Internet Engineering Task Force (IETF)
Request for Comments: 5810
Category: Standards Track
ISSN: 2070-1721

A. Doria, Ed.
Lulea University of Technology
J. Hadi Salim, Ed.
Znyx
R. Haas, Ed.
IBM
H. Khosravi, Ed.
Intel
W. Wang, Ed.
L. Dong
Zhejiang Gongshang University
R. Gopal
Nokia
J. Halpern
March 2010

Спецификация протокола ForCES

Forwarding and Control Element Separation (ForCES) Protocol Specification

Аннотация

В этом документе описан протокол разделения элементов управления и пересылки ForCES[1]. Протокол ForCES служит для коммуникаций между элементами управления CE[2] и элементами пересылки FE[3] в элементах сети ForCES NE[4]. Данная спецификация соответствует требованиям к протоколу ForCES, определенным в RFC 3654. В дополнение к протоколу ForCES данная спецификация определяет также требования к уровню транспортного отображения TML[5].

Статус документа

Документ относится к категории Internet Standards Track.

Документ является результатом работы IETF[6] и представляет согласованный взгляд сообщества IETF. Документ прошёл открытое обсуждение и был одобрен для публикации IESG[7]. Дополнительную информацию о стандартах Internet можно найти в разделе 2 в RFC 5741.

Информацию о текущем статусе документа, ошибках и способах обратной связи можно найти по ссылке http://www.rfc-editor.org/info/rfc5810.

Оглавление

1. Введение....3
2. Термины и соглашения....3
2.1. Уровни требований....3
2.2. Обозначения....3
2.3. Целые числа....4
3. Определения....4
4. Обзор....5
4.1. Схема протокола....5
4.1.1. Уровень протокола (PL)....6
4.1.2. Уровень транспортного отображения (TML)....6
4.1.3. Интерфейс FEM/CEM....6
4.2. Фазы протокола ForCES....7
4.2.1. До объединения....7
4.2.2. После объединения....8
4.2.2.1. Этап создания ассоциации....8

[1]Forwarding and Control Element Separation.
[2]Control Element.
[3]Forwarding Element.
[4]ForCES Network Element
[5]Transport Mapping Layer.
[6]Internet Engineering Task Force.
[7]Internet Engineering Steering Group.

4.2.2.2. Этап наличия ассоциации....8
4.2.2.3. Этап потери ассоциации....8
4.3. Механизмы протокола....8
4.3.1. Транзакции, неделимость, выполнение и отклики....9
4.3.1.1. Выполнение операций....9
4.3.1.1.1. Все или ничего....9
4.3.1.1.2. Продолжать при возникновении отказа....9
4.3.1.1.3. Выполнять до первого отказа....9
4.3.1.2. Транзакция и неделимость....9
4.3.1.2.1. Определение транзакции....9
4.3.1.2.2. Протокол транзакций....9
4.3.1.2.3. Восстановление....10
4.3.1.2.4. Пример сообщений транзакции....10
4.3.2. Масштабируемость....11
4.3.2.1. Серия операций....11
4.3.2.2. Конвейер команд....11
4.3.3. Механизм Heartbeat....11
4.3.4. Объект FE и протокольные LFB....11
4.4. Протокольные сценарии....12
4.4.1. Состояние создания ассоциации....12
4.4.2. Состояние созданной ассоциации и установившееся состояние....12
5. Требования к TML....13
5.1. Параметризация TML....14
6. Инкапсуляция сообщений....15
6.1. Общий заголовок....15
6.2. Структуры TLV....16
6.2.1. Вложенные TLV....17
6.2.2. Область действия типа (T) в TLV....17
6.3. ILV....17
6.4. Вопросы инкапсуляции....17
6.4.1. Пути....17
6.4.2. Ключи....18
6.4.3. TLV данных....18
6.4.4. Адресация объектов LFB....18
7. Устройство протокола....18
7.1. Кодирование....20
7.1.1. Правила упаковки данных....20
7.1.2. Флаги пути....20
7.1.3. Связь операционных флагов с глобальными флагами сообщения....20
7.1.4. Выбор пути к содержимому....20
7.1.5. LFBselect-TLV....20
7.1.6. OPER-TLV....21
7.1.7. RESULT TLV....21
7.1.8. DATA TLV....22
7.1.9. Взаимосвязи SET и GET....22
7.2. Визуализация протокольного кодирования....22
7.3. Основные LFB протокола ForCES....24
7.3.1. FE Protocol LFB....25
7.3.1.1. Возможности FE Protocol....25
7.3.1.1.1. SupportableVersions....25
7.3.1.1.2. Компоненты FE Protocol....25
7.3.1.1.2.1. CurrentRunningVersion....25
7.3.1.1.2.2. FEID....25
7.3.1.1.2.3. MulticastFEIDs....25
7.3.1.1.2.4. CEHBPolicy....25
7.3.1.1.2.5. CEHDI....25
7.3.1.1.2.6. FEHBPolicy....25
7.3.1.1.2.7. FEHI....25
7.3.1.1.2.8. CEID....25
7.3.1.1.2.9. LastCEID....25
7.3.1.1.2.10. BackupCE....26
7.3.1.1.2.11. CEFailoverPolicy....26
7.3.1.1.2.12. CEFTI....26
7.3.1.1.2.13. FERestartPolicy....26
7.3.2. FE Object LFB....26
7.4. Семантика направления передачи сообщений....26
7.5. Сообщения для ассоциаций....26
7.5.1. Сообщение Association Setup....26
7.5.2. Сообщение Association Setup Response....27
7.5.3. Сообщение Association Teardown....27
7.6. Конфигурационные сообщения....28
7.6.1. Сообщение Config....28
7.6.2. Сообщение Config Response....28
7.7. Запросы....29
7.7.1. Сообщение Query....29
7.7.2. Сообщение Query Response....30
7.8. Уведомление о событии....30

7.9. Сообщение PacketRedirect....31
7.10. Сообщение Heartbeat....33
8. Поддержка высокого уровня доступности....33
8.1. Связь с FE Protocol....33
8.2. Ответственность за HA....34
9. Вопросы безопасности....34
9.1. Без защиты....35
9.1.1. Проверка подлинности конечных точек....35
9.1.2. Проверка подлинности сообщений....35
9.2. ForCES PL и защитные услуги TML....35
9.2.1. Проверка подлинности конечных точек....35
9.2.2. Проверка подлинности сообщений....35
9.2.3. Конфиденциальность....35
10. Благодарности....35
11. Литература....35
11.1. Нормативные документы....35
11.2. Дополнительная литература....36
Приложение A. Взаимодействие с IANA....36
A.1. Пространство типов сообщений....36
A.2. Выбор операции....36
A.3. Флаги заголовка....37
A.4. Типы TLV....37
A.5. Коды результата в RESULT-TLV....37
A.6. Отклик Association Setup....38
A.7. Сообщение Association Teardown....38
Приложение B. Схема LFB протокола ForCES....38
B.1. Возможности....41
B.2. Компоненты....41
Приложение C. Примеры кодирования данных....41
Приложение D. Варианты использования....43

1. Введение

Разделение элементов управления и пересылки (ForCES) определяет архитектурную модель и сопутствующие протоколы для стандартизации обмена информацией между уровнем правления (control plane) и уровнем пересылки (forwarding plane) в элементах сети ForCES (ForCES NE). В RFC 3654 определены требования к ForCES, а в RFC 3746 - модель ForCES. Хотя в архитектуре ForCES могу применяться многочисленные протоколы, в этом документе термины «протокол ForCES» и «протокол» относятся к только протоколу, служащему для стандартизации информационного обмена между элементами управления CE и элементами пересылки FE.

Модель ForCES FE [RFC5812] представляет формальный способ определения логических функциональных блоков FE LFB с использованием XML. Конфигурационные компоненты, возможности и события LFB определяются при формальном создании LFB. Логические блоки LFB внутри FE согласованно контролируются стандартизованным способом по протоколу ForCES.

Этот документ определяет спецификацию протокола ForCES, который работает в режиме «ведущий-ведомый» (master-slave), где FE играют роль ведомых, а CE - ведущего. Протокол включает команды для транспортировки конфигурационных данных LFB, организации связей (association), данных состояния, уведомлений о событиях и т. п.

В разделе 3 приведён глоссарий используемых в документе терминов.

Раздел 4 содержит обзор протокола с обсуждением модели и описанием протокольного уровня PL[1], уровня транспортного отображения TML и механизмов протокола ForCES. В параграфе 4.4 рассмотрено несколько протокольных сценариев и приведено описание обмена сообщениями.

Документ не определяет TML, но в разделе 5 описаны услуги, которые TML **должен** обеспечивать (требования к TML).

