MPEG-2, произведенный мультиплексированием одного или нескольких выходных каналов.

3.1.8 **идентификатор типа пакета** (packet identifier; PID): Тринадцатибитовый указатель в заголовке транспортного пакета MPEG-2, определяющий принадлежность пакета тому или иному потоку данных.

3.1.9 **интерфейс прикладных программ (прикладной программный интерфейс)** (Application Program Interface; API): Набор программ, используемых приложением для управления системными процедурами.

3.1.10 **кадр перехода** (post black): Условное обозначение кадра на границе между основным каналом и каналом ввода.

3.1.11 **канал** (Channel): Синоним «Служба» в терминологии DVB или синоним «Программа» в терминологии MPEG.

3.1.12 **канал ввода** (Insertion Channel): Канал мультиплекса ввода, который заменяет основной канал на полном интервале вставки события или на части этого интервала.

3.1.13 **контент** (content): Содержание, мультимедийный продукт канала ввода (например: рекламное объявление, анонс услуг общего пользования, замещающая телевизионная программа или материал программы, созданный путем соединения частей программы с сервера).

3.1.14 **мультиплекс** (Multiplex): Один канал или совокупность нескольких каналов, которые могут включать сопряженную информацию о службе. Мультиплекс представляет собой транспортный поток MPEG-2 за исключением случая мультиплекса ввода.

3.1.15 **мультиплекс ввода** (Insertion Multiplex): Мультиплекс, содержащий канал ввода. Мультиплекс может быть произведен сервером (в ряде случаев с исключением программно-зависимой информации (Program Specific Information; PSI)), в таких случаях мультиплекс может быть несовместимым с транспортным потоком MPEG-2.

3.1.16 **медиа** (media): В контексте стандарта – информационные сообщения, передаваемые по каналам вещания (кадры звука MPEG, кадры изображения MPEG, кадры изображения JPEG, файлы текста, субтитров, загружаемых шрифтов, графическая информация в формате PNG).

3.1.17 **основной канал (основная служба)** (Primary Channel): Канал основного мультиплекса, который заменяется полностью или частично каналом ввода. Единственный основной канал может порождать множество выходных каналов.

3.1.18 **основной мультиплекс** (Primary Multiplex): Источник основного канала или основных каналов.

3.1.19 **протокол обнаружения слушателей (узлов) многоадресной передачи** (Multicast Listener Discovery; MLD): один из протоколов в стеке протоколов IPV6. Описан в стандарте IETF [1].

3.1.20 **пользователь** (user): Оконечная система, которая может передавать или принимать информацию от других таких же оконечных систем с использованием сети и которая может функционировать как клиент, сервер или как клиент и сервер одновременно.

3.1.21 **последовательный ввод** (Back-To-Back Insertion): Ввод двух или более непрерывных сеансов (на локальном интервале времени) без возврата к основной службе между сеансами.

3.1.22 **поток битов DVB**: Собирательный термин, относящийся к потокам, формируемым кодерами, совместимыми со стандартами DVB.

3.1.23 **приложение** (application): 1. Программное обеспечение, предоставляющее клиенту возможность решения определенной задачи и реализуемое в среде клиента. 2. Функциональная

реализация программного обеспечения, обслуживающего один или несколько взаимодействующих аппаратных объектов.

3.1.24 **программный поток данных** (Program Stream; PS): Поток данных, образованный путем мультиплексирования элементарных потоков видеоданных и звукоданных цифрового вещательного телевидения, имеющих одну общую тактовую частоту, и сформированный из программных пакетов вещательного телевидения переменной длины.

3.1.25 **сеанс** (Session): Вставка (ввод) контента. Каждый сеанс идентифицирован уникальным SessionID.

3.1.26 **секция** (section): Синтаксическая структура, используемая для отображения всей сервисной информации в пакетах транспортного потока.

3.1.27 **семантика** (semantics): Система правил, предназначенная для определения смысловых значений отдельных конструкций алгоритмического языка.

3.1.28 **сервер** (server): Устройство, которое порождает канал ввода (или каналы ввода) для вставки в основной канал (или в основные каналы). Сервер обменивается со сплайсером информацией о времени вставки конкретных каналов ввода.

3.1.29 **сервис (служба, услуга)** (service): 1. Последовательность программ, которая под управлением вещателя может быть в режиме вещания передана как часть расписания. 2. Логический объект в системе предоставляемых функций и интерфейсов, поддерживающий одно или множество приложений, отличие которого от других объектов заключается в доступе конечного пользователя к управлению шлюзом сервисов.

3.1.30 **синтаксис** (syntax): Часть языка программирования, которая описывает структуру программ как наборов символов.

3.1.31 **сокет** (socket): Нестандартный программный интерфейс между прикладной программой и стеком протоколов TCP/IP.

3.1.32 **соединение API** (API Connection): Соединение, устанавливаемое между сервером и сплайсером через TCP/IP сокет, для передачи сообщений API.

3.1.33 **сплайсер** (splicer): Устройство, которое вставляет каналы ввода в основной канал (или в основные каналы). Допускается работа сплайсера по сообщениям меток стандарта ANSI/SCTE [2]. Сплайсер обменивается с сервером информацией о времени вставки конкретных каналов ввода.

3.1.34 **транспортный поток**; ТП (transport stream; TS): Набор из нескольких программных потоков данных цифрового вещательного телевидения, сформированный из программных пакетов постоянной длины с коррекцией ошибок и независимым тактированием от своих источников синхронизации. Параметры транспортного потока определяются стандартом ISO/IEC [3] (2.4).

3.1.35 **пакетированный элементарный поток**; ПЭП (Packetized Elementary Stream; PES): Пакетированный элементарный поток, в котором данные разбиты на пакеты и снабжены заголовками.

3.1.36 **I-кадр черного поля и кадр «тихого» звука** (black video and muted audio; BVMA): Кадры, отделяющие данные контента канала ввода от данных контента основного канала в точке выхода.

3.2 В настоящем стандарте применены следующие сокращения:

ВПР – время вещания, пригодное для размещения рекламы;

МЭК (International Electrotechnical Commission / Committee, IEC) – Международная электротехническая комиссия;

МСЭ (International Telecommunications Union, ITU) – Международный союз электросвязи;

НТВ – ОАО «Телекомпания НТВ»;

ПЭП (Packetized Elementary Stream, PES) – пакетированный элементарный поток;

ТП (Transport Stream, TS) – транспортный поток (цифрового вещательного телевидения);

ЭП (Elementary Stream, ES) – элементарный поток;

AAL (ATM Adaptation Level) – уровень адаптации ATM;

AC-3 (Dolby AC-3 audio coding system) – система кодирования аудио Dolby;

API (Application Program Interface) – интерфейс прикладных программ (прикладной программный интерфейс);

ASCII (American Standard Code for Information Interchange) – Американский стандартный код обмена информацией;

ATM (Asynchronous Transfer Mode) – режим асинхронной передачи;

AVC (Audio Video Coding) – стандарт кодирования H.264/AVC;

BVMA (black video and muted audio) – I-кадр черного поля и кадр «тихого» (отключенного) звука;

CNN (Cable News Network) – сеть кабельного вещания США; Си-Эн-Эн;

CRC (Cyclic Redundancy Check) – циклический контроль по четности;

DTS (Decoder Time Stamp) – временная метка декодирования;

DVB (Digital Video Broadcasting) – цифровое телевизионное вещание;

EAS (Emergency Alert System) – чрезвычайная система предупреждений;
ES (Elementary Stream) – элементарный поток; ЭП;
JPEG (Joint Photographic Experts Group) – Объединенная экспертная группа по фотографии (стандарт на алгоритм компрессии неподвижных полутоновых и цветных изображений);
IEC (International Electrotechnical Commission / Committee) – Международная электротехническая комиссия; МЭК;
IANA (Internet Assignned Numbers Auhority) – центр по присвоению имен в Интернете;
ID (Identifier) – идентификатор;
IGMP (Internet Group Management Protocol) – протокол управления группами в сети Интернет; версия протокола IGMPv3 определяется в соответствии со стандартом IETF [4];
IP (Internet Protocol) – Интернет протокол;
IPV4 (Internet Protocol version 4) – Интернет протокол версия 4;
IPV6 (Internet Protocol version 6) – Интернет протокол версия 6;
ISO (International Standards Organizations) – Международная организация по стандартизации;
ITU (International Telecommunications Union) – Международный союз электросвязи; МСЭ;
ITU-T (International Telecommunications Union – Telecommunication Standardization Sector) – Сектор стандартизации электросвязи МСЭ;
GPS (Global Positioning System) – Глобальная система позиционирования;
MAC-адрес (Media Access Control) – уникальный идентификатор, присваиваемый каждой единице оборудования компьютерных сетей;
MLD (Multicast Listener Discovery) – слушатель (получатель) многоадресной передачи;
MPEG (Motion Pictures Expert Group) – группа экспертов по движущимся изображениям (группа стандартов сжатия аудио- и видеоинформации);
MPEG-2 – стандарт цифрового сжатия аудио- и видеоинформации;
MPTS (Multi-Program Transport Stream) – многопрограммный транспортный поток;
MUX (Multiplexer) – мультиплексор;
NTP (Network Time Protocol) – Сетевой Протокол Системного Времени;
PAT (Program Association Table) – таблица ассоциации программ;
PCR (Program Clock Reference) – ссылка на программные часы;
PES (Packetized Elementary Stream) – пакетированный элементарный поток; ПЭП;
PID (Packet Identifier) – идентификатор типа пакета;
PNG (Portable Network Graphic) – переносимая сетевая графика;
PMT (Program Map Table) – таблица состава программы;
PS (Program Stream) – программный поток данных;
PSI (Program Specific Information) – программно-зависимая информация;
PTS (Presentation TimeStamp) – метка времени представления;
RAM (Ramdom Access Memory) – оперативное запоминающее устройство с произвольным доступом;
SCTE (Society of Cable Telecommunications Engineers) – общество инженеров кабельных телекоммуникаций;
SDT (Service Description Table) – таблица описания служб;
SI (Service Information) – информация о службах;
SPTS (Single Program Transport Stream) – однопрограммный транспортный поток;
TCP (Transmission Control Protocol) – протокол управления передачей (из стека протоколов TCP/IP);
TCP/IP – стек протоколов сетевого и транспортного уровня;
TS (Transport Stream) – транспортный поток (цифрового вещательного телевидения); ТП;
UTC (Coordinated Universal Time) – всемирное координированное время;
V2 – версия 2;
V3 – версия 3;
VBV (video-black-video) – видео-черное поле-видео;
VCI (Virtual Channel Identification) – Идентификатор Виртуального Канала;
VOD (Video on Demand) – видео по требованию;
VPI (Virtual Path Identification) – Идентификатор виртуального пути.

3.3 В настоящем стандарте применены следующие аббревиатуры:

tcimsbf – дополнение до двух целого, старший бит следует первым;
uimsbf – целое число без знака, старший бит следует первым.

4 Система вставки

4.1 Вводная часть. Общие сведения о системе

В состав оборудования цифровой вставки входят серверы и сплайсеры, которые при использовании API выполняют вставки контента в выходные данные мультиплекса MPEG-2. API может использоваться при различных конфигурациях оборудования цифровой вставки:

- с единственным сервером и единственным сплайсером;
- с несколькими серверами и сплайсерами.

На рисунке 1 представлена структурная схема оборудования цифровой вставки в конфигурации с несколькими серверами и сплайсерами.

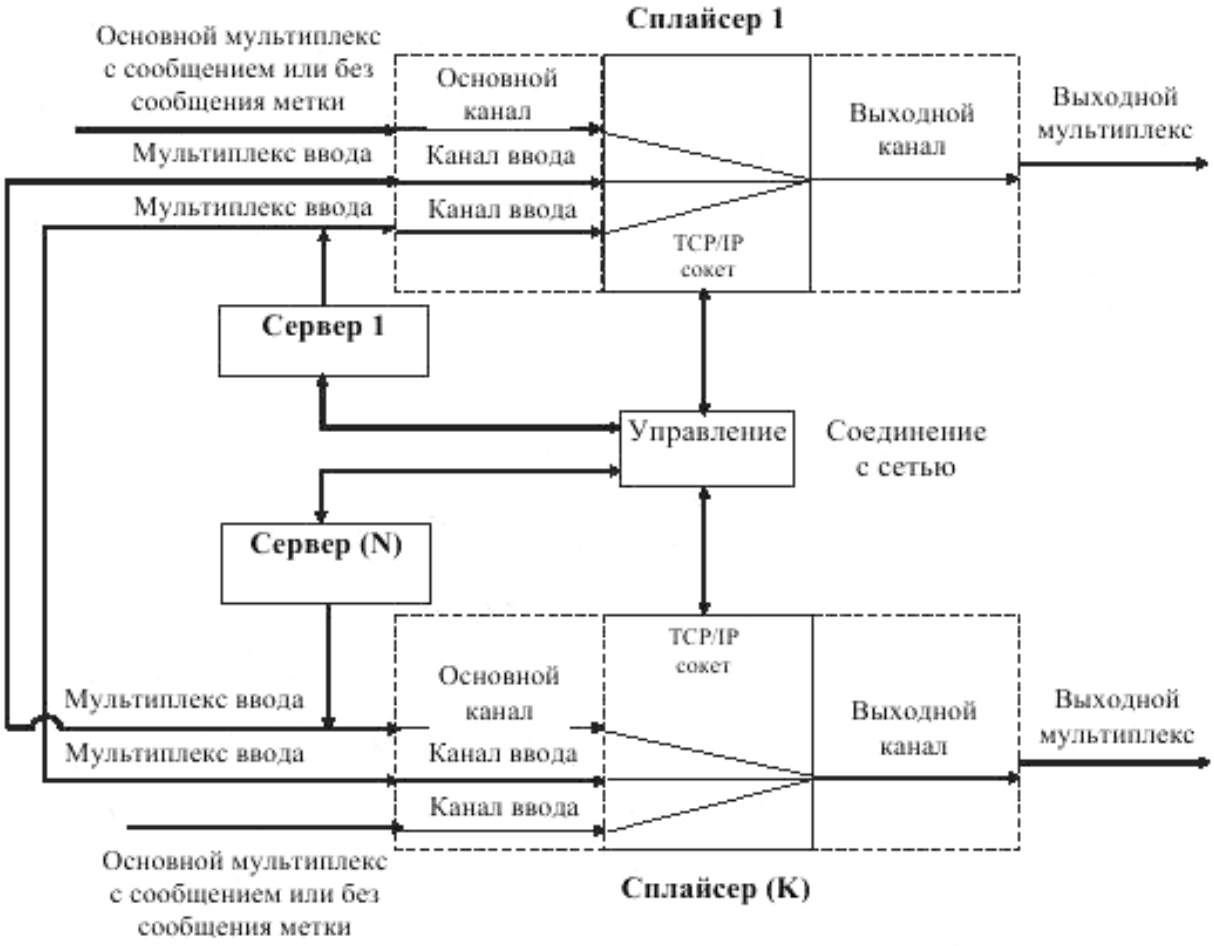

Рисунок 1 – Конфигурация с несколькими серверами и сплайсерами

В состав этой модели входят K сплайсеров и N серверов. На входы сплайсеров поступают основные мультиплексы; от серверов на сплайсеры поступают мультиплексы ввода.