Протокол ForCES определяет общий заголовок для всех сообщений. Этот заголовок описан в параграфе 6.1, а протокольные сообщения определены в разделе 7.

В разделе 8 описана поддержка протоколом механизмов высокой доступности, включая избыточность (redundancy) и отказоустойчивость (fail over).

В разделе 9 определены механизмы защиты, обеспечиваемые уровнями PL и TML.

2. Термины и соглашения

2.1. Уровни требований

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не следует** (SHALL NOT), **следует** (SHOULD), **не нужно** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **не рекомендуется** (NOT RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе интерпретируются в соответствии с [RFC2119].

2.2. Обозначения

В таблицах 1 и 2 используются приведённые ниже обозначения.

(value)+ - один или несколько экземпляров value;

(value)* - необязательные экземпляры value.

[1]Protocol Layer.

2.3. Целые числа

Все целые числа представляются беззнаковыми двоичными значениями подходящего размера.

3. Определения

В этом документе используются термины, определённые для требований к ForCES в [RFC3654] и модели ForCES в [RFC3746]. Определения для удобства повторены.

Addressable Entity (AE) - адресуемый объект (элемент)
Физическое сетевое устройство, которое непосредственно адресуется в данной технологии соединения. Например, в сетях IP - это устройства, к которым можно обращаться по адресу IP, а в матрице коммутации (switch fabric) - это устройства, к которым можно обращаться по номеру порта в матрице.

Control Element (CE) - элемент управления
Логический объект, который реализует протокол ForCES и инструктирует один или множество FE по части обработки пакетов. Функциональность CE включает исполнение протоколов управления и сигнализации.

CE Manager (CEM) - менеджер элементов управления
Логический объект, отвечающий за генерацию базовых задач управления CE. Используется, в частности, на этапе до объединения (pre-association phase) для определения FE, с которым CE следует взаимодействовать. Этот процесс называется обнаружением FE и может включать изучение менеджером CE возможностей доступных FE.

Data Path - путь (передачи) данных
Концептуальный путь, по которому пакеты проходят через уровень пересылки внутри FE.

Forwarding Element (FE) - элемент пересылки
Логический элемент, реализующий протокол ForCES. Элементы FE используют базовое оборудование для обработки каждого пакета и управляются (контролируются) одним или множеством CE по протоколу ForCES.

FE Model - модель элемента пересылки
Модель, описывающая функции логической обработки в FE. Модель FE определяется с использованием логических функциональных блоков (LFB).

FE Manager (FEM) - менеджер FE
Логический элемент, которые работает в фазе до объединения и отвечает за определение CE, с которыми элементу FE следует взаимодействовать. Этот процесс называется обнаружением CE и может включать определение менеджером FE возможностей доступных CE. Менеджер FE может применять все, что угодно от статической конфигурации до протокола фазы до объединения (см. ниже) для определения используемого CE. Однако протокол фазы до объединения выходит за рамки документа. Будучи логическим устройством менеджер FE может быть физически объединён с любыми логическими элементами, упомянутыми в этом разделе, типа FE.

ForCES Network Element (NE) - элемент сети ForCES
Объект, состоящий из одного или множества CE и одного или множества FE. Для внешних наблюдателей NE представляется единой точкой управления и скрывает свою внутреннюю структуру от внешних наблюдателей.

High Touch Capability - работа с верхними уровнями
Этот термин обозначает возможности некоторых устройство пересылки (forwarder) выполнять операции над содержимым или заголовками пакетов, на основе данных, не входящих в заголовок IP. Примерами таких возможностей служат правила качества обслуживания (QoS), виртуальные частные сети, межсетевое экранирование, распознавание содержимого L7.

Inter-FE Topology - внешняя топология FE
См. FE Topology.

Intra-FE Topology - внутренняя топология FE
См. LFB Topology.

LFB (Logical Function Block) - логический функциональный блок
Базовый блок, с которым работает протокол ForCES. LFB - это чётко определённый, логически разделяемый функциональный блок, который размещается в FE и управляется CE по протоколу ForCES. LFB может размещаться в пути данных FE и обрабатывать потоки, а может быть чистым объектом управления и настройки FE, с которым работает CE. Отметим, что LFB является функционально точной абстракцией возможностей обработки FE, а не точным аппаратным представлением реализации FE.

FE Topology - топология FE
Представление соединений между множеством FE в одном NE. Иногда это называют внешней топологией FE, чтобы отличать от внутренней топологии FE (т. е. топологии LFB).

LFB Class and LFB Instance - класс и экземпляр LFB
LFB делятся на классы. Экземпляр LFB представляет существование класса (или типа) LFB. В FE может присутствовать множество экземпляров одного класса (или типа) LFB. Класс LFB представляется идентификатором класса и экземпляром LFB, представленным идентификатором экземпляра. В результате идентификатор класса и связанный с ним идентификатор экземпляра однозначно указывают наличие LFB.

LFB Meta Data - метаданные LFB
Метаданные служат для передачи информации о состоянии на уровне отдельного пакета из одного блока LFB в другой, но без передаче через сеть. Модель FE определяет для таких данных идентификацию, обработку, и восприятие (потребление) другими LFB. Модель определяет функциональность, но не задаёт представления метаданных внутри реализации.

LFB Component - компонента LFB
Рабочие параметры LFB, которые должны быть видимы для элементов CE и концептуально заданы моделью FE как компоненты LFB. Эти компоненты включают, например, флаги, аргументы отдельных параметров, комплексные аргументы и таблицы, которые элемент CE может читать и/или записывать с помощью протокола ForCES (см. ниже).

LFB Topology - топология LFB
Представление логических связей между экземплярами LFB и их размещения в пути данных внутри одного элемента FE. Иногда используется термин «внутренняя топология FE», но не следует путать её с топологией соединений между FE (inter-FE topology).

Pre-association Phase - фаза до объединения
Интервал времени, в течение которого менеджер FE (см. ниже) и менеджер CE (см. ниже) определяют каким FE и CE следует быть частью одного сетевого элемента. Все разделение PFE и PCE происходит на этом этапе.

Post-association Phase - фаза после объединения

Интервал времени, в течение которого FE знает управляющие им устройства CE и наоборот, включая интервал, в течение которого CE и FE организуют связи между собой.

ForCES Protocol - протокол ForCES

Хотя архитектура ForCES может включать множество протоколов, термины «протокол» и «протокол ForCES» относятся к опорным точкам (интерфейсам) Fp в модели ForCES [RFC3746]. Этот протокол не применяется для коммуникаций между парами элементов CE, парами элементов FE или менеджерами FE и CE. Протокол ForCES в основном работает в режиме «ведущий-ведомый», где элементы FE являются ведомыми, а CE - ведущими. Этот документ задаёт спецификацию протокола ForCES.

ForCES Protocol Layer (ForCES PL) - уровень протокола ForCES

Уровень архитектуры протокола ForCES, который определяет протокольные сообщения ForCES, схему передачи состояний протокола и архитектуру самого протокола ForCES (включая требования ForCES TML, как показано ниже). Спецификация ForCES PL также задана в этом документе.

ForCES Protocol Transport Mapping Layer (ForCES TML) - уровень транспортного отображения ForCES

Уровень архитектуры протокола ForCES, которые использует возможности существующих транспортных протоколов для решения задач доставки протокольных сообщений, таких как отображения этих сообщений на различные транспортные среды (TCP, IP, ATM, Ethernet и пр.), обеспечения гарантий доставки, групповой адресации, сохранения порядка и т. п. Спецификации ForCES TML задаются в отдельных документах ForCES для каждого типа TML.

4. Обзор

Читателям рекомендуется обратиться к документу [RFC3746] и, в частности, к его разделам 3 и 4, где приведён обзор архитектуры и описано протокол ForCES в рамках этой архитектуры. Содержимое упомянутого документа и данного раздела могут перекрываться в целях обеспечения ясности. Этот документ является полномочным для протокола, а [RFC3746] - для архитектуры.

4.1. Схема протокола

Рисунок 1, заимствованный из описания архитектуры, показывает NE с двумя элементами CE и двумя FE.

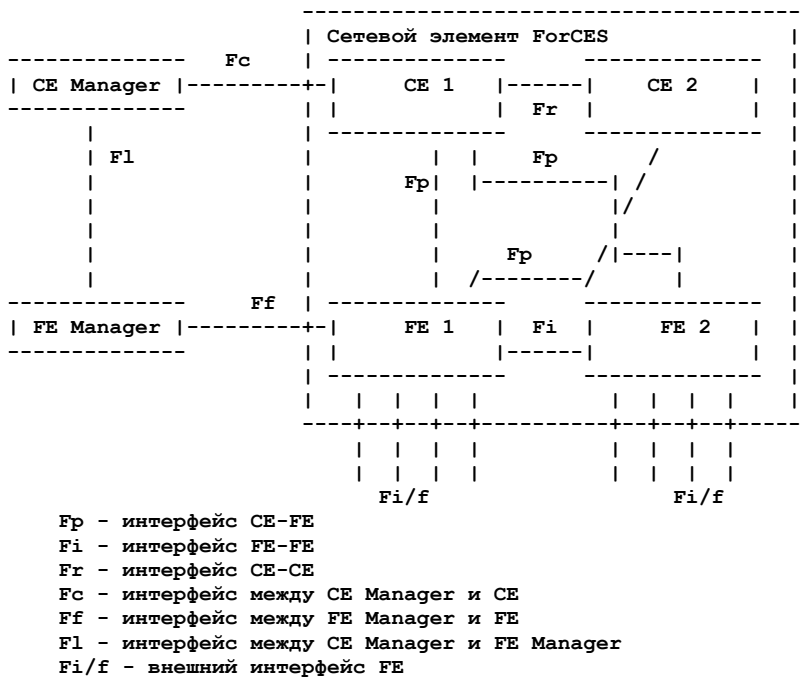

Рисунок 1. Архитектура ForCES.

Областью действия протокола ForCES являются опорные точки (интерфейсы) Fp. Опорные точки настройки конфигурации элементов протокола Fc и Ff также играют роль при загрузке протокола ForCES. Настройка конфигурации элементов протокола (опорные точки Fc, Ff и Fl в [RFC3746]) выходит за рамки протокола ForCES, но будет упоминаться в этом документе при рассмотрении FEM и CEM, поскольку это является частью фазы протокола pre-association (до объединения).

На рисунке 2 представлена детализация интерфейса Fp с помощью примера, включающего сетевой элемент MPLS с поддержкой QoS.

Интерфейс ForCES, показанный на рисунке 2, состоит из двух частей - PL и TML, как показано на рисунке 3.

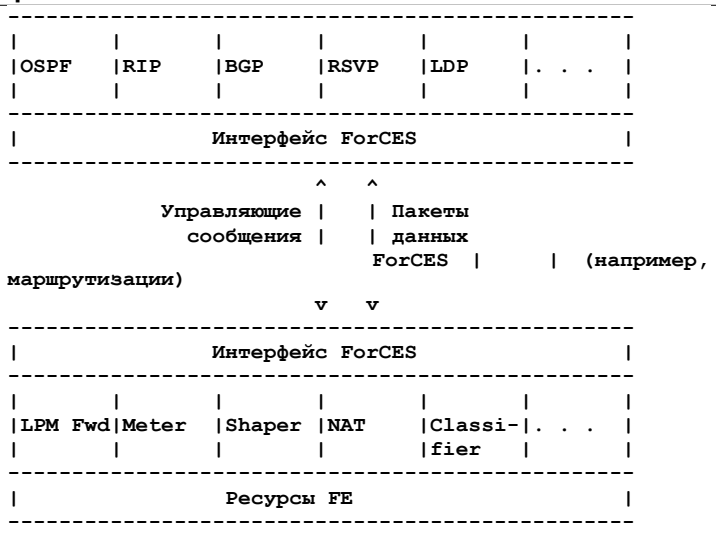

Рисунок 2. Примеры функций CE и FE.

Уровень PL фактически является протоколом ForCES. Его семантика и схема сообщений определяются в этом документе. Уровень TML требуется для соединения между двумя ForCES PL, как показано на рисунке3. TML выходит за рамки этого документа, но входит в ForCES. Данный документ определяет требования PL, которые должны выполняться уровнем TML.

Уровни PL и TML стандартизуются IETF. Несмотря на определение одного PL, предполагается стандартизация различных TML. Для обеспечения взаимодействия между TML в элементах CE и FE предполагается их соответствие одному определению.

При передаче PL доставляет свои сообщения уровню TML. Локальный уровень TML доставляет сообщение TML получателя, а тот передаёт их уровню PL получателя.

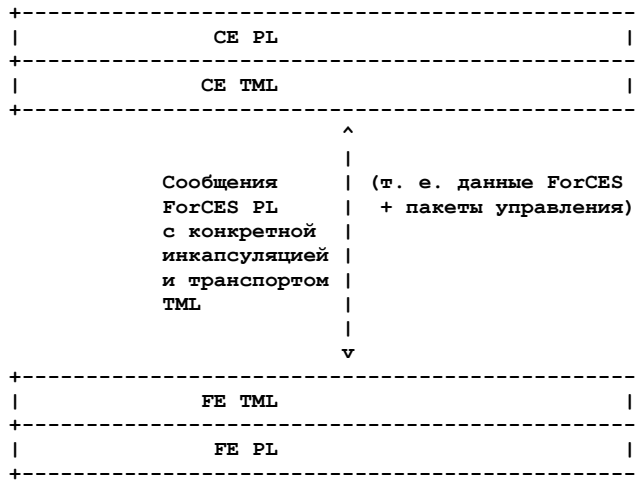

Рисунок 3. Интерфейс ForCES.

4.1.1. Уровень протокола (PL)

PL является общим для всех реализации ForCES и стандартизуется IETF в соответствии с данным документом. PL отвечает за привязку (ассоциацию FE или CE к сетевому элементу NE, а также за разрыв таких ассоциаций (отсоединение). FE использует уровень PL для передачи событий, на которые задана подписка, в CE PL, а также откликов на различные запросы состояния со стороны CE PL. Элемент CE настраивает рабочие параметры FE и связанных с ним LFB, используя уровень PL. Кроме того, CE может передавать разные запросы элементам FE для их активации или отключения, изменения параметров HA[1], подписки на определённые события и т. п. Более подробное описание этого приведено в разделе 7.

4.1.2. Уровень транспортного отображения (TML)

Уровень TML доставляет сообщения PL. С TML связаны вопросы обеспечения на транспортном уровне гарантированной доставки, контроля перегрузок, групповой адресации, соблюдения порядка и т. п. Предполагается стандартизация более одного уровня TML. Разные TML могут варьировать свои реализации с учётом возможностей базовой среды и транспорта. Однако стандартизация каждого TML будет гарантировать взаимодействие в случае поддержки конечными точками одного TML. Все реализации протокольного уровня ForCES **должны** обеспечивать переносимость между разными TML, поскольку все TML **должны** иметь на верхнем уровне одинаковую семантику, определённую в этом документе.

4.1.3. Интерфейс FEM/CEM

Компоненты FEM и CEM нужны на этапе организации и настройки как PL так TML, но не входят в протокол ForCES. Лучше всего рассматривать их как настройку и параметризацию PL и TML до того, как эти уровни станут активными (или начнут работать). В простейшем случае элемент FE или CE считывает конфигурацию из файла. В RFC 3746

[1]High Availability - высокий уровень доступности.

приведено более подробное описание возможного использования FEM и CEM. Фаза pre-association, где могут применяться CEM и FEM, кратко описана в параграфе 4.2.1.

Примером того, что может быть настроено с помощью FEM/CEM, являются параметры уровня TML:

a. организация соединения TML (например, используемый адрес IP, транспортный режим и т. п.);

b. ID для FE (FEID) или CE (CEID), который также будет вводиться в фазе pre-association;

c. параметры защиты типа ключей и т. п.;

d. параметры организации соединения.

Примером параметров организации соединения могут быть:

- простые параметры - передача до трёх сообщений каждую секунду;
- сложные параметры - передача до 4 сообщения с экспоненциально растущим интервалом.

4.2. Фазы протокола ForCES

Протокол ForCES, применительно к элементам NE, включает две фазы - pre-association, где происходит настройка, инициализация и загрузка уровней TML и PL, а также post-association, где протокол ForCES манипулирует параметрами FE.

После организации связи (ассоциации) FE может пересылать пакеты в зависимости от конфигурации его конкретных LFB. Элемент FE, связанный с CE, будет продолжать пересылку пакетов, пока не получит сообщения Association Teardown или не потеряет связь. Несвязанный FE может продолжать передачу пакетов при поддержке им свойства высокой доступности. Дополнительные детали приведены в разделе 8 и не рассматриваются здесь до чёткого разъяснения основ.

Смена состояний FE контролируется с помощью компоненты FE Object LFB FEState, определённой в параграфе 5.1 [RFC5812] и описанной в параграфе 7.3.2 данного документа.

FE инициализируется в состоянии FEState OperDisable. Когда элемент FE готов для обработки пакетов в пути данных, он переводит себя в состояние OperEnable.