Сплайсер логически разделяет каналы мультиплексов и подает их на полнодоступный коммутатор (на рисунке не показан). В исходном состоянии основные каналы скоммутированы на выходные каналы. Сервер может инициировать сплайсер на отключение основного канала от выходного канала с подключением к выходному каналу канала ввода на интервал времени определенной продолжительности. Сервер может инициировать сплайсер на переключение на другой канал ввода после первоначального переключения. Сплайсер выполняет сращивание элементарных потоков (аудио, видео и данные) и создает на своем выходе транспортный поток MPEG-2, выполненный мультиплексированием одного или нескольких выходных каналов. Выходные каналы должны быть совместимыми со стандартом ISO/IEC [3].

Сплайсер выполняет сращивание элементарных потоков (аудио, видео и данные) в точке, оптимальной по времени.

Сплайсер может выполнять вставку контента, который хранится только на сервере и поступает на сплайсер в единственном входном мультиплексе. Это дает возможность использовать данный API в ситуации, когда на выходе сервера формируется один многопрограммный транспортный поток (MPTS), который содержит программу и промежуточный материал для вставки. В этом случае API может использовать сплайсер для создания необходимых вставок между контентом.

API поддерживает все варианты взаимодействия одиночного сервера или нескольких серверов с одиночным сплайсером или с несколькими сплайсерами.

В некоторых случаях на вход сплайсера может быть подключено либо несколько серверов либо несколько каналов в мультиплексе ввода. В этих случаях сплайсер будет связан с выходным каналом несколькими Соединениями API. Если в основном канале принимается сообщение метки стандарта ANSI/SCTE [2], то сообщение **Cue_Request** должно быть отправлено на серверы по всем Соединениям API, которые были выполнены для соответствующего выходного канала. Допускается передача сообщения **Splice_Request** для одной и той же вставки для выходного канала одновременно более чем через одно Соединение API.

4.2 Приоритеты вставки каналов ввода

При появлении на входе сплайсера одновременно нескольких каналов ввода возникает конфликтная ситуация, которая разрешается применением нескольких уровней доступа каналов ввода с уровнем приоритета от 0 до 9, что гарантирует использование корректного по уровню доступа канала ввода. Уровень доступа каналов ввода с уровнем приоритета 0 обладает самым низким приоритетом. Уровень доступа каналов ввода с уровнем приоритета 9 обладает самым высоким приоритетом, который может переопределить соединение с более низким приоритетом. Флаг **OverridePlaying** в сообщении **Splice_Request** определяет приоритет вставки в тот момент, когда сплайсер ставит канал ввода в очередь на вставку или выполняет вставку. Если флаг установлен в «1», то ввод с более высоким приоритетом может прервать или понизить приоритет канала ввода, который в настоящий момент проигрывается. Если флаг установлен в «0», сплайсер не будет заменять вставку, проигрываемую настоящий момент, даже если новый запрос будет иметь более высокий приоритет.

Сообщение **Splice_Request** должно быть отправлено не менее чем за 3 с до времени вставки (splice time( )). Если это условие не выполняется, то стандарт не определяет последствия обработки сообщения **Splice_Request** данным API. Если несколько серверов инициируют запросы вставки в один и тот же интервал времени с одним и тем же приоритетом, сплайсер расположит приоритеты запросов в порядке поступления. Все другие запросы будут отвергаться, и ошибка коллизии будет отправлена в сообщении **Splice_Response** (если не установлен флаг **OverridePlaying**).

Например, на интервале времени, предшествующем инициированию вставки, могут иметь место следующие ситуации:

- если запрос с приоритетом 5 **Splice_Request** получен одновременно с запросом приоритета 3 **Splice_Request**, то фиксируется ошибка коллизии, которая возвращается в ответ на запрос приоритета 3;

- если запрос с приоритетом 7 **Splice_Request** получен позже запроса с приоритетом 5 в том же интервале времени, ошибка коллизии возвращается на запрос приоритета 5 и запрос приоритета 7 поставлен в очередь;

- если второй запрос с приоритетом 7 получен с флагом **OverridePlaying**, установленным в 0, тогда второй запрос приоритета 7 примет ошибку коллизии.

Однако если флаг **OverridePlaying** установлен в 1 во втором запросе с приоритетом 7, первоначальный запрос с приоритетом 7 получит ошибку коллизии и будет переопределен.

На рисунке 2 показана диаграмма обработки запросов двух каналов ввода сплайсера. Затененные области на диаграмме обозначают каналы ввода, которые будут направлены к выходному каналу в указанные моменты времени.

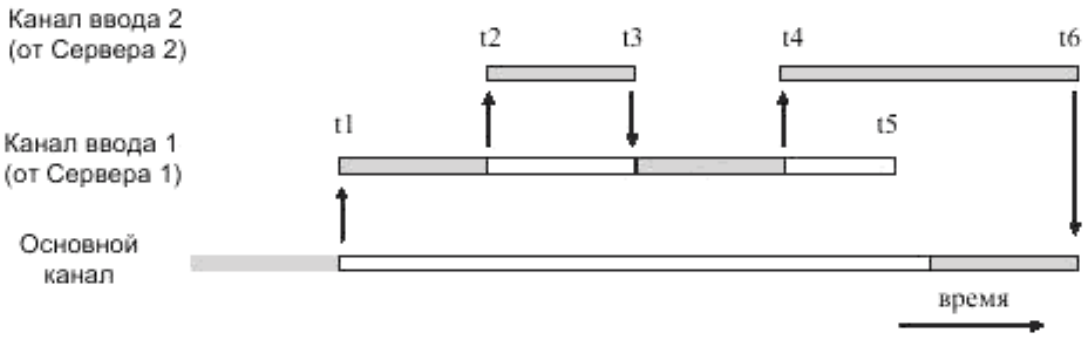

Рисунок 2 – Операция флага **OverridePlaying**

Момент времени t1: Сервер 1 выпускает сообщение **Splice_Request** и начинает передавать свой поток к сплайсеру. Сплайсер переключает этот поток канала ввода на выходной канал. В сообщении **Splice_Request** был сделан запрос о вставке продолжительностью от времени t1 до времени t5. Сплайсер должен отправить Серверу 1 сообщение **SpliceComplete_Response** с **SpliceType** с установкой флага к **Splice_in** и с установкой значения Кода Результата «100»: «Успешный Ответ».

Момент времени t2: Сервер 2 выпускает сообщение **Splice_Request** с флагом **OverridePlaying** установленным в «1», которое имеет равный или более высокий приоритет (по сравнению с каналом ввода (от Сервера 1)). Во время, определенное в сообщении **Splice_Request**, поток канала ввода 2 переключен на выходной канал (заменяет продолжающийся поток от Сервера 1). Сервер 2 сообщением **Splice_Request** запросил вставку продолжительностью от времени t2 до времени t3. Сплайсер должен отправить Серверу 1 сообщение **SpliceComplete_Response** с флагом **SpliceType**, установленным в Splice_out и с Кодом Результата «125»: «Переопределение Канала». Сплайсер отправляет Серверу 2 сообщение **SpliceComplete_Response** с установкой флага **SpliceType** в Splice_in и устанавливает значение Кода Результата «100»: «Успешный Ответ». Если Сервер 1 решает, что переопределение канала является ошибкой, он может отправить сообщение **Abort_Request** и завершить поток.

Момент времени t3: Вставка от Сервера 2 завершена, и сплайсер возвращается к материалу от Сервера 1, направляя его к выходному каналу. Сплайсер не переключает основной канал к выходному каналу. Сплайсер передает Серверу 1 сообщение **SpliceComplete_Response** с набором флага **SpliceType** к Splice_in и устанавливает значение Кода Результата «125»: «Переопределение канала». Сплайсер передает на Сервер 2 сообщение **SpliceComplete_Response** с установкой флага **SpliceType** к Splice_out и устанавливает значение Кода Результата «100»: «Успешный Ответ».

Момент времени t4: Сервер 2 выпускает сообщение **Splice_Request** с установкой флага **OverridePlaying,** установленным в «1». Во время, определенное сообщением **Splice_Request**, поток канала ввода 2 переключается на выходной канал (заменяя поток от Сервера 1). Сервер 2 сообщением **Splice_Request** запрашивает вставку продолжительностью от времени t4 до времени t6. Сплайсер должен отправить Серверу 1 сообщение **SpliceComplete_Response** с флагом **SpliceType**, установленным в Splice_out и со значением Кода Результата «125»: «Переопределение канала». Сплайсер отправляет на Сервер 2 сообщение **SpliceComplete_Response** с установкой флага **SpliceType**, установленным в Splice_in, и со значением Кода Результата «100»: «Успешный Ответ».

Момент времени t5: Сервер 1 заканчивает проигрывать вторую часть потока вставки, переопределенную Сервером 2.

Момент времени t6: Заключительная часть вставки завершена, сплайсер возвращается к материалу от основного канала и направляет его к выходному каналу. Сплайсер передает Серверу 2 сообщение **SpliceComplete_Response** с установкой флага **SpliceType** к Splice_out и устанавливает значение Кода Результата «100»: «Успешный Ответ».

Возможна ситуация, когда несколько серверов должны будут разделить сообщение **Cue_Request** с возможностью вставки общей продолжительностью 60 с, когда один Сервер будет использовать первые 30 с, а второй Сервер будет использовать последние 30 с. В зависимости от приоритетов и в случае, если будут приняты сообщения **Splice_Request,** сплайсер должен установить Код Результата «109» «Коллизия вставки», если она возникнет. Настоящий стандарт не устанавливает параметры API, обеспечивающие возможность координации двух Серверов. Указанная координации может быть выполнена по взаимному соглашению между Серверами или API в версии сервер – сервер.

4.3 Аномальные завершения вставки

Было показано, что во время воспроизведения (проигрывания) вставка будет переопределена вставкой с более высоким приоритетом. В этом случае сплайсер должен возвратиться к переопределенной вставке при окончании вставки с более высоким приоритетом. Если вставка с более высоким приоритетом прервана сообщением **Abort_Request**, сплайсер должен возвратиться к переопределенной вставке. Если первоначальный канал ввода более не доступен, то сплайсер не должен возвращаться к основному каналу, если это возможно.

Если сервер запрашивает вставку на основном канале, в котором в настоящий момент нет допустимых данных, сплайсер должен выполнить вставку и в сообщении **SpliceComplete_Response** к серверу сообщить Код Результата «111»: «Не найден основной канал». Аналогичным образом вставка от канала ввода назад к основному каналу, у которого нет допустимых (правильных) данных, завершится с Кодом Результата «111»: «Не найден основной канал».

Разработчикам сплайсера рекомендуется, устанавливать дополнительное программное обеспечение, которое всегда приводило бы сплайсер к возврату к основному каналу. Это позволит повысить устойчивость сплайсера к отказам основного канала при любых ошибках, которые могли бы привести к прекращению передачи выходного канала.

4.4 Требования к процессу вставки

Сплайсеру необходима информация о канале ввода до начала процедуры вставки канала ввода в основной канал. Часть этой информации должна быть отправлена в Соединение API, часть этой информации может быть отправлена в мультиплексе MPEG. Вся информация должна запрашиваться перед вставкой.

Поле **ChannelName** используется для идентификации выходного канала. Поле **ChannelName** содержит уникальное имя, присвоенное каждому выходному каналу (например, HTB или CNN), оно устанавливается в сплайсер, и необходимо серверу для определения основного канала, который будет заменен каналом ввода.

Сплайсер должен знать канал ввода, который необходимо вставлять в основной канал. Эта информация должна включать данные о расположении мультиплекса ввода и расположения канала в мультиплексе ввода. Эта информация доступна в сообщении **Splice_Request**.

4.5 Параметры передачи информации между сервером и сплайсером

Передачу информации между сервером и сплайсером производят через один TCP/IP сокет. После установления Соединения API оно сохраняется до тех пор, пока одно из устройств его не завершит. Для повторной инициализации требуется новая установка соединения.

Все сообщения, которыми обмениваются сплайсер и сервер, используют общий формат, детализированный в разделе 7 настоящего стандарта. Для передачи между сплайсером и сервером необходимо использовать только сообщения, поддерживающие этот формат. Формат разрешает класс сообщений типа «Определен пользователем», который может использоваться в качестве шаблона для частных сообщений между сплайсером и сервером. Нормирование этого класса сообщений настоящим стандартом не предусмотрено.

Все сообщения запроса требуют ответа или от сплайсера, или от сервера в зависимости от того, какое устройство обращается с просьбой. Большая часть ответов на сообщения указыват на результат и не содержит никаких других данных, но они необходимы для запрашивающего как гарантия того, что сообщение получено и интерпретировано правильно. При появлении сбоев при обмене передача сообщения может быть повторена.