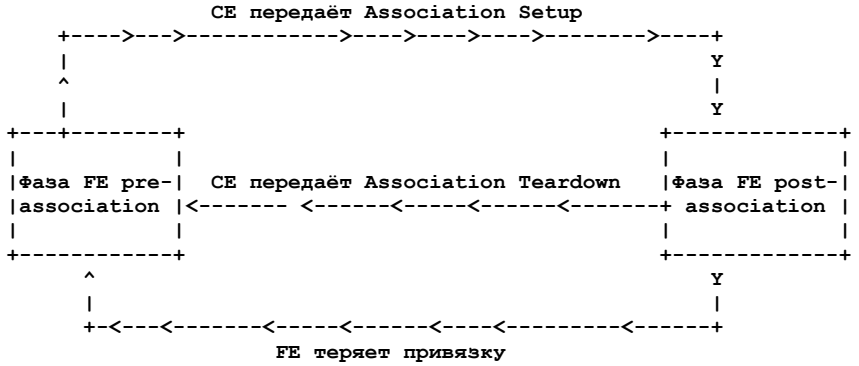

Рисунок 4. Фазы протокола FE.

Элемент CE может принять решение о приостановке работы FE, находящегося в состоянии OperEnable, с помощью установки FEState AdminDisable. Элемент FE будет сохранять состояние AdminDisable до явного указания со стороны CE о переходе в состояние OperEnable.

Когда элемент FE теряет связь с CE, он может перейти в фазу, если это задано политикой. Для корректного перехода FE в состояние AdminDisable элемент **должен** прекратить пересылку пакетов и это может влиять на множество LFBS. Решение этой задачи выходит за рамки данной спецификации.

4.2.1. До объединения

Интерфейс ForCES настраивается в фазе pre-association. В простом варианте конфигурация задаётся статически и обычно считывается из файла. Все параметры ассоциации становятся известными по завершении фазы pre-association. Для получения конфигурационных параметров могут использоваться протоколы типа DHCP вместо считывания данных из конфигурационного файла. Отметим, что это все равно считается статической фазой pre-association. Динамическая настройка конфигурации может выполняться с использованием интерфейсов Fc, Ff и Fl (см. [RFC3746]). Производители могут применять фирменные протоколы обнаружения для передачи параметров. По сути, здесь приводятся лишь рекомендации, а детали отдаются на откуп разработчикам.

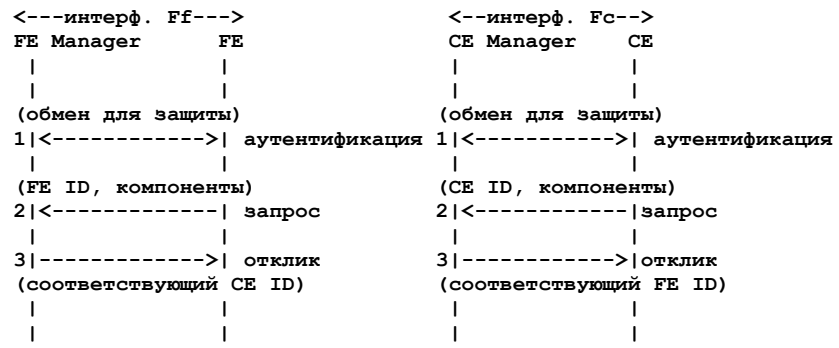

Рисунок 5. Пример обмена сообщениями через интерфейсы Ff и Fc.

Рисунки из документа с описанием схемы показывают пример фазы до объединения.

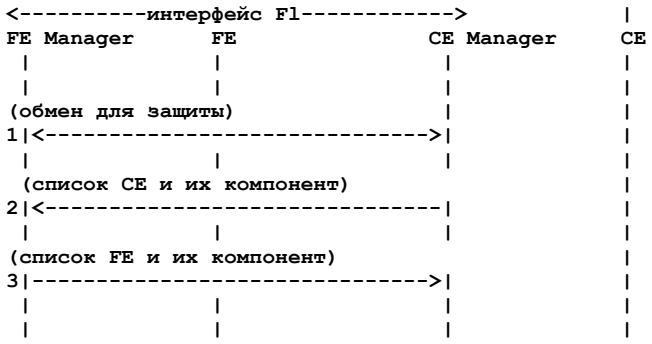

Рисунок 6. Пример обмена сообщениями через интерфейс Fl.

До перехода в фазу объединения FEM будет организовывать контакт с компонентой CEM. Инициализация интерфейса ForCES завершена, проверка подлинности и определение возможностей также могут быть завершены. Элементы FE и CE будут иметь всю информацию, нужную при соединении для настройки, учёта, идентификации и проверки подлинности. По окончании этого этапа обе стороны будут иметь все требуемые параметры протокола типа таймеров и пр. Интерфейс Fl может продолжать работу в фазе объединения и может применяться для отсоединения элементов FE или CE. Конкретные взаимодействия CEM и FEM, которые относятся к фазе pre-association, выходят за рамки спецификации и не будут далее рассматриваться. Читателю рекомендуется обратиться к [RFC3746].

4.2.2. После объединения

В этой фазе компоненты FE и CE взаимодействуют по протоколу ForCES (PL на основе TML), как определено в этом документе. Здесь имеются три субфазы:

- этап создания ассоциации;
- этап наличия ассоциации (связи);
- этап потери ассоциации.

4.2.2.1. Этап создания ассоциации

FE пытается присоединиться к элементу NE, попытка может быть принята или отвергнута. После получения доступа в NE происходит обмен информацией о возможностях по запросу от CE к FE. После получения CE сведений о возможностях FE элемент CE может предложить начальную конфигурацию (возможно для восстановления состояния), а также запросить некоторые компоненты LFB или самого FE.

Более подробная информация приведена в параграфе 4.4.

При успешном завершении этого этапа FE присоединяется к NE и переходит на этап наличия связи (Established Stage).

4.2.2.2. Этап наличия ассоциации

На этом этапе FE постоянно получает обновления и запросы. FE может также передавать асинхронные уведомления о событиях элементу CE и синхронные уведомления heartbeat, если это задано. Этот этап продолжается по потери связности или прерывания ассоциации по инициативе CE или FE.

Поведение протокола на этом этапе подробно описано в параграфе 4.4.

4.2.2.3. Этап потери ассоциации

На этом этапе CE или FE (или оба элемента) сообщает другой стороне об потере ассоциации. Это может произойти по инициативе любой из сторон (административные причины) или в результате потери связности или иных технических причин.

Определена основа LFB, называемая FEPO[1] (см. Приложение B и параграф 7.3.1). FEPO задаёт различные таймеры, которые могут применяться в месте с чувствительным к трафику механизмом heartbeat (параграф 4.3.3) для обнаружения потери связности.

Потеря связности между TML не указывает потерю ассоциации между соответствующими уровнями PL. Если TML не может восстановить утраченный транспорт до завершения отсчёта заданных FEPO таймеров, связанных с FE, ассоциация между соответствующими уровнями PL будет потеряна.

FEPO определяет несколько правил, которые могут быть заданы для управления поведением в случае потери ассоциации. Заданная FEPO политика CE при отказах (см. раздел 8, параграфы 7.3.1, 4.3.3 и Приложение B) определяет, что будет происходить при потере ассоциации.

Для определённой в этом документе версии протокола элемент FE при повторном создании ассоциации **должен** отбрасывать любое состояние, как непригодное, и получать новое состояние. Такой подход продиктован соображениями простоты (в отличие от эффективности).

4.3. Механизмы протокола

Семантика, раскрываемая пользователям протокола через заголовки PL включает возможности транзакций и их неделимость, двухэтапное представления, группировку и распараллеливание операций, высокую доступность и восстановление при отказах, а также конвейеры команд.

Флаги EM (режим выполнения - Execution Mode), AT (неделимая транзакция - Atomic Transaction) и TP (фаза транзакции - Transaction Phase), определённые для общего заголовка (параграф 6.1), относятся к этим механизмам.

[1]FE Protocol Object - протокольный объект FE.

4.3.1. Транзакции, неделимость, выполнение и отклики

В режиме «ведущий-ведомый» CE инструктирует один или множество элементов FE о выполнении операций и отчётах о результате.

В этом параграфе описаны разные режимы, которые CE может запросить у элементов FE (см. параграф 4.3.1.1). Описаны также режимы форматирования откликов после выполнения запрошенной операции, которые CE может попросить у элементов FE. Эти режимы связаны с двухэтапными операциями транзакций.

4.3.1.1. Выполнение операций

Имеется 3 режима выполнения операций, которые могут быть запрошены для серии операций, охватывающей один или множество селекторов LFB (см. параграф 7.1.5) в одном протокольном сообщении. Флаг EM в общем заголовке (параграф 6.1) задаёт для протокольного сообщения режим выполнения операций:

a. execute-all-or-none (выполнить все или ничего);

b. continue-execute-on-failure (продолжать выполнение при возникновении отказа);

c. execute-until-failure (выполнять до первого отказа).

4.3.1.1.1. Все или ничего

При установке этого режима выполнения независимые операции в сообщении **могут** направляться одним или множеством селекторов LFB в элементе FE. Все эти операции выполняются последовательно и элемент FE **должен** не иметь отказов при выполнении какой-либо из этих операций. Если при выполнении любой из операций возникает отказ, все предыдущие операции, выполненные до отказа, будут отменены (с возвратом к прежнему состоянию). Т. е. это режим выполнения операций с откатом при возникновении любой ошибки.

Использование этого режима в транзакциях описано в параграфе 4.3.1.2.2. В этом случае не выполняется ни одной операции, пока не будет представления (commit) от CE.

При использовании этого режима следует соблюдать осторожность, поскольку конфигурационные ошибки могут приводить к потере трафика. Для обеспечения успеха операции её следует применять в этом режиме в транзакциях, как описано в параграфе 4.3.1.2.2

4.3.1.1.2. Продолжать при возникновении отказа

Если несколько независимых операций направляются одним или множеством селекторов LFB, выполнение операций продолжается в FE даже при возникновении одного или множества отказов.

4.3.1.1.3. Выполнять до первого отказа

В этом режиме все операции последовательно выполняются в FE, пока не возникнет первый отказ. Остальные операции не выполняются, а выполненные до отказа операции не отменяются. Т. е. этот режим не использует отката.

4.3.1.2. Транзакция и неделимость

4.3.1.2.1. Определение транзакции

Транзакция определяется как набор из одной или множества операций ForCES в одном или множестве сообщений PL, которые **должны** обладать свойствами ACIDity [ACID], как показано ниже.

Atomicity - неделимость
В транзакции, содержащей более одной дискретной части информации, выполняются все части или ни одной.

Consistency - согласованность
Транзакция создаёт новое действительное состояние данных или (при возникновении любого отказа) возвращает все данные в том состоянии, которое они имели до начала транзакции.

Isolation - изоляция
Транзакция, которая выполняется и не завершена, **должна** оставаться изолированной от всех прочих транзакций.

Durability - живучесть
Собранные данные сохраняются системой таким образом, что даже при возникновении отказа и перезапуске системы, данные остаются доступными в корректном состоянии.

Существуют ситуации, когда CE точно знает память и детали реализации FE (например, для пар FE-CE одного производителя, где FE и CE тесно связаны между собой). В таких случаях транзакции могут быть дополнительно упрощены за счёт дополнительных вычислений в CE. Этот случай более подробно не рассматривается в данном документе, но отмечается, что такое поведение не запрещено.

Как отмечено выше, транзакция всегда неделима и может выполняться в двух вариантах.

a. ***Внутри отдельного FE***
Например, обновление множества связанных между собой таблиц. При отказе во время обновления одной из таблиц уже обновлённые **должны** быть возвращены в исходное состояние.

b. ***Распределена в NE***
Например, обновление связанных таблиц в разных элементах FE (типа таблиц, связанных с пересылкой L3).

4.3.1.2.2. Протокол транзакций

Используя режим выполнения, как определено в параграфе 4.3.1.протокол обеспечивает механизм транзакций в рамках отдельного сообщения. Режим «все или ничего» может соответствовать требованиям ACID.

Для транзакций со множеством сообщений, выполняемых в одном или множестве FE, поддерживается классический протокол транзакций, известный как двухэтапная подача (two-phase commit - 2PC) [2PCREF], для обеспечения транзакционных операций, использующих сообщения Config (параграф 7.6.1).

Операции COMMIT и TRCOMP вместе с флагами AT и TP в общем заголовке (параграф 6.1) обеспечиваются для транзакций 2PC, включающих множество сообщений.

Флаг AT указывает, что сообщение относится к неделимой транзакции. Все сообщения для транзакции **должны** иметь флаг AT. Если флаг не установлен, это означает, что сообщение является автономным и не участвует в транзакции, включающей множество сообщений.

Флаг TP указывает фазу транзакции, к которой относится сообщение. Для транзакций возможны 4 фазы:

SOT (старт транзакции);

MOT (середина транзакции);

EOT (конец транзакции);

ABT (прерывание).

Операция COMMIT применяется CE для сигнализации элементам FE о представлении транзакции. При использовании с флагом ABT TP операция COMMIT сигнализирует элементам FE об отказе (т. е. un-COMMIT) от представленной ранее транзакции.

Операция TRCOMP является небольшим дополнением к классическому подходу 2PC. TRCOMP передаётся элементом CE для сигнала FE о представлении транзакции. Это даёт элементам FE возможность очистить состояние которое они могут сохранять для выполнения отката (при необходимости).

Транзакция начинается с сообщения, в котором флаг TP имеет значение SOT (старт транзакции). Многокомпонентные сообщения после первого указываются флагом MOT (середина транзакции). Все сообщения от CE **должны** иметь флаг AlwaysACK (параграф 6) для запроса откликов от FE.

До того, как CE отдаст команду на выполнение (см. ниже), элемент FE **должен** лишь подтвердить возможность операции, не выполняя её .

Любое уведомление от FE об отказе заставляет CE прервать транзакцию на всех вовлечённых в неё элементах FE. Это обеспечивается передачей сообщения Config с флагом ABT и операцией COMMIT.

Если ни на одном из участвующих в транзакции FE не обнаружено отказов, CE передаёт элементам FE сообщение Config с флагом EOT и операцией COMMIT.

Элемент FE **должен** ответить на сообщение EOT от CE. Существуют два случая отказов, при которых CE **должен** прервать транзакцию (как описано выше).

a. Если участвующий в транзакции элемент FE отвечает сообщением об отказе, относящемся к транзакции.

b. Если в течение заданного времени от участвующего в транзакции FE не получено отклика.

Если все участвующие FE возвращают индикацию успеха, элемент CE **должен** ввести операцию TRCOMP на всех участвующих FE. Элементам FE **недопустимо** отвечать на TRCOMP.

Отметим, что транзакции в общем случае неделимы, поэтому во всех сообщениях транзакции должен использоваться одинаковый режим execute-all-or-none. Если флаг EM указывает другой режим выполнения, это будет приводить к отказу транзакции.

Как отмечено выше, транзакция может включать множество сообщений. Элемент CE должен отслеживать остающиеся сообщения, участвующие в транзакции. Например, может использоваться поле корреляции для маркировки транзакций и поле номера (sequence) для обозначения разных сообщений одной неделимой транзакции, но этот вопрос выходит за рамки спецификации.

4.3.1.2.3. Восстановление

Любой из участвующих FE, CE или связи между ними могут столкнуться с отказом после того, как сообщение EOT Response было передано FE, но до того, как элемент CE получит все отклики (например, отклик EOT не достигнет CE).

В этом выпуске протокола, как указано в параграфе 4.2.2.3, элементу FE, потерявшему ассоциацию, потребуется новое состояние от заново привязанного CE после восстановления ассоциации. Хотя такой подход проще и обеспечивает очевидность потери трафика данных, он выбран для снижения сложности поддержки изящного перезапуска. Для поддержки непрерывной работы при отказах FE обеспечиваются механизмы HA (раздел 8).

Гибкость обеспечивает способом реагирования на потерю связи элементом FE. Это продиктовано политикой CE для восстановления при отказах (см. раздел 8 и параграф 7.3).

4.3.1.2.4. Пример сообщений транзакции

В этом параграфе приведён пример успешного двухфазного представления между CE и FE в простом случае.

Для показанного на рисунке 7 примера выполняются перечисленные ниже действия.

- На этапе 1 элемент CE передаёт сообщение Config с операцией типа DEL или SET. Устанавливаются флаги начала транзакции SOT, её неделимости AT и режима execute-all-or-none (все или ничего).
- FE проверяет возможность выполнения запроса и возвращает CE подтверждение в п. 2.
- На этапе 3 CE повторяет конструкцию этапа 1, но с флагом середины транзакции MOT.
- FE проверяет возможность выполнения запроса и возвращает CE подтверждение в п. 4.
- Обмен CE-FE продолжается аналогичным образом, пока все операции и их параметры не будут переданы элементу FE (это произойдёт на этапе N-1).
- На этапе N элемент CE передаёт команду представления (commit) с помощью сообщения Config с операцией типа COMMIT и флагом завершения транзакции EOT. Важно отметить, что это «пустое» сообщение

запрашивает у элемента FE выполнение всех операций, собранных с начала транзакции (сообщение 1).