5 Синтаксис API

В разделе устанавливаются параметры синтаксиса сообщений и функций:

- сообщения Splicing_API_Message;
- функции передачи сообщений;
- функции инициализации передачи сообщений:
- сообщение запроса Init_Request;
- сообщение ответа Init_Response;
- встроенные сообщения меток Cue_Request;
- сообщения вставки:

- Splice_Request;
- Splice_Response;
- SpliceComplete_Response;
- сообщения о состоянии (статусе) сплайсера:
- Alive_Request;
- Alive_Response;
- сообщения расширенных данных:
- ExtendedData_Request;
- ExtendedData_Response;
- сообщения прерывания:
- прерывания вставок Abort_Request;
- Abort_Response;
- TearDownFeed_Request;
- TearDownFeed_Response;
- сообщения запроса параметров конфигурации:
- GetConfig_Request;
- GetConfig_Response;
- General_Response.

5.1 Синтаксис сообщения Splicing_API_Message

Все сообщения API настоящего стандарта имеют общую структуру, которая является оболочкой данных конкретного отправляемого сообщения.

Сообщение **Splicing_API_Message** кодируется в соответствии с таблицей 1.

Таблица 1 – Кодирование сообщения **Splicing_API_Message**

Синтаксис	Количество байтов	Тип
Splicing_API_Message {		
MessageID	2	uimsbf
MessageSize	2	uimsbf
Result	2	uimsbf
Result_Extension	2	uimsbf
data( )	*	*
}		

MessageID: Значение поля указывает на параметры отправляемого сообщения в соответствии с таблицей 2.

Таблица 2 – Параметры отправляемого сообщения в зависимости от значения поля **MessageID**

Значение поля **MessageID**	Имя сообщения	Источник сообщения	Описание
0x0000	General_Response	Сплайсер или сервер	Используется для передачи асинхронной информации между устройствами. Это не данные (data( )), связанные с этим сообщением
0x0001	Init_Request	Сервер	Первоначальное сообщение сплайсеру на порт 5168
0x0002	Init_Response	Сплайсер	Первоначальный ответ серверу на установленное соединение
0x0003	ExtendedData_ Request	Сервер	Запрос на детализированное воспроизведение информации от сплайсера.
0x0004	ExtendedData_Response	Сплайсер	По запросу сервера отправляет на сервер структуру, содержащую детализированные данные о воспроизведении
0x0005	Alive_Request	Сервер	Отправляет сообщение запроса статуса сплайсера для того, чтобы получить оценку его текущего статуса
0x0006	Alive_Response	Сплайсер	Ответ на запрос текущего статуса сплайсера
0x0007	Splice_Request	Сервер	Запрос вставки в определенное время
0x0008	Splice_Response	Сплайсер	Ответ, указывающий что **Splice_Request** был получен и сплайсер обрабатывает вставку
0x0009	SpliceCoplete_ Response	Сплайсер	Ответ о начале вставки и об окончании вставки
0x000A	GetConfig_Request	Сервер	Запрос для получения конфигурации текущей вставки для этого Соединения API
0x000B	GetConfig_Response	Сплайсер	Содержит всю информацию о вставке для Соединения API
0x000C	Cue_Request Splicer	Сплайсер	Сплайсер, отправляющий секцию метки серверу
0x000D	Cue_Response	Сервер	Подтверждение, что секция метки была получена
0x000E	Abort_Request	Сервер	Запрос немедленного возврата к основному каналу или к переопределению канала ввода
0x000F	Abort_Response	Сплайсер	Подтверждение получения сообщения **Abort_Request**. В случае необходимости должно быть сгенерировано сообщение SpliceComplete_Response
0x0010	TearDownFeed_Request	Сервер	Запрос на удаление выходного канала, созданный с помощью дескриптора create_feed_descriptor( )
0x0011	TearDownFeed_Response	Сплайсер	Ответ, показывающий, что выходной канал был удален
0x0012-0x7FFF	Зарезервировано		Диапазон значений зарезервирован для будущей стандартизации
0x8000 - 0xFFFE	Определяется пользователем		Диапазон функций, доступный для определения пользователем

MessageSize: Размер поля data( ) в байтах.

Result: Поле передается в ответ на запрашиваемое сообщение. Детализация Кодов Результата в соответствии с приложением А. На сообщениях запроса в поле установлено 0xFFFF.

Result_Extension: В поле должно быть установлено 0xFFFF, если в сообщении ответа не передается информация дополнительного результата.

data( ): Специфическая структура данных отправляемого сообщения. Подробности о каждом из сообщений содержат данные, описанные ниже. Размер этого поля указан в поле **MessageSize** и определен размером данных, добавляемых к сообщению. Не все сообщения используют поле data( ).

5.2 Требования и соглашения передачи сообщений

Передача сообщений выполняется при следующих требованиях и соглашениях:

- каждое сообщение, которое содержит данные с полями данных и типами данных, в общих чертах описано ниже. Дополнительные структуры описаны в разделе 7 настоящего стандарта;

- все строки имеют пространство, зарезервированное для нулевого символа окончания строки, и должны завершаться нулевым символом. Например, строка, длина которой определена в 16 символов, может иметь не больше 15 символов данных, сопровождаемых символом нуля (0x00) сразу после последнего символа данных. После символа нуль в остальной части строки символы имеют произвольное значение. Размер строки, является постоянным и не изменяется в зависимости от ее длины. В настоящем стандарте в строке используются символы ASCII на 8 битов;

- все значения времени устанавливаются в соответствии с UTC;

- в поле, состояние которого безразлично, устанавливаются только 1. В 4-байтовом поле это значение было бы 0xFFFFFFFF;

- сообщения ответа должны отсылаться без неоправданных задержек. Устройство, должно ожидать ответ в интервале 5 с, не указывая на ошибку задержки (тайм-аут). Когда сервер подозревает тайм-аут, он должен отправить сообщение Alive_Request. Если сплайсер не дает ответа, предусмотренного настоящим стандартом, соединение для этого канала должно быть удалено и восстановлено;

- сервер, принимающий сообщение ответа, указывающее на отказ проанализировать сообщение (код ошибки 123) должен передать сообщение Alive_Request. Если он не получит соответствующее сообщение Alive_Response, соединение для этого канала должно быть удалено и восстановлено;

- поле Result в сообщении Splicing_API_Message используется для возврата Кода Результата. Множество кодов ответа могут быть возвращены в любое время отправлением нескольких сообщений **General_Response**;

- если сплайсер или сервер не могут проанализировать сообщение запроса, они должны возвратить **General_Response** с Кодом Результата «123». Имя результата в соответствии с таблицей А.1 приложения А настоящего стандарта.

5.3 Инициализация передачи сообщений

Первоначальное сообщение начинается со сплайсера, прослушиванием порта 5168, и сервером, открывающим Соединение API со сплайсером. Сервер отправляет сплайсеру сообщение **Init_Request**. После этого сервер прислушивается к ответу от сплайсера по установленному Соединению API. Все дальнейшие передачи выполняются на этом Соединении API. Сплайсер или сервер могут завершить связь, закрывая это Соединение API. Каждое устройство ответственно за то, что обнаружило и должным образом обработало Соединение API.

Когда сплайсер подготавливает к работе приемную сторону TCP, порт 5168, он должен увеличить не менее чем в три раза число каналов ввода для Соединений API со сплайсером. Например, если сплайсер управляет 70 каналами, из которых 40 пригодны для вставки, то он должен предусмотреть одновременное подключение 120 API.

5.3.1 Сообщение запроса **Init_Request**

Поле data( ) этого сообщения содержит структуру Init_Request_Data, приведенную в таблице 3.

Таблица 3 – Структура Init_Request_Data поля data( ) сообщения запроса **Init_Request**

Синтаксис	Количество байтов	Тип

Init_Request_Data {
Version( )
ChannelName
SplicerName
Hardware_Config( )
для (i=0; i <N; i++)
splice_API_descriptor( )
}
	
	32
	32
		
		Строка
		Строка

Version( ): В соответствии с подразделом 6.1 настоящего стандарта.

ChannelName: Логическое имя выходного канала этого соединения. Оно также используется для проверки корректности Соединения API при ответе сплайсера серверу.

SplicerName: Имя сплайсера для того случая, когда сервер использует API для связи с устройством, управляющим несколькими сплайсерами.

Hardware_Config( ): В соответствии с подразделом 6.2 настоящего стандарта.

splice_API_descriptor( ): Синтаксис этого дескриптора должен быть в соответствии с подразделом 6.5 настоящего стандарта. Для запроса Init_Request может использоваться дескриптор missing_Primary_Channel_action_descriptor( ).

5.3.2 Сообщение ответа **Init_Response**

После того, как сервером отправлен запрос **Init_Request**, сплайсер по открытому Соединению API отправляет сообщение **Init_Response**. Прием этого сообщения позволяет серверу проверить, что версия, отправленная сплайсеру, поддерживается и что у него есть Соединение API с корректным основным каналом. Поле data( ) этого сообщения содержит структуру Init_Response_Data, приведенную в таблице 4.

Таблица 4 – Структура Init_Response_Data поля data( ) сообщения ответа **Init_Response**

Синтаксис	Количество байтов	Тип

Init_Response_Data {
Version( )
ChannelName
}
	
	32
		
		Строка

Version( ): В соответствии с подразделом 6.1 настоящего стандарта. Сплайсер должен сообщить самый высокий номер версии API, которую он может поддерживать.

ChannelName: Поле возвращается сервером, что указывает на корректное выполнение соединения.

5.4 Встроенные сообщения меток

Сплайсеры имеют возможность получать встроенные сообщения метки, предусмотренные стандартом ANSI/SCTE [2]. При получении сплайсером сообщений метки они должны быть переданы на сервер. Сообщение **Cue_Request** используется для передачи сообщений метки

от сплайсера к серверу. Когда сплайсер получает сообщение метки, он отправляет серверу всю секцию splice_info_section( ), включающую время вставки. Сервер подтвердит прием этого сообщения сообщением **Cue_Response**. Сообщение **Cue_Response** состоит только из сообщений Splicing_API_Message и не имеет связанных данных data( ), но может иметь код возврата. Сплайсер должен дешифровать секцию splice_info_section( ) (если она будет зашифрована) прежде, чем отправить ее серверу.

Если сплайсер получит сообщение метки, имеющее недопустимое значение CRC, то он должен отправить **General_Message** к серверу с Кодом Результата «117»: «Недопустимое сообщение метки». В этом случае сплайсер не должен отправлять сообщение **Cue_Request**.

Допускается возможность изменения сплайсером конфигурации сообщения **Cue_Request** в соответствии со стандартом ANSI/SCTE [2]. Конфигурации сообщения могут включать:

- возможность передачи сообщения новых версий стандарта ANSI/SCTE [2], которые сплайсер в старой версии стандарта ANSI/SCTE [2] не понимает;

- возможность не передавать сообщения резервирования пропускной способности. Резервирование пропускной способности должно выполняться «по умолчанию»;

- возможность не передавать сообщения Splice_Null, если к ним не присоединены дескрипторы;

- возможность не передавать сообщения, которые не могут быть дешифрованы.

Поле data( ) сообщения **Cue_Request** содержат структуру Cue_Request_Data, приведенную в таблице 5.

Таблица 5 – Структура Cue_Request_Data поля data( ) сообщения меток **Cue_Request**

Синтаксис

Cue_Request_Data {
time( )
splice_info_section( )
}

time( ): Значение поля формируется из поля splice_time( ) в секции

splice_info_section( ) сообщений метки сплайсера стандарта ANSI/SCTE [2]. Если в стандарте ANSI/SCTE [2] используется режим вставки компонентов splice_info_section, то поле time( ) ссылается на время вставки «по умолчанию», детализированному в стандарте ANSI/SCTE [2] (7.5.2.1). В случае, если секция splice_info_section( ) не содержит поле pts_time( ), это требует преобразования аналогичного случаю команды splice_schedule( ), тогда временная структура должна быть заполнена последовательностью «1», чтобы обозначить определенное время. Выбор правила отображения времени PTS к времени UTC для связи с сервером зависит только от сплайсера. Оно может изменяться для различных сплайсеров, чтобы должным образом управлять своими внутренними буферами. Синтаксис структуры time( ) показан в подразделе 6.4 настоящего стандарта.

Подробности структуры секции splice_info_section( ) изложены в стандарте ANSI/SCTE [2].

5.5 Сообщения вставки

После инициализации и конфигурирования сплайсера для инициирования сеанса сервер может выпустить сообщение **Splice_Request**. На сообщение от сервера **Splice_Request** сплайсер отвечает сообщениями **Splice_Response** и **SpliceComplete_Response**. Продолжительность передачи сервером сообщения **Splice_Request** должна быть не менее 3 с до окончания передачи в сообщении **Splice_Request** поля time( ). Это позволяет сплайсеру установить свою конфигурацию и приготовить вставку. Поток канала ввода для сеанса должен начаться в интервале времени между 300 и 600 мс перед полем time( ) при измерении на входе сплайсера. Ссылка на программные часы (PCR) должна быть передана не позднее первого видео модуля доступа потока канала ввода. Видеопоток контента вставки должен начаться с заголовка последовательности и I-кадра. Сплайсер должен поддерживать в очереди не менее 10 сообщений **Splice_Request** на данном Соединении API. Если очередь сообщения сплайсера будет заполнена, то сплайсер ответит Кодом Результата «114»: «Очередь вставки заполнена». Подробности физического соединения предоставлены в сообщении **Init_Request**. Предусмотрены два способа индикации канала в мультиплексе вставки и используемого PID:

- если в сообщении **Splice_Request** идентификатор **ServiceID** не равен 0xFFFF, то поле **ServiceID** определяет номер программ в PAT, который указывает на связанную PMT. PAT и PMT должны быть стабильными в канале вставки на интервале не менее 200 мс перед отправлением сообщения **Splice_Request** и должны оставаться стабильными на интервале сеанса. Они должны соответствовать расширенным версиям таблиц MPEG;

- если **ServiceID** равно 0xFFFF, то в сообщении **Splice_Request** необходимо использовать структуру splice_elementary_stream( ) (PCR, видео, аудио и PID данных).

Примечание – Если будет использован этот метод, то в поле **ServiceID** должен быть установлен 0xFFFF. Сплайсер для выходного мультиплекса должен предоставлять совместимые транспортные потоки MPEG-2 (мультиплекс ввода не должен содержать PSI).