- Элемент FE выполняет всю транзакцию целиком и передаёт CE подтверждение, содержащее COMMIT-RESPONSE, на этапе (N+1).
- CE в этом случае просто передаёт операцию TRCOMP элементу FE на этапе (N+2).

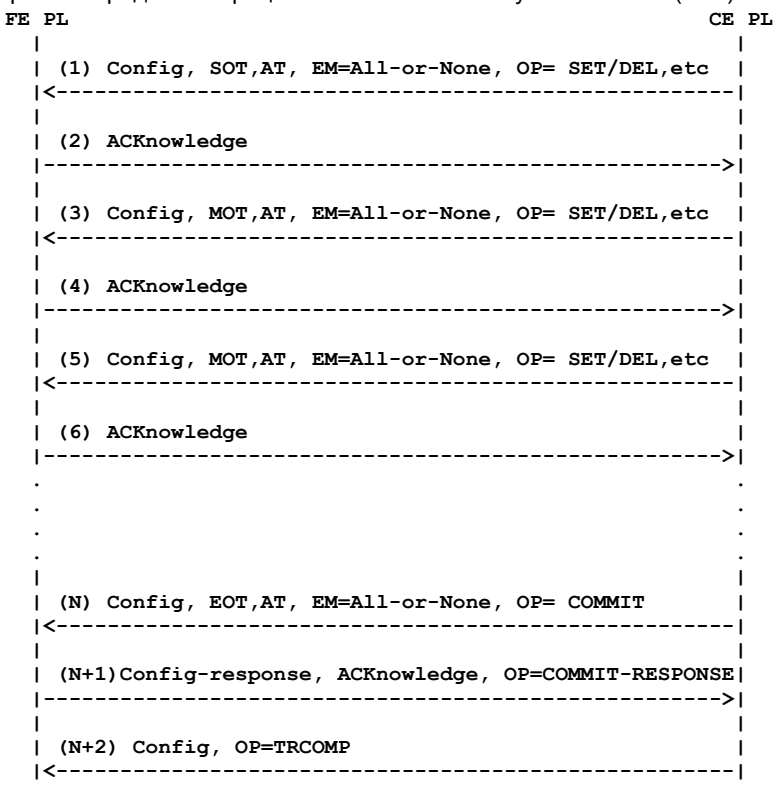

Рисунок 7. Пример 2-этапного представления.

4.3.2. Масштабируемость

Желательно, чтобы уровень PL не создавал «пробок», когда становятся доступными более широкополосные каналы. Например, при наличии каналов 100 Гбит/с и достаточном объёме работы уровню PL следует поддерживать возможность использования канала 100G. Для этого обеспечивается два механизма - пакетирование (batching) и конвейер команд (command pipeline).

Пакетирование обеспечивает возможность передачи множества команд (например, Config) в одном PDU[1]. Размер пакета (серии) команд зависит в том числе и от значения MTU для пути. Команды могут относиться к одной транзакции или быть не связанными между собой частями разных транзакций.

Конвейерная обработка позволяет организовать конвейер транзакций, которые не влияют одна на другую. Каждая из таких независимых транзакция может включать один или множество пакетов команд.

4.3.2.1. Серия операций

Существует несколько уровней пакетирования в разных протокольных иерархиях:

- множество PL PDU может объединяться в одно сообщение TML;
- множество классов и экземпляров LFB (указанных в селекторе LFB) могут размещаться в одном PL PDU;
- множество операций может быть сосредоточено в одном классе и экземпляре LFB.

4.3.2.2. Конвейер команд

Протокол позволяет элементу CE ввести любое число сообщений до того, как от FE будут получены подтверждения (если они запрошены). Поэтому конвейерная обработка является просто частью природы протокола. Сопоставление откликов с запросами может быть организовано с помощью специального поля correlator в заголовке сообщения.

4.3.3. Механизм Heartbeat

Сообщения Heartbeat (HB) между элементами FE и CE зависят от трафика. Сообщение HB передаётся только в том случае, когда в течение заданного интервала не было обычного трафика PL между CE и FE. Это снижает объем трафика HB в периоды занятости PL.

Сообщения HB могут передаваться элементами CE или FE. При отправке сообщения элементом CE может быть запрошен отклик (аналогично ICMP ping). FE может генерировать сообщения HB только в том случае, когда это задано в настройках элементом CE (см. параграфы 7.3.1 и 7.10).

4.3.4. Объект FE и протокольные LFB

Все сообщения PL оперируют конструкциями LFB и это обеспечивает гибкость для будущих расширений. Это означает, что поддержка и настройка FE, NE и самого протокола ForCES **должна** выражаться в терминах архитектуры LFB. По этой причине были созданы специальные LFB, реализующие эти потребности.

[1]Protocol Data Unit - блок данных протокола.

Кроме того, это показывает, как сам протокол ForCES может контролироваться с помощью того же типа структур (LFB), которые служат для управления пересылкой IP, фильтрацией и т. п.

Для решения этих задач были добавлены специализированные блоки LFB, которые перечислены ниже.

- FE Protocol LFB для управления протоколом ForCES.
- FE Object LFB для управления компонентами, относящимися к FE (FEState [RFC5812], производитель, и т. п.).

Эти LFB подробно описаны в параграфе 7.3.

4.4. Протокольные сценарии

В этом параграфе приводится описание верхнего уровня для примеров последовательностей сообщений между элементами CE и FE. Представление протокольных сообщений описано в параграфе 6.1, а семантика протокола - в параграфе 4.3.

4.4.1. Состояние создания ассоциации

Ассоциация между элементами CE и FE инициируется сообщением Association Setup от FE. Если элемент CE принимает запрос (Setup Request), он возвращает сообщение Setup Response элементу FE. Если CE и FE работают в незащищённой среде, между ними организуется защищённая связь до обмена какими-либо сообщениями ассоциации. Уровень TML **должен** обеспечить организацию защищённых связей.

После этого обычно следуют запросы возможностей, определение топологии и пр. Когда элемент FE готов начать обслуживание пути данных, он устанавливает для компоненты FEO FEState состояние OperEnable (см. [RFC5812]) и сообщает об этом элементу CE, к которому он обращался с первым запросом. Готовность FE к обслуживанию пути данные обычно предполагается ещё до создания ассоциации, но бывают (редкие) случаи, когда требуется некоторое время для той или иной предварительной обработки. В таких случаях FE будет начинать с состояния OperDisable и по мере готовности перейдёт в состояние OperEnable. Пример FE, начинающего работу в состоянии OperDisable и затем переходящего в OperEnable, показан на рисунке 8. Элемент CE может в любой момент запретить операции FE в пути данных, устанавливая для FEState значение AdminDisable. Элементу FE **недопустимо** обрабатывать пакеты, пока CE не восстановит для него состояние OperEnable. Эти последовательности сообщений показаны на рисунке 8.

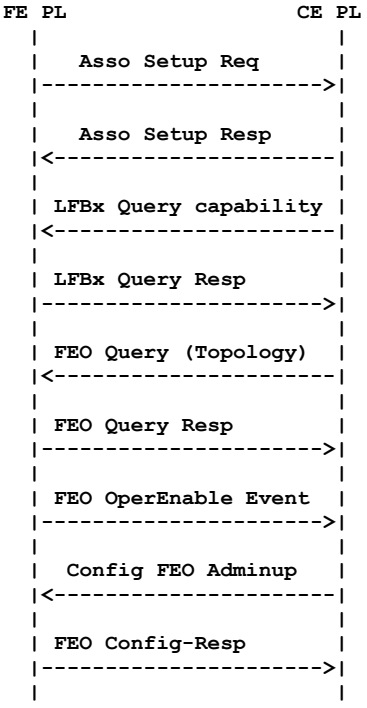

Рисунок 8. Обмен сообщениями между CE и FE для создания ассоциации NE.

При успешном завершении этого состояния FE присоединяется к элементу NE.

4.4.2. Состояние созданной ассоциации и установившееся состояние

В этом состоянии FE продолжает получать обновления и запросы. FE может также передавать элементу CE асинхронные уведомления о событиях, синхронные сообщения Heartbeat, а также сообщения Packet Redirect. Это продолжается до прерывания (или деактивации) ассоциации по инициативе CE или FE. Рисунок 9 иллюстрирует это состояние.

Отметим, что показанная на рисунке последовательность сообщений служит лишь примером и может отличаться от реальных последовательностей обмена сообщениями. Отметим также, что описанные здесь сценарии протокола не включают обменов сообщениями при возникновении отказа. Этот случай описан в разделе 8.