Должен обеспечиваться следующий порядок отправления сообщений вставки. Первое сообщение из последовательного ввода передается непрерывно и должно использовать поле time( ), в то время как все другие сообщения **Splice_Request** могут использовать поле **PriorSession**. Номер поля **PriorSession** должен ссылаться на существующий сеанс, который еще не завершился. Во всех других случаях возвращается код ошибки 123, указывающий на поле PriorSession или поле time( ).

Сервер в мультиплексе ввода выбирает PID элементарных потоков. PID не могут быть общими для смежных сеансов одного и того же сервера одного и того же самого мультиплекса ввода, вследствие того, что потоки смежных сеансов могут накладываться во времени из-за требований API.

5.5.1 Сообщение вставки **Splice_Request**

Поле data( ) этого сообщения содержит структуру Splice_Request_Data, приведенную в таблице 6.

Таблица 6 – Структура Splice_Request_Data поля data( ) сообщения вставки **Splice_Request**

Синтаксис	Количество байтов	Тип
Splice_Request_Data {		
SessionID	4	uimsbf
PriorSession	4	uimsbf
time( )		
ServiceID	2	uimsbf
if (**ServiceID** = 0xFFFF)		
{		
PcrPID	2	uimsbf
PIDCount	4	uimsbf
for(j=0; j <**PidCount**; j ++)		
splice_elementary_stream( )		
}		
Duration	4	uimsbf
SpliceEventID	4	uimsbf
PostBlack	4	uimsbf
AccessType	1	uimsbf
OverridePlaying	1	uimsbf
ReturnToPriorChannel	1	uimsbf
for(i=0; i <N; i ++)		
splice_API_descriptor( )		
}		

SessionID: Идентификатор сеанса. Используется для того, чтобы отличать этот запрос от других запросов «это было или будет выпущено». Не разрешается выполнять конкурирующие сообщения **Splice_Request** с одинаковыми **SessionID**.

PriorSession: Это поле позволяет применять упрощенный метод последовательного ввода вставок. Это поле содержит **SessionID** сеанса, который ему предшествует. Значение этого поля 0xFFFFFFFF указывает, что этот сеанс использует поле time( ) для инициирования вставки, а не **SessionID** предыдущего сеанса. Это поле будет иметь допустимый **SessionID** только в том случае, если предыдущий сеанс выполнен тем же самым сервером. Для создания режима последовательного ввода вставок от множества серверов должно использоваться поле time( ), но не поле **PriorSession**.

time( ): Время вставки события. Это поле, как правило, соответствует полю time( ) сообщения **Cue_Request** (от сплайсера). Если событие не было инициировано **Cue_Request**, тогда оно определяет время, когда сервер форсирует (вынуждает) вставку события. Это поле должно быть игнорировано, если **PriorSession** не равен 0xFFFFFFFF. Если это значение не связано с сообщением метки стандарта ANSI/SCTE [2], то могут возникать различия между сплайсерами в зависимости от буфера и модели фактического выполнения вставки. Синтаксис структуры поля time( ) приведен в подразделе 6.4 настоящего стандарта.

ServiceID: Номер канала в мультиплексе ввода, который будет вставлен вместо основного канала. Если поле имеет значение 0xFFFF, то необходимо использовать поля splice_elementary_stream( ) и **PIDCount**.

PCR: Указывает на PID PCR.

PIDCount: Указывает количество PID в канале ввода (не считая PID PCR).

Duration (продолжительность): Количество тактов системных часов на 90 кГц, которое сервер запрашивает у сплайсера для вставки. Это поле может быть использовано для корректировки значения продолжительности по стандарту ANSI/SCTE [2]. В этом поле может быть установлен «0», чтобы указать, что сплайсер должен переключиться на канал ввода до прибытия новых сообщений **Splice_Request** или **Abort_Request**.

SpliceEventID: Это поле используется, чтобы связать это событие вставки с сообщением метки стандарта ANSI/SCTE [2], которое могло инициировать эту вставку. Это поле должно быть эквивалентно полю splice_event_id из команды splice_insert, связанной с сообщением метки стандарта ANSI/SCTE [2]. Это сообщение должно быть одинаковым для всех сообщений **Splice_Request**, имеющих отношение к сообщению метки стандарта ANSI/SCTE [2].

Для события, которое не инициировалось сообщением метки стандарта ANSI/SCTE [2], в этом поле будет установлено 0xFFFFFFFF.

PostBlack: Количество тактов системных часов на 90 кГц BVMA, которое будет проигрываться в конце воспроизведения контента вставки. Интервал **PostBlack** не включается в отрезок времени, определенный полем **Duration**. Если поле **PostBlack** не затребовано, то в этом поле будет установлен «0».

AccessType: Указывает на тип доступа, который имеет это соединение. Это целое число от 0 до 9. Число 0 обладает самым низким приоритетом, число 9 обладает самым высоким приоритетом.

OverridePlaying: Когда этот флаг равен «0», поле **Splice_Request** не может переопределить вставку, проигрываемую в настоящий момент. Если этот флаг будет установлен в 1, то поле **Splice_Request** должно переопределить любую в настоящий момент проигрываемую вставку с равным или более низким приоритетом. Проигрывание вставки происходит между точкой входа в вставку и точкой выхода из вставки.

ReturnToPriorChannel: Когда этот флаг равен «0», то при завершении сообщения **Splice_Request** сплайсер не должен возвращаться к основному каналу или к переопределенному каналу ввода. Предполагается, что новый **Splice_Request** будет выпущен перед завершением этой вставки. Если новый **Splice_Request** не будет получен, то сплайсер должен прекратить передачу на этом выходном канале. Когда этот флаг будет равен 1, он не должен возвращаться к предшествующему каналу, если последующее сообщение **Splice_Request** принято и указывает иные параметры вставки.

splice_API_descriptore( ): Это дескриптор, который должен поддерживать синтаксис, определенный в подразделе 6.5 настоящего стандарта.

5.5.2 Сообщение **Splice_Response**

Поле data( ) этого сообщения содержит структуру Splice_Response_Data, основные параметры которой приведены в таблице 7. Сообщение **Splice_Response_Message** может содержать код ошибки. Поле **Splice_Offset** использует сплайсер для информирования сервера о смещении времени поставки контента для этого сообщения. Это поле не влияет на точку входа в основном канале, где произойдет вставка.

Таблица 7 – Структура Splice_Response_Data поля data( ) сообщения вставки **Splice_Response**

Синтаксис	Количество байтов	Тип

Splice_Response_Data {
Splice_Offset
}
	2	tcimsbf

Splice_Offset: В этом поле должен быть установлен «0», если поле не используется для передачи информации о смещении времени. Величина смещения времени оценивается в мс. Отрицательная величина означает, что запрос на вставку контента канала был доставлен ранее, положительная величина – что запрос на вставку контента канала был доставлен позднее.

Поле **Splice_Offset** будет использоваться для устройств сплайсинга, которые выполняют операции вставки «без шва», изменяя задержку распространения основного канала в сплайсере. Когда такое устройство управляет вставкой в отсутствие сообщений метки стандарта ANSI/SCTE [2], сплайсер не имеет возможности выполнять опережение или задержку синхронизации события сервера событий (объявлений) изменением уравнения в преобразовании pts_time ко времени UTC (потому что отсутствуют сообщения метки стандарта ANSI/SCTE [2] и, следовательно, невозможны преобразования и сообщения запроса метки). В таком случае, сплайсер может использовать новое поле

Splice_Offset для ускорения или замедления доставки данных сервером объявлений, для согласования службы выходной синхронизации сплайсера после каждого сообщения **Splice_Request**.

5.5.3 Сообщение **SpliceComplete_Response**

Сообщения **SpliceComplete_Response** отправляются в начале вставки и по окончании вставки. Это правило справедливо и для случая последовательных вводов. Например, при проигрывании двух частей контента возвращаются четыре сообщения

SpliceComplete_Response: одно – в начале первой части контента, одно – после завершения первой части контента, одно – в начале второй части контента и одно – после завершения второй части контента. Если вставка перестала работать, то Код Результата в заголовке должен указывать причину отказа для того, чтобы сервер мог принять необходимые меры. Вход в вставку и выход из нее являются отдельными событиями и должны обрабатываться как отдельные события. Сообщение **SpliceComplete_Response** отправляется непосредственно после отказа любого события вставки, не ожидая окончания вставленного контента.

Поле data( ) этого сообщения содержит структуру SpliceComplete_Response_Data, основные параметры которой приведены в таблице 8.

Таблица 8 – Структура SpliceComplete_Response_Data поля data( ) сообщения вставки **SpliceComplete_Response**

Синтаксис	Количество байтов	Тип
SpliceComplete_Response_Data {		
SessionID	4	uimsbf
SpliceTypeFlag 1	1	uimsbf
if (**SpliceTypeFlag** = 0)		
{		
time( )		
} else		
{		
Bitrate	4	uimsbf
PlayedDuration	4	uimsbf
}		
}		

SessionID: ID Сеанса. Указывает, что используется сообщение Splice_Request.

SpliceTypeFlag: В этом поле устанавливается «0», чтобы указать на вход в вставку, и устанавливается «1», чтобы указать на выход из вставки.

time( ): Содержит время обнаружения сплайсером первого байта потока вставки от сервера. Сервер может использовать поле time( ) для корректировки времени поступления в сплайсер следующего контента канала ввода, когда механизм поставки имеет возможность изменять задержку во времени.

Bitrate (Скорость передачи): Средняя скорость передачи для сеанса. Это поле определяется в битах в секунду (бит/с), включая непроизводительные потери пакетов для этого канала.

PlayedDuration: Число тактов системных часов на 90 кГц на интервале времени фактически воспроизводимых вставок (за исключением кадров перехода или BVMA).

5.6 Сообщения о состоянии (статусе) сплайсера

После завершения инициализации сервер может отправить сообщения **Alive_Request**, чтобы проверить готовность сплайсера к функционированию. Каждое сообщение сплайсера **Alive_Response** для сервера содержит оценку его состояния. Если соединение TCP/IP было не активировано в течении 60 с, сервер должен отправить сообщение **Alive_Request**.

5.6.1 Сообщение **Alive_Request**

Поле data( ) этого сообщения содержит структуру Alive_Request_Data, параметры которой приведены в таблице 9.

Таблица 9 – Структура Alive_Request_Data поля data( ) сообщения о состоянии (статусе) слайсера **Alive_Request**

Синтаксис	Количество байтов	Тип

Alive_Request_Data {
time( )
}		

time( ): Поле содержит текущее значение таймера UTC передающего устройства,

установленное на момент отправки сообщения. Это позволяет обеспечить синхронизацию сплайсера и сервера к началу работы, обеспечивающую достаточно надежную вставку. Синтаксис структуры time( ) приведен в подразделе 6.4 настоящего стандарта.

5.6.2 Сообщение **Alive_Response**

Поле data( ) этого сообщения содержит структуру Alive_Response_Data, параметры которой приведены в таблице 10.

Таблица 10 – Структура Alive_Response_Data поля data( ) сообщения о состоянии (статусе) слайсера **Alive_Response**

Синтаксис	Количество байтов	Тип

Alive_Response_Data {
State
SessionID
time( )
}
	
	4
	4
		
		uimsbf
		uimsbf

State (Состояние): Поле описывает состояние выходного канала в соответствии с таблицей 11.

Таблица 11 – Сообщения состояния **Alive_Response**

Состояние выходного канала	Описание
0x00	Нет выхода
0x01	На основном канале
0x02	На канале ввода

SessionID: Идентификатор воспроизведения вставки в настоящее время. Его применение допускается только для состояния выходного канала, равного 0x02.

time( ): Текущее значение таймера UTC передающего устройства на момент отправки сообщения. Синтаксис структуры time( ) приведен в подразделе 6.4 настоящего стандарта.

5.7 Сообщения расширенных данных

Данная структура определена для передачи от сплайсера на сервер детализированных данных о воспроизведении. После приема **SpliceComplete_Response** расширенные данные могут быть получены с помощью **ExtendedData_Request**. Идентификатор **SessionID**, используемый в этом сообщении, аналогичен **SessionID**, используемому при установке этого сеанса и в **SpliceComplete_Response**.

5.7.1 Сообщение **ExtendedData_Request**

Поле data( ) этого сообщения содержит структуру ExtendedData_Request_Data, параметры которой приведены в таблице 12.

Таблица 12 – Структура ExtendedData_Request_Data поля data( ) сообщения расширенных данных **ExtendedData_Request**

Синтаксис	Количество байтов	Тип

ExtendedData_Request_Data {
SessionID 4 uimsbf
ExtendedDataType 4 uimsbf
}
	
	4
	4
		
		uimsbf
		uimsbf

SessionID: Идентификатор завершенного сеанса.

ExtendedDataType: Запрашиваемый ответ от сплайсера о типе данных на сообщение **ExtendedData_Response**. Его значение может быть установлено в 0xFFFFFFFF, чтобы показать, что этот тип данных по умолчанию должен быть возвращен. Этот стандарт резервирует значения от 0x00000000 до 0x7FFFFFFF для версий будущей стандартизации. Диапазон от 0x80000000 до 0xFFFFFFFE предназначен для использования уникальными поставщиками.

5.7.2 Сообщение **ExtendedData_Response**

Сервер должен использовать поле **MessageSize**, чтобы определить объем данных, которые необходимо считывать через сообщение **ExtendedData_Response**.

Поле data( ) этого сообщения содержит структуру ExtendedData_Response_Data, параметры которой приведены в таблице 13.

Таблица 13 – Структура ExtendedData_Response_Data поля data( ) сообщения расширенных данных **ExtendedData_Response**

Синтаксис	Количество байтов	Тип

ExtendedData_Response_Data {
SessionID
for(i=0; i <n; i ++)
splice_API_descriptor( )
}
	4	uimsbf

SessionID: Идентификатор достоверных данных. Формат дескриптора splice_API_descriptor( ) определен в подразделе 6.5 настоящего стандарта.

5.8 Сообщения прерывания

Сервер может отправить сообщение **Abort_Request** в любое время, это вызовет возврат сплайсера к переопределенному каналу ввода или основному каналу. Сплайсер отправит сообщение **Abort_Response**, подтверждая получение **Abort_Request**. Если сообщение **Abort_Request** вызвало выход из вставки, то сервер отправляет сообщение **SpliceComplete_Response** с Кодом Результата «116»: «Прерванный ввод». Если в выходе из вставки не было необходимости, то сообщение **SpliceComplete_Response** не отправляется. Детализация процесса прерывания последовательных вставок должна быть в соответствии со стандартом ANSI/SCTE [5] (пункт 7.8).

5.9 Сообщение Abort_Request

Поле data( ) этого сообщения содержит структуру Abort_Request_Data, параметры которой приведены в таблице 14.

Таблица 14 – Структура Abort_Request_Data поля data( ) сообщения прерывания **Abort_Request**

Синтаксис	Количество байтов	Тип

Abort_Request_Data {
SessionID
}
	4	uimsbf

SessionID: Поле несет сообщение о прерывании сеанса с **SessionID** и всех последующих сеансов, соединенных через поле **PriorSession**.

5.10 Сообщение Abort_Response

Сообщение **Abort_Response** указывает, что сообщение **Abort_Request** было получено. Это сообщение, при необходимости, может содержать соответствующий Код Результата.

Поле data( ) этого сообщения содержит структуру Abort_Response_Data, параметры которой приведены в таблице 15.

Таблица 15 – Структура Abort_Response_Data поля data( ) сообщения прерывания **Abort_Response**

Синтаксис	Количество байтов	Тип

Abort_Response_Data {
SessionID
}
	4	uimsbf

SessionID: Идентификатор сеанса **SessionID** и всех последующих сеансов, связанных через поле **PriorSession**, которые были прерваны.

5.11 Сообщение TearDownFeed_Request

Сообщение **TearDownFeed_Request** не содержит данных. Это сообщение используется для разъединения каналов, создаваемых сообщениями **Init_Request** с дескриптором create_feed_descriptor( ).

5.12 Сообщение TearDownFeed_Response

Сообщение **TearDownFeed_Response** не содержит данных и указывает, что сообщение **TearDownFeed_Request** было получено и действие произошло. Это сообщение может содержать соответствующий Код Результата.

5.13 Запрос параметров конфигурации

Текущие настройки конфигурации для Соединения API могут быть возвращены. Они включают части информации в **Init_Request**.

5.13.1 Сообщение **GetConfig_Request**

Сообщение **GetConfig_Request** не содержит данных.

5.13.2 Сообщение **GetConfig_Response**

Поле data( ) этого сообщения содержит структуру GetConfig_Response_Data, параметры которой приведены в таблице 16.

Таблица 16 – Структура GetConfig_Response_Data поля data( ) сообщения запроса параметров конфигурации **GetConfig_Response**

Синтаксис	Количество байтов	Тип

GetConfig_Response_Data {
ChannelName
Hardware_Config( )
TS_program_map_section( )
}
	32	Строка

ChannelName: Логическое имя, данное выходному каналу этого соединения.

Hardware_Config( ): Синтаксис этой структуры должен быть в соответствии с подразделом 6.2 настоящего стандарта для синтаксиса структуры Hardware_Config( ).

TS_program_map_section( ): Это полная секция PMT выходного канала в соответствии с стандартом ISO/IEC [3]. Если сплайсер изменяет PMT, он должен сигнализировать об этом изменении серверу в сообщении **General_Response** с Кодом Результата «128». Имя результата в соответствии с таблицей А.1 приложения А настоящего стандарта.

5.14 Сообщение General_Response

Сообщение **General_Response** используется для передачи асинхронной информации между сервером и сплайсером. Поле data( ), связанное с этим сообщением, не используется. В сообщении **General_Response** может быть отправлен любой Код Результата. Это сообщение обычно используется, чтобы сообщить об изменении PMT выходного канала или о недопустимых сообщениях запроса.

6 Дополнительные структуры API

Дополнительные структуры API обеспечивают выбор версии API, согласованной между сервером и сплайсером, и устанавливают:

- конфигурацию и параметры аппаратного интерфейса между сплайсером и сервером;
- структуру и параметры компонентов транспортного потока;
- параметры полей:

1) splice_API_descriptor( );
2) playback_descriptor( );
3) muxpriority_descriptor( );
4) missing_Primary_Channel_action_descriptor( );
5) port_selection_descriptor( );
6) asset_id_descriptor( );
7) create_feed_descriptor( );
8) source_info_descriptor( ).

6.1 Структура Версия

Структура Версия используется для поддержки корректного управления версиями API. Предполагается, что этот API со временем будет развиваться и для того, чтобы учесть возможность такого развития, имя структуры Версия указывается в сообщениях **Init_Request** и **Init_Response**, что позволяет обеспечить поддержку сплайсером и сервером одинаковых версий. Кодирование осуществляется в соответствии с таблицей 17.

Таблица 17 – Кодирование структуры Версия

Синтаксис	Количество байтов	Тип

Version {
Revision_Num
}
	2	uimsbf

Revision_Num: Это поле существует в двух версиях.

Сервер и сплайсер должны установить и проверить это поле, чтобы обеспечить взаимную работоспособность в соответствующей версии.

6.2 Структура Hardware_Config

Эта структура описывает параметры аппаратного интерфейса между сервером и сплайсером. Кодирование осуществляется в соответствии с таблицей 18.

Таблица 18 – Кодирование структуры Hardware_Config

Синтаксис	Количество байтов	Тип
Hardware_Config{		
Length	2	uimsbf
Chassis	2	uimsbf
Card	2	uimsbf
Port	2	uimsbf
Logical_Multiplex_Type	2	uimsbf
Logical_Multiplex( )	Примечание	uimsbf
}		
Примечание – Согласно таблице 19 настоящего стандарта.		

Length (Длина): Длина в байтах структуры Hardware_Config( ), которая следует после этого поля.

Chassis (Шасси): Целое число, указывающее маркировку шасси сплайсера, соединенного с мультиплексом ввода сервера. В случаях, если карта маркирована в алфавитном порядке, преобразование должно быть выполнено в целочисленных значениях (т. е. А соответствует 1; В - 2 и т. д.).

Card (Карта): Целое число, указывающее на карту сплайсера, с которой соединен мультиплекс ввода сервера. В случаях, когда карта маркирована в алфавитном порядке, преобразование выполняется в целочисленных значениях (т. е. А соответствует 1; В - 2 и т. д.).

Port (Порт): Номер аппаратного порта, к которому присоединен мультиплекс ввода сервера.

Logical_Multiplex_Type: Значение поля представлено в таблице 19.

Таблица 19 – Значение поля **Logical_Multiplex_Type**

Тип	Количество байтов	Имя	Описание
0x0000	**0**	Не применяется	Поле **Logical_Multiplex** не должно идентифицировать мультиплекс
0x0001	Переменное	Определяется пользователем	Использование поля **Logical_Multiplex** не определено этой спецификацией и должно быть согласовано между сплайсером и сервером
0x0002	6	MAC адрес	Поле **Logical_Multiplex** содержит MAC адрес
0x0003	6	IPV4 Адрес	Старшие значащие 4 байта поля **Logical_Multiplex** содержат адрес IP мультиплекса, остающиеся 2 байта содержат номер порта IP, где мультиплекс может быть найден
0x0004	18	IPV6 Адрес	Старшие значащие 16 байтов поля **Logical_Multiplex** содержат адрес IP (IPV6) мультиплекса, остающиеся 2 байта содержат номер порта IP, где мультиплекс может быть найден
0x0005	5	ATM Адрес	Поле **Logical_Multiplex** содержит координаты режима асинхронной передачи (ATM), переносящей мультиплекс. Старшие значащие 2 байта логического поля мультиплексирования содержат идентификатор виртуального пути (VPI), следующие 2 байта содержат идентификатор виртуального канала (VCI) тракта. Наименее существенный байт содержит номер уровня адаптации (AAL) ATM
0x0006	Переменное	Адрес IPV4 с поддержкой SPTS	Адрес IPV4 с поддержкой транспортного потока одной программы (SPTS). Этот тип используется VOD и серверами рекламы, где переназначение PID непрактично или нежелательно. В этих случаях необходимо использовать SPTS на порт UDP. Структура этого типа приведена в таблице 20
0x0007	Переменное	Адрес IPV6 с поддержкой SPTS	Адрес IPV6 с поддержкой транспортного потока одной программы (SPTS). Этот тип используется VOD и серверами рекламы, где переназначение PID непрактично или нежелательно. В этих случаях необходимо использовать SPTS на порт UDP. Структура типа 0x0007 приведена в таблице 21
0x0008 – 0xFFFF	Переменное	Зарезервировано	Зарезервировано для будущей стандартизации

Структура поля **Logical_Multiplex_Type** типа 0x0006 приведена в таблице 20.

Таблица 20 – Структура поля **Logical_Multiplex_Type** типа 0x0006

Синтаксис	Количество байтов	Тип
Type 0x0006 structure {		
number_of_destination_ips	1	uimsbf
for (j=0; j <**number_of_destination_ips**; j ++) {		
dest_ip_address	4	uimsbf
}		
number_of_source_ips	1	uimsbf
for (j=0; j <**number_of_source_ips**; j ++) {		
source_ip_address	4	uimsbf
}		
base_port	2	uimsbf
number_of_ports	1	uimsbf
}		

number_of_destination_ips: Определяет количество адресов **dest_ip_address** в последовательности. Диапазон допустимых значений 1 – 32.

dest_ip_address: Адрес IPV4, который сплайсер должен использовать для контента, связанного со вставкой.

number_of_source_ips: Определяет количество адресов **source_ip_address**. Допустимый диапазон значений 0 – 32.

source_ip_address: Адрес IPV4 источника, который сплайсер может использовать в соединении по протоколу IGMP V3 для соответствующего многоадресного **dest_ip_address**.

base_port: Базовый порт UDP, который сплайсер должен использовать для контента, связанного с Splice_Request, имеющей указанное значение time( ). Диапазон значений базового порта UDP должен присваиваться IANA.

number_of_ports: Это поле содержит количество смежных портов для резервирования. Количество смежных портов может изменяться от 1 до 4, включая базовый порт. Разрешенные номера портов определяются в следующем порядке:

- n – номер базового порта;
- n +1 – номер базового порта +1;
- n +2 – номер базового порта +2;
- n +3 – номер базового порта +3.

Все команды Splice_Requests, использующие поле time( ), должны пользоваться базовым портом IPV4 Address:Port, если не применяется дескриптор port_selection_descriptor( ). Первая команда Splice_Request времени, пригодного для рекламы (ВПР), должна использовать поле time( ). Последующие сеансы этого ВПР, которые также используют поле time( ), если не используется дескриптор port_selection_descriptor( ), должны использовать базовый IPV4 Address:Port. При последующих splice_requests, использующих PriorSession вместо поля time( ), будет использоваться базовый IP порт n, IP порт n+1, затем IP порт n+2 и так далее. Для следующего splice_request процедура повторяется с возвращением к базовому порту.

Дескриптор port_selection_descriptor( ) может быть использован в любой команде Splice_Request, если существует аппаратная конфигурация Logical_Multiplex, тип 0x0006, для изменения портов, установленных по умолчанию.

Порт может использоваться с любой допустимой комбинацией одноадресной или многоадресной передачи с адресацией IPV4:Port. Сплайсер выполняет соединение IGMP на многоадресном IP.

Logical_Multiplex: Если порт обрабатывает несколько мультиплексов вставки на одном входе, то это поле позволяет сплайсеру определить, какие мультиплексы от этого сервера необходимо использовать для вставки. Значение поля и формат этого поля определены полем **Logical_Multiplex_Type**. В случае нестандартного определения **Logical_Multiplex** необходимо в **Logical_Multiplex_Type** установить 0x0001 для имени «Определяется пользователем».

Структура поля **Logical_Multiplex_Type** типа 0x0007 приведена в таблице 21.

Таблица 21 – Структура поля **Logical_Multiplex_Type** типа 0x0007

Синтаксис	Количество байтов	Тип

Type 0x0007 structure {
number_of_destination_ips
for (j=0; j <**number_of_destination_ips**; j ++)
{
dest_ip_address
}
number_of_source_ips
for (j=0; j <**number_of_source_ips**; j ++) {
source_ip_address
}
base_port
number_of_ports
}
	
	1
	
	16
	
	1
	
	16
	
	2
	1
		
		uimsbf
		
		uimsbf
		
		uimsbf
		
		uimsbf
		
		uimsbf
		uimsbf

number_of_destination_ips: Определяет количество адресов **dest_ip_address** в последовательности. Диапазон допустимых значений 1 – 32.

dest_ip_address: Адрес IPV6, который сплайсер должен использовать для контента, связанного со вставкой.

number_of_source_ips: Определяет количество адресов **source_ip_address**. Допустимый диапазон значений 0 – 32.

source_ip_address: Адрес IPV6 источника, который сплайсер может использовать в протоколе MLD V2 для соответствующего многоадресного **dest_ip_address**.

base_port: Базовый порт UDP, который сплайсер должен использовать для контента, связанного с Splice_Request, имеющей указанное значение time( ). Диапазон значений базового порта UDP должен быть присвоен IANA.

number_of_ports: В этом байте записано количество смежных портов для резервирования. Количество смежных портов может колебаться от 1 до 4, в их число входит базовый порт. Разрешенные номера портов определяются в следующем порядке:

- n – номер базового порта;
- n+1 – номер базового порта +1;
- n+2 – номер базового порта +2;
- n+3 – номер базового порта +3.

Все команды Splice_Requests, которые используют поле time( ), должны использовать базовый порт IPV6 Address:Port, если не используется port_selection_descriptor( ). Первая команда Splice_Request времени ВПР должна использовать поле time( ). Последующие сеансы того же самого ВПР, которые также используют поле time( ), должны использовать базовый IPV6 Address:Port, если не используется port_selection_descriptor(). При следующем и последующих splice_requests, использующих PriorSession вместо поля time( ), будет использоваться базовый IP порт n, IP порт n+1, затем IP порт n+2 и так далее вплоть до применения разрешенного номера порта. После этого для следующего splice_request процедура повторяется с возвращением к базовому порту.

Дескриптор port_selection_descriptor( ) может быть использован в любой команде Splice_Request, если существует аппаратная конфигурация с Logical_Multiplex, тип 0x0007, для изменения портов, установленных по умолчанию.

Порт может использоваться с любой допустимой комбинацией одноадресной или многоадресной передачи с адресацией IPV6:Port. Сплайсер выполняет соединение MLD на многоадресном IP.