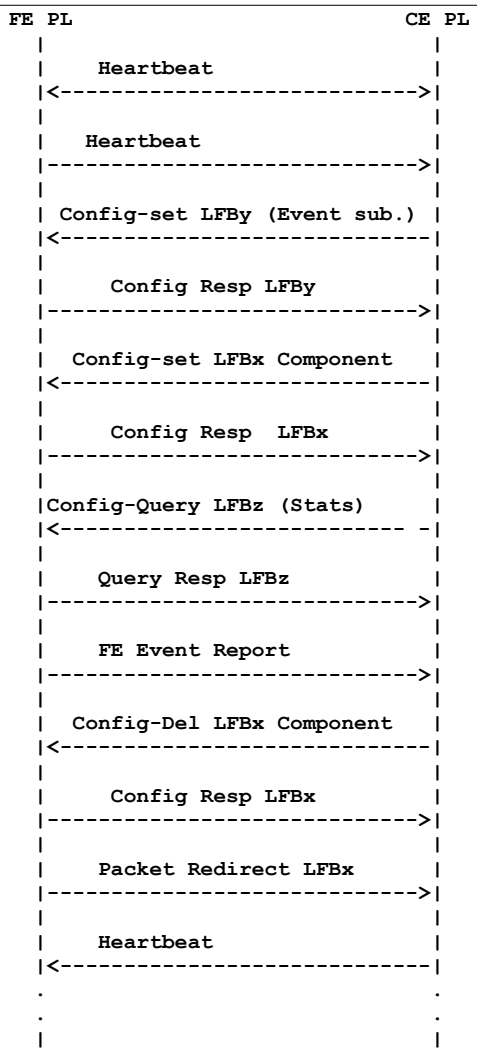

Рисунок 9. Обмен сообщениями между CE и FE в стационарном состоянии.

5. Требования к TML

Предполагается, что уровень TML будет соответствовать приведённым ниже требованиям. Этот текст не определяет способы реализации этих механизмов. Например, механизмы выполнения требований могут быть реализованы в оборудовании или между двумя и более программными процессами TML на разных CE и FE в схемах протокольного уровня.

Каждый TML **должен** описывать свой вклад в выполнение приведённых здесь требований ForCES. Если по какой-либо причине перечисленная здесь услуга не предоставляется, это должно быть обосновано.

Реализации, поддерживающие спецификацию протокола ForCES, **должны** реализовать [RFC5811]. Отметим, что в будущем могут быть заданы дополнительные TML и в случаях, когда новые TML будут обеспечивать лучшее выполнение приведённых требований, соответствующие требования могут быть переопределены.

1. ***Надёжность***
   Различные сообщений ForCES будут требовать разных уровней гарантии доставки через TML. Уровень TML отвечает за обеспечение таких гарантий и описание отображения разных сообщений ForCES на соответствующие уровни гарантий.

   Наиболее общий уровень надёжности, который мы называем строгим или твёрдым, обеспечивает отсутствие потерь, повреждений или нарушения порядка следования данных при транспортировке с обеспечением их своевременной доставки.

   Данные типа конфигурации от CE и откликов от FE имеют критическое значение и должны доставляться с обеспечением твёрдых гарантий. Поэтому для информации этого типа уровень TML **должен** должен включать встроенные протокольные механизмы или использовать транспортный протокол с твёрдыми гарантиями.

   Некоторые типы данных (например, перенаправленные пакеты или выборки пакетов) могут не требовать строгой гарантии (устойчивы к некому уровню потерь). Для этого типа данных TML может определять механизм без твёрдых гарантий (пр соблюдении остальных требований к TML типа контроля перегрузок).

   Для части информации типа пакетов heartbeat своевременность доставки может оказаться важнее гарантий надёжности. Для такой информации TML может определять применение механизма без твёрдых гарантий (пр соблюдении остальных требований к TML типа контроля перегрузок).

2. ***Безопасность***
   Уровень TML обеспечивает услуги защиты для ForCES PL. Поскольку уровень ForCES PL используется для работы NE, основными целями сетевых атак можно считать создание путаницы, запрет или утечку информации из основанных на протоколе ForCES элементов NE.

Атакующий с возможностью вставки ложных сообщений в поток PL, может воздействовать на обслуживание элементами FE пути данных (например, передавая ложные данные управления от имени CE) или сам элемент CE (изменяя уведомления о событиях или отклики от FE). По этой причине проверка подлинности элементов CE и FE, а также аутентификации сообщений TML имеют важное значение.

Сообщения PL могут также содержать важную для атакующего информацию, включая данные конфигурации сети, ключи шифрования и другие деликатные данные, поэтому следует принимать меры по ограничению доступа к таким данным (только уполномоченные пользователи).

- TML **должен** обеспечивать механизм проверки подлинности элементов ForCES CE и FE для предотвращения доступа неуполномоченных элементов CE и FE к управлению и обслуживанию пути данных для ForCES NE.
- TML **следует** обеспечивать механизм проверки подлинности сообщений с данными PL , передаваемых между элементами CE и FE для предотвращения вставки некорректных данных в сообщения PL.
- TML **следует** обеспечивать механизм защиты конфиденциальности данных, передаваемых от ForCES PL для предотвращения утечки информации уровня PL, передаваемой через TML.

Уровню TML **следует** поддерживать эти услуги с помощью TLS или IPsec.

3. ***Контроль насыщения***
   Должна быть определена схема контроля транспортных перегрузок, используемая TML. Механизм контроля насыщения, определённый TML, **должен** предотвращать коллапс насыщения [RFC2914] на стороне FE или CE.

4. ***Индивидуальная, групповая и широковещательная адресация и доставка***
   При наличии каких-либо отображений между индивидуальной, групповой и широковещательной адресацией между уровнями PL и TML, они должны быть определены.

5. ***Высокий уровень доступности***
   Предполагается, что за доступность транспортных соединений отвечает уровень TML. Однако с учётом конфигурации уровня PL он может принимать участие в схемах восстановления при отказах, поэтому уровень TML **должен** поддерживать эту возможность.

   Более подробная информация представлена в разделе 8.

6. ***Используемая инкапсуляция***
   Разные типы TML будут инкапсулировать сообщения PL с использованием разных типов заголовков. Уровень TML должен указывать применяемую инкапсуляцию.

7. ***Приоритизация***
   Предполагается, что TML может поддерживать до 8 уровней приоритета, требуемых PL, и будет обеспечивать соответствующее обслуживание.

   Хотя от TML требуется определить, как обеспечивается поддержка приоритизации, следует отметить, что требование поддержки до 8 уровней приоритета вовсе не означает, что TML **должен** реально обеспечить эти уровни. Если базовый сервис TML не поддерживает 8 уровней приоритета, поддерживаемые уровни следует поделить между доступными в TML уровнями приоритета. Например, если TML поддерживает лишь два уровня, приоритеты 0-3 могут быть отображены на один из них, а 4-7 - на другой.

   Уровню TML **недопустимо** менять порядок следования пакетов config с одинаковым уровнем приоритета.

8. ***Предотвращение перегрузки узла***
   Уровень TML **должен** определять механизмы, которые он использует для предотвращения перегрузки узла.

   Перегрузка приводи к истощению вычислительных и/или канальных ресурсов узла и это снижает рабочую производительность ForCES NE. Перегрузка узла NE может быть осознанно спровоцирована злонамеренным узлом для атаки ForCES NE и отказа в обслуживании (DoS[1]). Перегрузка может возникать и по многим другим причинам типа значительных обновлений данных протоколов (например, осцилляции BGP), которые вызывают передачу многочисленных обновлений от CE в таблицы FE, переключения HA или отказа компонент, в результате которого работа FE или CE переносится на новый элемент FE или CE, и т. п. Хотя среды, в которых работают протоколы SIP и ForCES различаются, [RFC5390] обеспечивает хорошее руководство с базовыми требованиями, которым должен соответствовать защищаемый узел.

   Процессор узла ForCES может быть перегружен в результате слишком высокой скорости входящих потоков. В таких случаях транспортные очереди растут и может возникнуть транспортная перегрузка. Процессор узла ForCES может также быть перегружен в результате преднамеренной передачи небольшого числа пакетов без перегрузки транспорта (например, DoS-атака на алгоритм хэширования таблиц, приводящая к их переполнению и/или высокой загрузке CPU, не позволяющей обрабатывать другие задачи). Решение TML по предотвращению перегрузки узла **должно** препятствовать возникновению обоих вариантов перегрузки.

5.1. Параметризация TML

Предполагается, что для настройки TML используются средства типа менеджеров FEM или CEM.