6.3 Структура компонентов транспортного потока splice_elementary_stream( )

Сообщение **Splice_Request** может содержать структуру splice_elementary_stream( ) для каждого компонента транспортного потока. Эта структура используется для описания элементов в программе MPTS. Идентификаторы пакета (PID) идентифицируют части транспортного потока (видео, аудио, данные). Настоящий стандарт не определяет правила отображения совокупности PID аудио/видео/данных на PID выходного канала.

Настоящий стандарт не определяет поведение сплайсера в случаях, когда несколько аудиотреков могут присутствовать или отсутствовать в канале ввода по сравнению с основным каналом. Требования к кодированию структуры splice_elementary_stream( ) приведено в таблице 22.

Таблица 22 – Кодирование структуры splice_elementary_stream( )

Синтаксис	Количество байтов	Тип
splice_elementary_stream {		
Length	1	uimsbf
PID	2	uimsbf
StreamType	2	uimsbf
AvgBitrate	4	uimsbf
MaxBitrate	4	uimsbf
MinBitrate	4	uimsbf
HResolution	2	uimsbf
VResolution	2	uimsbf
for(i=0;I<N;I++)		
descriptor( )		
}		

Length (Длина): Поле определяет полную длину в байтах структуры splice_elementary_stream( ), включая это поле.

PID: Поле определяет номер используемого PID и должно содержать номер PID 13 битов, выровненный по правому краю как целое число на 16 битов (от 0x0000 до 0x1FFF). Наличие PID PCR обязательно.

StreamType: Поле определяет тип потока конкретного PID (аудио, видео и т. д.). Значения PID определяются спецификацией PMT в соответствии со стандартом ISO/IEC [3].

AvgBitrate: Поле определяет скорость передачи потока в бит/с с этим PID, усредненную по всей части контента. Если скорость передачи не известна, в поле установлено 0xFFFFFFFF.

MaxBitrate: Поле определяет максимальную скорость передачи потока этого PID. Если скорость передачи не известна, поле установлено в 0xFFFFFFFF.

MinBitrate: Поле определяет минимальную скорость передачи потока этого PID. Если скорость передачи не известна, поле установлено в 0xFFFFFFFF.

HResolution: Поле определяет ширину видеоизображений потока с данным PID количеством пикселей. Если поток не содержит видеоизображения или если сервер не может предоставить это значение, в поле должно быть установлено 0xFFFF.

VResolution: Поле определяет высоту видеоизображений потока с данным PID количеством пикселей. Если поток не содержит видеоизображения или если сервер не может предоставить это значение, в поле должно быть установлено 0xFFFF.

descriptor( ): Используется любой дескриптор, соответствующий PMT. Для PID потоков аудио дескрипторы языка должны быть в соответствии со стандартом ISO/IEC [3].

6.4 Определение поля time( )

Структура поля time( ) используется для определения времени. Параметры кодирования структуры time( ) приведены в таблице 23.

Таблица 23 – Кодирование структуры time( )

Синтаксис	Количество байтов	Тип
time {		
Seconds	4	uimsbf
MicroSeconds	4	uimsbf
}		

Seconds: Время, прошедшее начиная с 12:00 1 января 1970 UTC, в секундах.

MicroSeconds: Величина смещения поля «секунды» в микросекундах.

6.5 Определение поля splice_API_descriptor( ). Параметры полей

Поле splice_API_descriptor необходимо использоваться в качестве эталона для дополнительных дескрипторов любых сообщений, определенных в этом стандарте. Использование дескрипторов допускается в сообщениях **Splice_Request**, **ExtendedData_Response** и **Init_Request**. В таблице 24 представлен общий формат дескрипторов, используемых в настоящем стандарте.

Таблица 24 – Общий формат дескрипторов, используемых в настоящем стандарте

Синтаксис	Количество байтов	Тип
splice_API_descriptor {		
Splice_Descriptor_Tag	1	uimsbf
Descriptor_Length	1	uimsbf
Splice_API_Identifier	4	uimsbf
for (i=0;i<n;i++)		
Private_Byte	1	uimsbf
}		

Splice_Descriptor_Tag: Поле может принимать значения от 0x00 до 0xFF для того, чтобы обозначить используемый конкретный дескриптор. Значения тега от 0x00 до 0xFF настоящим стандартом зарезервированы для использования в будущем. Поставщик может использовать уникальный дескриптор поставщика **Splice_API_Identifier**, чтобы обеспечить больший диапазон значений тега и более устойчивый метод добавления уникальных дескрипторов поставщика.

Descriptor_Length: Поле определяет длину дескриптора следующего за ним поля в байтах.

Splice_API_Identifier: Идентификатор организации, которая определила этот дескриптор. Для всех дескрипторов в этом документе применяется идентификатор 0x53415049 (ASCII «SAPI»). Этот идентификатор был выбран для исключения конфликтов с дескрипторами любого другого известного идентификатора.

Private_Byte: Эта часть дескриптора выделена полям данных, ее определяют в соответствии с требованиями к дескриптору.

6.5.1 Определения поля **playback_descriptor( )**

Дескриптор playback_descriptor( ) предсталяет собой реализацию формата поля, установленного выше в дескрипторе splice_API_descriptor( ), используемого для аварийного прекращения работы. Дескриптор playback_descriptor( ) предназначен для использования в сообщении **Splice_Request**.

Критерием аварийного прекращения работы является скорость воспроизведения, определяемая как скорость передачи выходного канала, усредненная за время 1 с. Величину сдвига смещения окна усреднения рекомендуется устанавливать не более 1 с.

Кодирование поля playback_descriptor( ) выполняется в соответствии с таблицей 25.

Таблица 25 – Кодирование поля playback_descriptor( )

Синтаксис	Количество байтов	Тип
playback_descriptor {		
Splice_Descriptor_Tag	1	uimsbf
Descriptor_Length	1	uimsbf
Splice_API_Identifier	4	uimsbf
BitrateRule	1	uimsbf
MinPlaybackRate	4	uimsbf
}		

Splice_Descriptor_Tag: В поле устанавливается 0x01.

Descriptor_Length: В поле устанавливается 0x09.

Splice_API_Identifier: В поле устанавливается 0x53415049, ASCII "SAPI".

BitrateRule: Флаг, обозначает правила интерпретации содержания поля **MinPlaybackRate** в соответствии с таблицей 26.

Таблица 26 – Значения флага **BitrateRule** для правил интерпретации содержания поля **MinPlaybackRate**

Значения **BitrateRule**	Описание
0x00	**MinPlaybackRate** игнорируется
0x01	Немедленно вернуть Код Результата «127» (имя результата в соответствии с таблицей А.1 приложения А настоящего стандарта), используя **General_Response**, если скорость воспроизведения падает ниже значения **MinPlaybackRate**, но не прерывать (вставку)
0x02	Аварийное прекращение работы, если скорость воспроизведения падает ниже значения **MinPlaybackRate**
0x03	Отмена сеанса до вставки, если сплайсер решает, что необходимое значение **MinPlaybackRate** не будет достигнуто. Сплайсер отправит **SpliceComplete_Response** или **General_Response** с Кодом Результата «127». Имя результата в соответствии с таблицей А.1 приложения А настоящего стандарта

MinPlaybackRate: Минимальная совокупная скорость передачи выходного канала, усредненная за 1 с для интервала вставки, в котором это может иметь значение относительно инициированного **BitrateRule**. Установка значения, равного 0x00, означает, что условие минимальной скорости не выполняется.

6.5.2 Определение поля **muxpriority_descriptor( )**

Дескриптор muxpriority_descriptor( ) является реализацией формата поля, установленного выше в splice_API_descriptor( ). Дескриптор muxpriority_descriptor( ) предназначен для использования в сообщении **Splice_Request**.

Кодирование поля muxpriority_descriptor( ) выполняется в соответствии с таблицей 27.

Таблица 27 – Кодирование поля muxpriority_descriptor( )

Синтаксис	Количество байтов	Тип
muxpriority_descriptor {		
Splice_Descriptor_Tag	1	uimsbf
Descriptor_Length	1	uimsbf
Splice_API_Identifier	4	uimsbf
MuxPriorityValue	1	uimsbf
}		

Splice_Descriptor_Tag: В поле устанавливается 0x02.

Descriptor_Length: В поле устанавливается 0x05.

Splice_API_Identifier: В поле устанавливается 0x53415049, ASCII «SAPI».

MuxPriorityValue: Значение поля изменяется в диапазоне от 1 до 10 (Значение 1 соответствует самому низкому уровню приоритета, значение 5 соответствует среднему уровню приоритета, значение 10 соответствует самому высокому приоритету) и изменяет сохраненное в сплайсере значение приоритета **MuxPriorityValue** основного канала. При значении поля **MuxPriorityValue**, равном 5, приоритет выходного основного канала не будет изменяться. При значении **MuxPriorityValue** меньше 5 приоритет основного канала будет уменьшен. При значении **MuxPriorityValue** больше чем 5, приоритет основного канал будет увеличен.

Использование **MuxPriorityValue** не будет гарантировать прерывание контента с конкретным уровнем качества. Фактическая эффективность применения **MuxPriorityValue** зависит от конфигурации всех вставок мультиплекса, от того насколько сплайсер должен понизить полную скорость передачи мультиплекса в любой момент времени, а также от работоспособности сплайсера в целом.

6.5.3 Определение поля **missing_Primary_Channel_action_descriptor( )**

Дескриптор missing_Primary_Channel_action_descriptor( ) является реализацией формата поля, установленного выше в splice_API_descriptor( ). Дескриптор предназначен для использования в сообщении **Init_Request**.

В том случае, если основной канал по какой-либо причине во время вставки завершился, декодер должен вывести на экран стоп-кадр последнего вставленного кадра в конце вставки. Этот деск-

риптор позволяет сплайсеру предписать вставку поля BVMA, чтобы очистить буфер декодера, если основной канал больше не существует. В таблице 28 установлены возможные варианты реализации поля missing_Primary_Channel_action_descriptor( ).

Таблица 28 – Возможные варианты реализации поля missing_Primary_Channel_action_descriptor( )

Синтаксис	Количество байтов	Тип
missing_Primary_Channel_action_descriptor {		
Splice_Descriptor_Tag	1	uimsbf
Descriptor_Length	1	uimsbf
Splice_Api_Identifier	4	uimsbf
MissingPrimaryChannelAction	1	uimsbf
}		

Splice_Descriptor_Tag: В поле устанавливается 0x03.
Descriptor_Length: В поле устанавливается 0x05.
Splice_API_Identifier: В поле устанавливается 0x53415049, ASCII «SAPI».
MissingPrimaryChannelAction: У этого параметра могу быть три возможных значения: 0, 1, 2. При значении 0 изменений не происходит. Значение 1 означает вставку одного I-кадра черного и одного кадра «аудио тишины». Значение 2 означает продолжение передачи I-кадров черного и «аудио тишины» до появления сигнала основного канала.

6.5.4 Определения поля **port_selection_descriptor( )**

Дескриптор port_selection_descriptor( ) является реализацией формата дескриптора splice_API_descriptor( ), определенного выше в подразделе 6.5. Дескриптор предназначен для использования в сообщении **Splice_Request** для аппаратных конфигураций логических типов мультиплекса 0x0006 или 0x0007. Если в процессе выполнения последовательности вставок сервер отправляет дескриптор port_selection_descriptor( ), сервер должен продолжать передавать дескриптор port_selection_descriptor( ) до появления следующего базового сообщения **Splice_Request**.

Дескриптор port_selection_descriptor( ) может быть использован для изменения операции выбора портов «по умолчанию» или для выбора новой динамически устанавливаемой комбинации портов IPV4 или IPV6 Address:Port.

Сплайсер должен динамически установить порт назначения, если **ps_ip_address** не был определен в аппаратной конфигурации. Если **ps_ip_address** будет передан в форме многоадресной рассылки, то сплайсер должен запустить соединение IGMP или MLD не более чем через 400 мс после прибытия сообщения **Splice_Request**. Задержка для установления многоадресной группы должна составить не менее 2 с.

Кодирование поля port_selection_descriptor( ) для IPV4 выполняется в соответствии с таблицей 29.

Таблица 29 – Кодирование поля port_selection_descriptor( ) для IPV4

Синтаксис	Количество байтов	Тип
port_selection_descriptor {		
Splice_Descriptor_Tag	1	uimsbf
Descriptor_Length	1	uimsbf
Splice_API_Identifier	4	uimsbf
ps_ip_address	4	uimsbf
ps_port	2	uimsbf
ps_number_of_source_ip	1	uimsbf
for(j=0; j <**ps_number_of_source_ip**; j ++) {		
ps_source_ip_address	4	uimsbf
}		
}		

Splice_Descriptor_Tag: В поле устанавливается 0x04.
Descriptor_Length: Определяет в байтах переменную длину поля, следующего после этого дес-

криптора.

Splice_API_Identifier: В поле устанавливается 0x53415049, ASCII «SAPI».

ps_ip_address: Поле определяет интернет-адрес протокола IPV4, который сплайсер должен использовать для контента, связанного со вставкой. Если это сочетание address:port отличается от адреса в таблице Logical_Mux_Type 0x0006, то это поле нужно считать динамическим запросом установки порта.

ps_port: Поле определяет номер порта UDP, который сплайсер должен использовать для контента, связанного со вставкой. Этот номер порта должен переопределить автоматический метод выбора порта Logical_Multiplex_Type 0x0006.

ps_number_of_source_ip: Поле определяет количество адресов **ps_source_ip_address**. Диапазон допустимых значений 0 – 32.

ps_source_ip_address: Поле определяет адрес IPV4 источника, который сплайсер должен использовать, чтобы присоединиться по протоколу IGMP V3 для соответствующей многоадресной передачи **ps_ip_address**.

Кодирование поля port_selection_descriptor( ) для IPV6 выполняется в соответствии с таблицей 30.

Таблица 30 – Кодирование поля port_selection_descriptor( ) для IPV6

Синтаксис	Количество байтов	Тип
port_selection_descriptor {		
Splice_Descriptor_Tag	1	uimsbf
Descriptor_Length	1	uimsbf
Splice_API_Identifier	4	uimsbf
ps_ip_address	16	uimsbf
ps_port	2	uimsbf
ps_number_of_source_ip	1	uimsbf
for (j=0; j <**ps_number_of_source_ip**; j ++) {		
ps_source_ip_address	16	uimsbf
}		
}		

Splice_Descriptor_Tag: В поле устанавливается 0x05.

Descriptor_Length: Дескриптор определяет переменную длину поля, следующего после этого поля, в байтах.

Splice_API_Identifier: В поле устанавливается 0x53415049, ASCII «SAPI».

ps_ip_address: Поле определяет интернет-адрес протокола IPV6, который сплайсер должен использовать для контента, связанного со вставкой. Если это сочетание address:port отличается от адреса в таблице Logical_Mux_Type 0x0007, то это поле нужно считать динамическим запросом установки порта.

ps_port: Поле определяет номер порта UDP, который сплайсер использует для контента, связанного с вставкой. Этот номер порта должен переопределять порт, выбранный автоматическим методом Logical_Multiplex_Type 0x0007.

ps_number_of_source_ip: Порт определяет количество адресов **ps_source_ip_address**. Диапазон допустимых значений 0 – 32.

ps_source_ip_address: Порт определяет адрес IPV6 источника, который сплайсер должен использовать, чтобы присоединиться по протоколу IGMP V3 или MLD для соответствующей многоадресной передачи **ps_ip_address**.

6.5.5 Определение поля **asset_id_descriptor( )**

Дескриптор asset_id_descriptor( ) является реализацией формата дескриптора splice_API_descriptor( ), который будет использоваться только в сообщении **Splice_Request**. Этот дескриптор применяется при воспроизведении MLD (массива данных), выполняемого сплайсером, и может использоваться также для идентификации этого объекта. Предлагается для рекламного контента вместо полного имени использовать идентификатор рекламной вставки или другой идентификатор. Для контента VOD длинной формы может потребоваться ввод полного имени объекта. Формат имени при управлении сервером и сплайсером настоящий стандарт не определяет.

Кодирование поля asset_id_descriptor( ) выполняется в соответствии с таблицей 31.

Таблица 31 – Кодирование поля asset_id_descriptor( )

Синтаксис	Количество байтов	Тип
asset_id_descriptor {		
Splice_Descriptor_Tag	1	uimsbf
Descriptor_Length	1	uimsbf
Splice_Api_Identifier	4	uimsbf
Asset_Upid_Type	1	uimsbf
Asset_Upid_Length	1	uimsbf
Asset_Upid( )	Примечание	uimsbf
}		
Примечание – Согласно стандарту ANSI/SCTE [2] (таблица 8-6).		

Splice_Descriptor_Tag: В поле устанавливается 0x06.

Descriptor_Length: Устанавливает переменную длину поля, следующего после этого поля, в байтах.

Splice_API_Identifier: В поле устанавливается 0x53415049, ASCII «SAPI».

Asset_Upid_Type: В поле устанавливается значение, соответствующее значению поля segmentation_upid_type стандарта ANSI/SCTE [2] (таблица 8-6). Если значение поля segmentation_upid_type в стандарте ANSI/SCTE [2] (таблица 8-6) равно 0x09 (ADI), то требования относительно этого ADI связаны со значением, указанным в стандарте ANSI/SCTE [2] (8.3.3.1), и являются частью настоящего стандарта.

Asset_Upid_Length: В поле устанавливается длина в байтах структуры **Asset_Upid**. Максимальное значение этого поля равно 245.

Asset_Upid( ): Определяет длину и идентификацию этого поля в соответствии с segmentation_upid_type стандарта ANSI/SCTE [2] (таблица 8-6). Содержание этой структуры и ее длина определяют поля Asset_Upid_Type и Asset_Upid_Length. Пример: тип 0x06 для V-ISAN длиной 12 байтов. Это поле тогда содержало бы идентификатор V-ISAN для контента, к которому относится этот дескриптор.

6.5.6 Определение поля **create_feed_descriptor( )**

Дескриптор create_feed_descriptor( ) является реализацией формата дескриптора splice_API_descriptor( ), который будет использоваться только в сообщении **Init_Request**. Этот дескриптор дает сплайсеру информацию, необходимую для формирования вывода SPTS.

Процесс использования этого дескриптора должен быть следующим:

1) сервер объявления (рекламы) должен открыть новое соединение TCP со сплайсером создаваемого канала;

2) сервер объявления должен отправить сообщение **Init_Request** с этим дескриптором через созданное соединение TCP;

3) поле **ChannelName** в сообщении **Init_Request** должно отразить имя недавно создаваемого канала;

4) сплайсер должен создать канал согласно сообщению **Init_Request**. Вновь созданный канал должен быть идентичным каналу, обозначенному **OriginalChannelName**. Идентификация данных должна относиться ко всем значениям PID, типам, дескрипторам, PID PCR, номерам программы в PMT недавно созданного канала и PAT;

5) сплайсер должен отправить сообщение **Init_Response** только после того, как новый канал будет создан.

Кодирование поля create_feed_descriptor( ) выполняется в соответствии с таблицей 32.

Таблица 32 – Кодирование поля create_feed_descriptor( )

Синтаксис	Количество байтов	Тип
create_feed_descriptor {		
Splice_Descriptor_Tag	1	uimsbf
Descriptor_Length	1	uimsbf
Splice_Api_Identifier	4	uimsbf
OriginalChannelName	32	uimsbf
Create_Feed_Descriptor_Type	1	uimsbf
if (**Create_Feed_Descriptor_Type** == 0) {		
IPV4_Dest_Address	4	uimsbf
Destination_Port	2	uimsbf
}		
else if (**Create_Feed_Descriptor_Type** == 1) {		
IPV6_Dest_Address	16	uimsbf
Destination_Port	2	uimsbf
}		
}		

Splice_Descriptor_Tag: В поле устанавливается 0x07.

Descriptor_Length: Устанавливается переменная длина поля, следующего после этого дескриптора в байтах.

Splice_API_Identifier: В поле устанавливается 0x53415049, ASCII «SAPI».

OriginalChannelName: В поле устанавливается логическое имя выходного канала, используемого в качестве шаблона для вновь созданного канала. Представляет собой строку, завершенную 0.

Create_Feed_Descriptor_Type: В поле устанавливается «0», чтобы указать, что используется адрес IPV4, и устанавливается «1», чтобы указать, что используется адрес IPV6.

IPV4_Dest_Address: В поле устанавливается IP-адрес приемника для создаваемой выходной службы. Он имеет формат IPV4.

IPV6_Dest_Address: В поле устанавливается IP-адрес приемника для создаваемой выходной службы. Он имеет формат IPV6.

Destination_Port: В поле устанавливается номер порта приемника UDP для создаваемой выходной службы.

6.5.7 Определение поля **source_info_descriptor( )**

Дескриптор source_info_descriptor( ) является реализацией дескриптора splice_API_descriptor( ), который может использоваться в сообщении **Cue_Request**. Этот дескриптор дает серверу информацию о соответствии параметров канала вставки и основного канала по параметрам типа, разрешения и частоте кадров. Этот дескриптор должен использоваться в случае, если основной канал будет содержать только одно видео.

Кодирование поля source_info_descriptor( ) выполняется в соответствии с таблицей 33.

Таблица 33 – Кодирование поля source_info_descriptor( )

Синтаксис	Количество байтов	Тип
source_info_descriptor {		
Splice_Descriptor_Tag	1	uimsbf
Descriptor_Length	1	uimsbf
Splice_Api_Identifier	4	uimsbf
StreamType	1	uimsbf
HResolution	2	uimsbf
VResolution	2	uimsbf
frame_rate_code	1	uimsbf
progressive_sequence	1	uimsbf
}		

Splice_Descriptor_Tag: В поле устанавливается 0x08.

Descriptor_Length: В поле устанавливается 0x0A.

Splice_API_Identifier: В поле устанавливается 0x53415049, ASCII «SAPI».

StreamType: В поле устанавливается число, соответствующее спецификации PMT, в соответствии с ISO/IEC [3].

HResolution: В поле устанавливается ширина видеоизображений в количестве пикселей.

VResolution: В поле устанавливается высота видеоизображений в количестве пикселей.

frame_rate_code: В поле устанавливается параметр, кодированный значением frame_rate_code из стандарта ISO/IEC [6] (таблица 6-4) для видео MPEG-2 и видео AVC, что соответствует кодированной частоте кадров, указанной в таблице 34.

Таблица 34 – **frame_rate_code** для **frame_rate_value**

Frame_rate_code	frame_rate_value
0000	Запрещено применять
0011	25 Гц
0110	50 Гц
1001	Зарезервировано
1111	Зарезервировано

7 Синхронизация сервера и сплайсера

Синхронизация сервера и сплайсера обеспечивает передачу контента вставок и сообщений между сервером и сплайсером в точно установленные моменты времени. Задержка передачи сообщений по протоколу TCP/IP непредсказуема, она зависит от других машин, работающих в сети. При условии синхронизации машин параметры времени могут передаваться между ними (в условиях нормальных (рабочих) сетевых задержек) и обеспечивать высокую точность вставок. Возможный метод обеспечения синхронизации между сервером и сплайсером использует Сетевой Протокол Системного Времени (NTP). Для этого серверы и сплайсеры для сохранения синхронизации должны быть клиентами NTP. Сетевой общий сервер NTP может использоваться в качестве хост-сервера.

Система синхронизации времени должна обеспечивать синхронизацию между сплайсером и сервером с ошибкой не более ± 15 мс. Система может использовать сообщения **Alive_Request** / **Alive_Response** для контроля состояния синхронизации между этими двумя устройствами и информирования оператора о потере синхронизации.

Поток битов основного канала подвергается различным задержкам, которые могут принимать значения от миллисекунд до секунд. Эти задержки не влияют на точность сообщения метки, встроенного в основной канал. Сообщение метки использует PCR, в котором указано корректное время вставки, что обеспечивает сохранение исходной точности вставки относительно контента.Сервер, обеспечивающий контент канала вставки, «знает» только время UTC и время окна вставки, с которым вставка была запрограммирована относительно времени UTC. Время передачи потока контента вставки определяет сплайсер, чтобы указать серверу точный момент начала передачи потока контента.

Сплайсер принимает поток битов программы со всеми задержками этого потока. Сплайсер может, используя PCR, связать его со временем часов (UTC) и затем отправить сообщение серверу, который определит точное время UTC, в которое нужно начать передачу потока. Канал вставки от сервера теперь достигает сплайсера, точно синхронизированным с основным каналом, что может обеспечить выполнение идеальной вставки. Любые дополнительные задержки, которые происходят в сплайсере, не существенны, так как входные потоки битов синхронизированы.

В том случае, если не все части оборудования системы могут использовать время UTC, то для синхронизации сплайсера и сервера эти части в качестве базового времени могут использовать время GPS. В этом случае все реализации системы вставок должны предусматривать преобразование базового времени суток GPS к времени UTC и последовательно учитывать или игнорировать смещение GPS относительно UTC на интервалах вычисленного времени.

8 Синхронизация в системе

8.1 Поток сигналов вставки

Рисунки 3 и 4 передают подробности использования и упорядочивания различных сообщений, обеспечиваемых API настоящего стандарта. Возможности фактического использования сообщений API не ограничено этими примерами.

Сценарий вставки единственного события показан на рисунке 3:

1) CRM: Сплайсер отправляет серверу сообщение **Cue_Request** вскоре после сообщений ANSI/SCTE [2], принятых сплайсером в основном канале.

2) CRespM: Сервер отвечает сообщением **Cue_Response**.

3) SRM: По крайней мере, за 3 с до намеченного времени вставки сервер передает сообщение **Splice_Request** сплайсеру.

Рисунок 3 – Сценарий вставки единственного события

4) SRespM: Сплайсер отвечает сообщением **Splice_Response**.

5) SCM: Вскоре после того, как вставка началась, сплайсер отправляет сообщение **SpliceComplete_Response**.

6) Сплайсер после завершения вставки отправляет другое сообщение **SpliceComplete_Response**.

Сценарий вставки нескольких событий показан на рисунке 4:

1) CRM: Сплайсер отправляет серверу сообщение **Cue_Request** вскоре после сообщений ANSI/SCTE [2], полученных сплайсером в основном канале (и не показанных на рисунке 4).

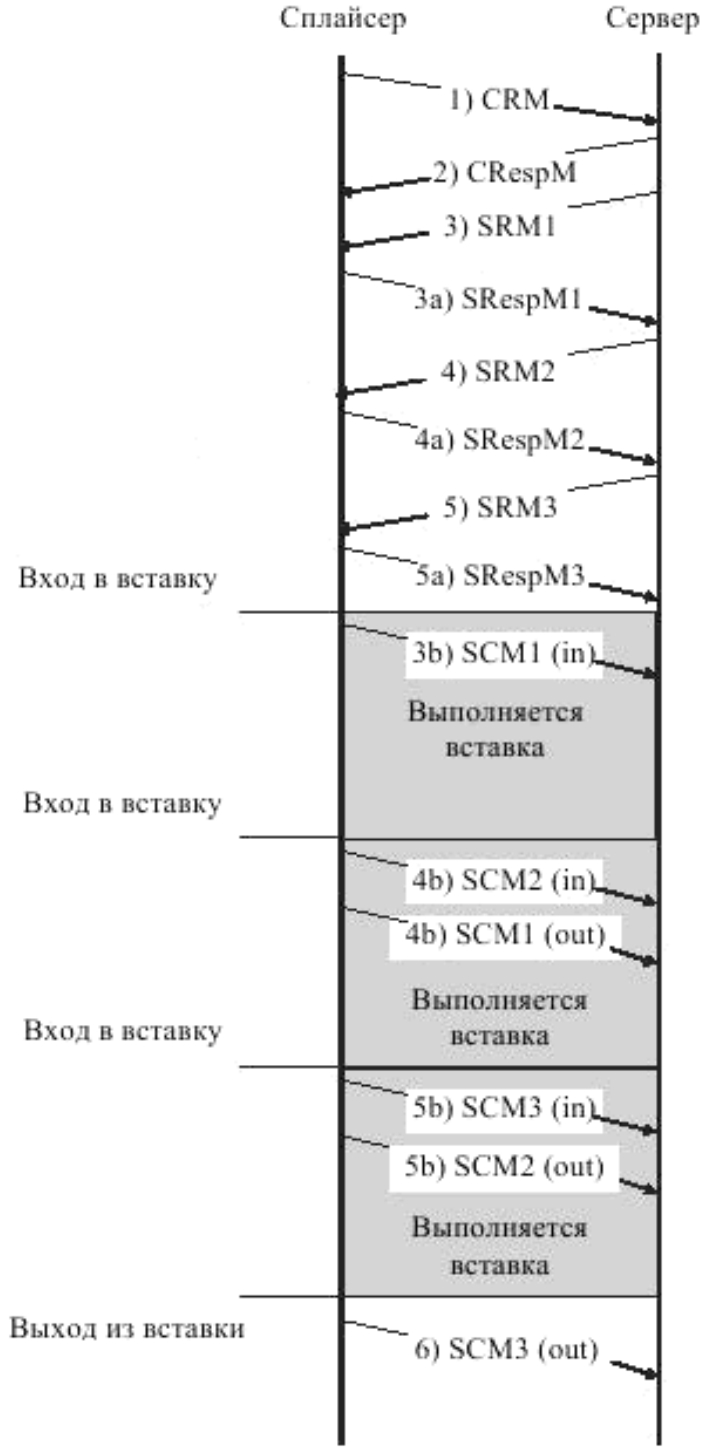

Рисунок 4 – Сценарий вставки нескольких событий

2) CRespM: Сервер отвечает сообщением **Cue_Response**.

3) SRM1: По крайней мере, за 3 с до намеченного времени вставки сервер передает первое сообщение **Splice_Request** к сплайсеру.

a) SRespM1: Splicer ответит первым сообщением **Splice_Response**.

b) SCM1 (in): Как только первая вставка выполнена, сплайсер отправит серверу первое сообщение **SpliceComplete_Response**.

4) SRM2: Для следующей вставки сплайсер отправит, а сплайсер получит второе сообщение **Splice_Request**, не менее чем за 3 с до намеченной вставки.

a) SRespM2: Сплайсер отвечает вторым сообщением **Splice_Response**.

b) SCM1 (out); SCM2 (in): При завершении первой вставки сплайсер отправит два сообщения **SpliceComplete_Response**, обозначающие конец первой вставки и начало второй вставки.

5) SRM3: Для третьей вставки сплайсер должен получить следующее сообщение **Splice_Request** не менее чем за 3 с до намеченной третьей вставки.

a) SRespM3: Сплайсер отвечает вторым сообщением Splice_Response.

b) SCM2 (out); SCM3 (in): При завершении первой вставки сплайсер отправит два сообщения **SpliceComplete_Response**, обозначающие конец второй вставки и начало третьей вставки.

6) SCM3 (out): Вскоре после того, как третья вставка завершится, сплайсер отправляет другое сообщение **SpliceComplete_Response**.

8.2 Временная шкала инициирования вставки

На рисунке 5 показан пример синхронизации продвижения событий до начала вставки программы или рекламы. Временные соотношения в действительности могут отличаться от показанных на этом рисунке. Показанный интервал времени может использоваться при обсуждении арбитража приоритетов, представленного в подразделе 6.2 настоящего стандарта. На рисунке 5 также показана связь выполняемых операций с сообщением метки стандарта ANSI/SCTE [2].

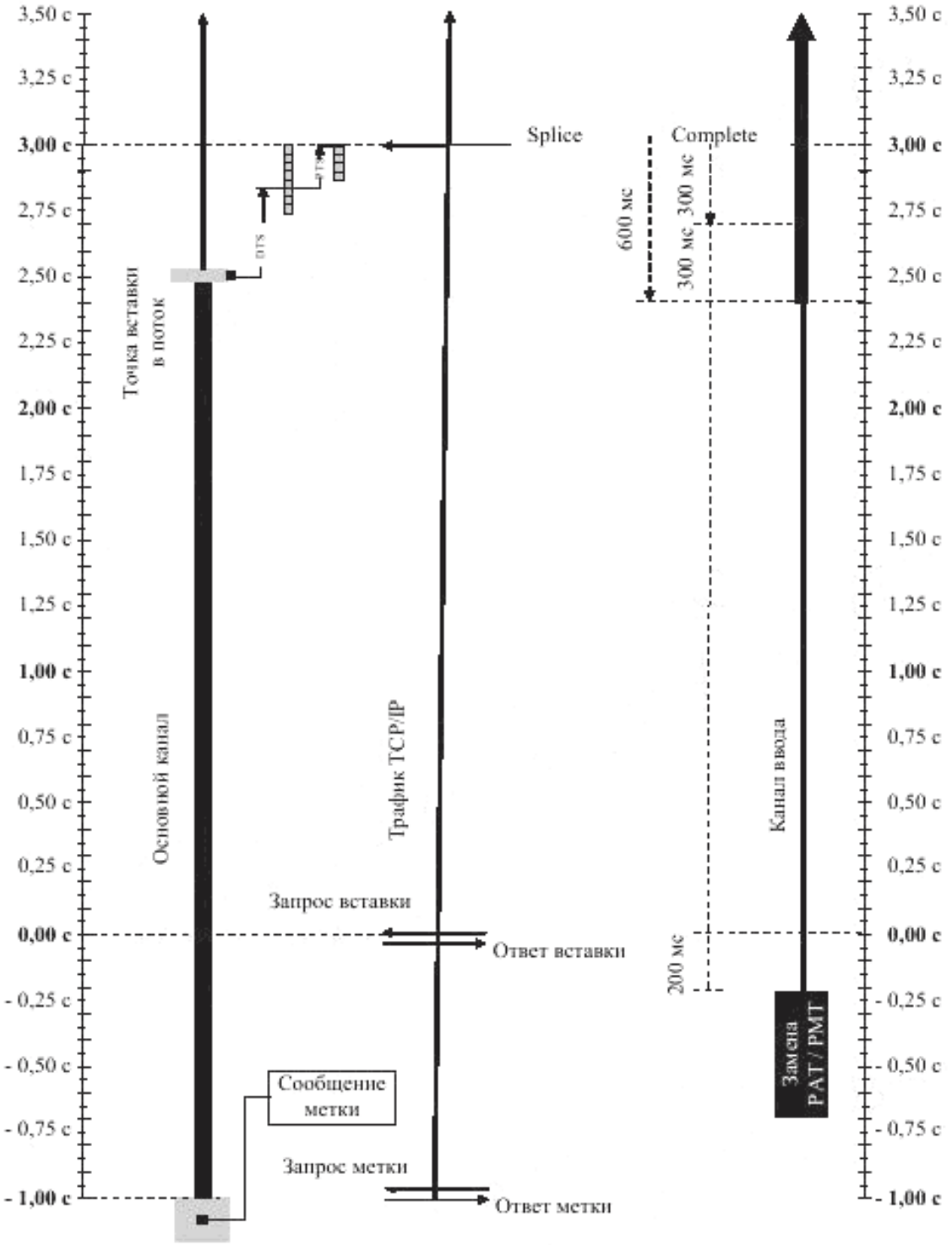

Рисунок 5 – Временная шкала инициирования вставки рекламы

На рисунке 5 жирные черные линии обозначают поток информации MPEG на линии основного канала и на линии канала ввода. Тонкая черная линия указывает на то, что информация MPEG или не передается в этот момент, или не вставлена, чтобы появиться в выходном канале.

Приложение А
(обязательное)

Коды Результата

Таблица А.1 – Коды Результата

Код Результата	Имя результата	Описание	Сообщение ответа
100	Успешный ответ		Все
101	Неизвестный отказ		Все
102	Недопустимая версия	Сервер и сплайсер используют разные версии этого API	**Init_Response**
103	Доступ запрещен	Возможная проблема лицензии	**Init_Response** **Splice_Response**
104	Недопустимые или неизвестные **ChannelName**	Возможная ошибка конфигурации	**Init_Response**
105	Недопустимое физическое соединение	Возможная ошибка конфигурации	**Init_Response**
106	Конфигурация не найдена	Сплайсер не способен определить конфигурацию этого соединения	**GetConfig_ Response**
107	Недопустимая конфигурация	Один или более параметров в конфигурации для этого соединения недопустимы	**GetConfig_Response**
108	Отказ вставки	Отказ по неизвестной причине	**SpliceComplete_Response**
109	Коллизия вставки	Более высокий или такой же приоритет на вставку уже установлен	**Splice_Response** **SpliceComplete_Response**
110	Каналы ввода не найдены	Эта ошибка должна быть возвращена, если канал ввода отсутствует	**SpliceComplete_Response**
111	Основной канал не найден	Этот результат должен быть возвращен, если основной канал отсутствует на интервале времени от точки входа до точки выхода	**SpliceComplete _Response**
112	Сообщение **Splice_Request** пришло слишком поздно	Сообщение **Splice_Request** было получено с опозданием (более чем за 3 с). Сплайсер должен инициировать вставку	**Splice_Response**
113	Точки вставки не были найдены	Сплайсер не в состоянии найти допустимые точки для вставки в основной канал	**SpliceComplete_Response**
114	Очередь вставки заполнена	Слишком много не выполненных сообщений **Splice_Request**	**Splice_Response**

Продолжение таблицы А.1

Код Результата	Имя результата	Описание	Сообщение ответа
115	Предположение о качестве проигрывания (воспроизведения) сеанса	Сплайсер обнаружил несоответствия видео или аудио, которые, возможно, влияли на воспроизведение	**SpliceComplete_Response**
116	Прерывание вставки	Сообщение **Abort_Request** вызвало выход из вставки	**SpliceComplete_Response**
117	Недопустимое сообщение метки	Сплайсер или сервер не могли проанализировать сообщение метки	**General_Response** **Cue_Response**
118	Устройство вставки не существует	**SplicerName** не было найдено	**Init_Response**
119	Сообщение **Init_Request** отвергнуто	Сплайсер не разрешает серверу соединяться	**Init_Response**
120	Неизвестный **MessageID**	Используйте сообщение Splicing_API_Message, чтобы отправить ответ на запрашивающую сторону. Ответьте на неизвестный **MessageID**	Все
121	Недопустимый **SessionID**	Сплайсеру неизвестен указанный **SessionID**	**Splice_Response** **Abort_Response** **ExtendedData_Response**
122	Сеанс не завершен	Сплайсер не смог проиграть полную продолжительность. Это может быть случай, когда сервер не предоставил достаточный по продолжительности контент	**SpliceComplete_Response**
123	Недопустимый запрос data( )	Сплайсер или сервер не смогли успешно проанализировать поле в запросе. Недопустимая позиция поля возвращена в поле **Result_Extension** сообщения Splicing_API_Message.	Все
124	Дескриптор не реализован	Сплайсер не понимает или не реализует дескриптор	Ответы на все сообщения, разрешенные дескрипторам.
125	Переопределение канала	Этот Код Результата указывает, что в настоящее время проигрывается вставка, которая была переопределена с изменением состояния точки выхода или была вновь введена с состоянием точки входа	**SpliceComplete_Response**

Продолжение таблицы А.1

Код Результата	Имя результата	Описание	Сообщение ответа
126	Вставка канала началась преждевременно	Это сообщение ошибки может быть выдано, если вставка канала была начата преждевременно и сплайсер был не в состоянии правильно определить начало вставки потока	**SpliceComplete_Response**
127	Скорость воспроизведения ниже порога	Для детализации смотри playback_descriptor( ) согласно 6.5.1 настоящего стандарта	**SpliceComplete_Response**
128	Изменение PMT	Это сообщение используется, чтобы указать серверу, что PMT для этого основного канала изменилась	**General_Response**
129	Сообщение недопустимых размеров	Длина сообщения была не в соответствии с настоящим стандартом	Все
130	Сообщение с недопустимым синтаксисом	Поля не соответствуют параметрам настоящего стандарта	Все
131	Ошибка коллизии порта	Сплайсер не в состоянии использовать указанное запрашиваемое сочетание IP:port	Init_Response
132	Поврежденная вставка – EAS активна	Это сообщение ошибки должно быть возвращено, если активна система аварийного оповещения (EAS)	**SpliceComplete_Response** **Splice_Response**
133	Компоненты вставки не найдены	Не найдены один или более компонентов вводимого потока	**SpliceComplete_Response**
134	Ресурсы не доступны	В соответствии со стандартом ANSI/SCTE [5] (приложение А)	Init_Response

Окончание таблицы А.1

Код Результата	Имя результата	Описание	Сообщение ответа
135	Несоответствие компонент	Этот код может быть возвращен сплайсеру, когда по запросу Splice_Request он не в состоянии определить компоненты канала ввода, соответствующие компонентам основного канала. Примеры: 1 Основной канал содержит компонент аудио AC-3, но Splice_Request определяет только компонент аудио MPEG-2. 2 Основной канал содержит видео компонент AVC, но Splice_Request определяет только видео компонент MPEG-2	Splice_Response
Примечание – Все коды результата могут использоваться в сообщении **General_Response**			

Библиография

[1]	IETF RFC 3810	Multicast Listener Discovery Version 2 (MLDv2) for IPV6
[2]	ANSI/SCTE 35 2007	Digital Program Insertion Cueing Message for Cable
[3]	ISO/IEC 13818-1:2000-12-01	Information Technology - Generic Coding of Moving Pictures and Associated Audio Information: Systems
[4]	IETF RFC 3376	Internet Group Management Protocol, Version 3
[5]	ANSI/SCTE 30 2009	Digital Program Insertion Splicing Application Program Interface
[6]	ISO/IEC 13818-2:2000	Information Technology – Generic Coding of Moving Pictures and Associated Audio Information: Video

УДК 621.397:681.327.8:006.354 ОКС 33.170 ОКП 657400

Ключевые слова: телевидение вещательное цифровое, транспортные потоки, вставка (сплайсинг), контент, прикладной программный интерфейс API

Подписано в печать 01.04.2014. Формат 60x84 1/8.
Усл. печ. л. 5,12 Тираж 31 экз. Зак. 1915.
Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

ФГУП «СТАНДАРТИНФОРМ»,

123995 Москва, Гранатный пер., 4.
www.gostinfo.ru info@gostinfo.ru