Ниже перечислены некоторые из настраиваемых параметров.

- PL ID.
- Тип соединения и связанные с ним данные. Например, если TML использует IP/TCP/UDP, настраиваемыми параметрами будут номер порта TCP/UDP и адрес IP.

[1]Denial of service.

- Число транспортных соединений.
- Ёмкость соединения (пропускная способность и т. п.).
- Разрешённые/поддерживаемые правила QoS (или контроля перегрузок) для соединения.

6. Инкапсуляция сообщений

Все PL PDU начинаются с общего заголовка, описанного в параграфе 6.1, за которым следует один или множество блоков TLV, описанных в параграфе 6.2, и могущих включать в себя другие TLV, как описано в параграфе 6.2.1. Все поля заголовка используют сетевой порядок байтов.

6.1. Общий заголовок

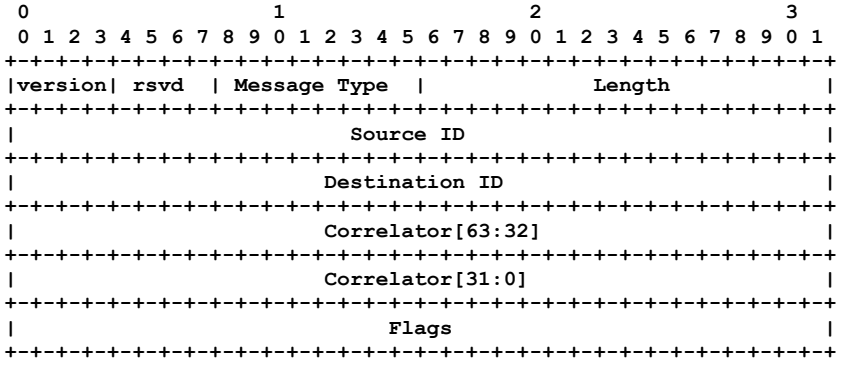

Рисунок 10. Общий заголовок.

Сообщения выравниваются по 32-битовым границам.

version (4 бита)
Номер версии (текущее значение 1).

rsvd (4 бита)
Резервное поле, которое не анализируется получателем. Отправитель **должен** установить значение 0, а получатель **должен** игнорировать это поле.

Message Type (8 битов)
Команды, определённые в разделе 7.

Length (16 битов)
Размер заголовка и остальной части сообщения в DWORD (4 байта).

Source ID (32 бита)
Dest ID (32 бита)

- Каждый идентификатор отправителя и получателя представляет собой 32-битовое значение, уникальное в масштабе NE, которое служит для указания конечной точки сообщения ForCES PL.
- Идентификаторы поддерживают индивидуальную, групповую и широковещательную адресации, как показано ниже.
  a. Используется расщеплённое адресное пространство для того, чтобы отличать элементы FE и CE. Хотя в больших NE число FE на один-два порядка больше числа CE, адресное пространство для простоты распределено однородно.
  b. Адресное пространство позволяет указать до 2^{30} (более миллиарда) элементов CE и столько же FE.

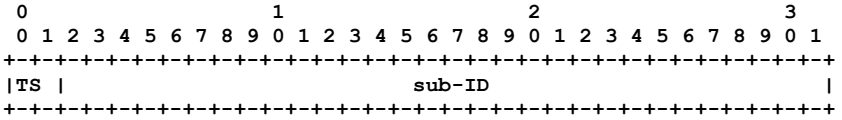

Рисунок 11. Формат ForCES ID.

  c. 2 старших бита TS (Type Switch) служат для разделения пространства идентификаторов:

TS	Диапазон идентификаторов	Назначение
0b00	0x00000000 - 0x3FFFFFFF	FE ID (2^30)
0b01	0x40000000 - 0x7FFFFFFF	CE ID (2^30)
0b10	0x80000000 - 0xBFFFFFFF	резерв
0b11	0xC0000000 - 0xFFFFFFEF	групповые ID (2^30 - 16)
0b11	0xFFFFFFF0 - 0xFFFFFFFC	резерв
0b11	0xFFFFFFFD	широковещание всем CE
0b11	0xFFFFFFFE	широковещание всем FE
0b11	0xFFFFFFFF	широковещание всем FE и CE (NE)

Рисунок 12. Пространство типов Switch ID.

- Групповые и широковещательные ID используются для групп конечных точек (типа CE и FE). Например, это может быть функциональная группа FE. Аналогично может быть выделена подгруппа CE, например, в режиме резервирования, чтобы скрыть их от FE:
  - Multicast ID могут использоваться для указания отправителей и получателей;
  - Broadcast ID могут применяться только для получателей.
- В этом документе не рассматривается связи конкретных групповых идентификаторов с той или иной группой, которая может быть задана в процессе настройки. Список идентификаторов, которыми владеют FE (или входят в список), указывается в FE Object LFB.

Correlator (64 бита)
Это поле устанавливается элементом CE для сопоставления сообщений ForCES Request с откликами от FE. По сути, это поле играет роль cookie. Поле correlator элемент FE **должен** копировать в отклик из соответствующего запроса. Если сообщение от CE не вызывает отклика, это поле не имеет смысла.

Поле correlator может использоваться разными реализациями по усмотрению элементов CE. Например, CE может разделить поле на 32-битовые идентификаторы транзакции и порядкового номера сообщения. Другим примером может служить 64-битовый указатель на блок контекста. Конкретные варианты использования поля correlator выходят за рамки спецификации.

Следует отметить, что поле correlator передаётся через сеть как 64-битовое целое число без знака, начиная с левого или старшего октета (биты 63-56).

Когда поле correlator не требуется по причине того, что не ожидается отклика, следует указывать correlator = 0.

Flags (32 бита)

Флаги заголовка показаны на рисунке.

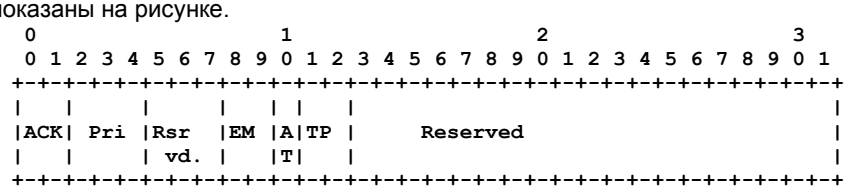

Рисунок 13. Флаги заголовка.

ACK - индикатор ACK (2 бита)

Флаг ACK используется только CE при передаче сообщения Config (параграф 7.6.1) или HB (параграф 7.10), чтобы указать получателю должен ли он передать отклик. Отметим, что для всех сообщений, кроме Config и HB, этот флаг **должен** игнорироваться.

Значения флага показаны ниже.

NoACK (0b00) указывает, что получателю **недопустимо** передавать в ответ сообщение Response.

SuccessACK (0b01) указывает, что получатель **должен** передать в ответ сообщение Response только после успешной обработки принятого сообщения.

FailureACK (0b10) указывает, что получатель **должен** передать в ответ сообщение Response только при возникновении отказа в процессе обработки (выполнения) сообщения. Иными словами, отправитель не ждёт отклика при успешной обработке сообщения получателем.

AlwaysACK (0b11) указывает, что получатель **должен** передать сообщение Response в любом случае.

Отметим, что в приведённых выше определениях термин «успех» означает полное выполнение без каких-либо отказов для сообщения. Все прочие результаты считаются отказами при обработке сообщения. В результате этого для режимов выполнения (определены в параграфе 4.3.1.1) типа execute-all-or-none, execute-until-failure и continue-execute-on-failure отказ любой операции из сообщения будет считаться отказом всего сообщения и приводить к отправке соответствующего отклика, если индикатор ACK имеет значение FailureACK или AlwaysACK.

Отметим также, что для сообщений, отличных от Config и HB, требования к откликам всегда задаются принятым по умолчанию способом, а не флагами ACK. Требования к принятой по умолчанию генерации откликов приведены ниже, а подробные описания приведены в определениях соответствующих сообщений.

- Для Association Setup всегда ожидается передача отклика.
- Для сообщений Association Teardown и Packet Redirect отклики не ожидаются.
- Для сообщения Query всегда ожидается отклик.
- Для сообщений Response дополнительные отклики не ожидаются.

Pri - Priority (3 бита)

Протокол ForCES определяет 8 уровней приоритета (0-7), которые могут применяться для того, чтобы различать разные типы сообщений, а также разные сообщения одного типа. Чем больше значение приоритета, тем более важным является PDU. Например сообщение REDIRECT может иметь разные уровни приоритета для протоколов маршрутизации и пакетов ARP, перенаправляемых от FE к CE. Обычный уровень приоритета имеет значение 1. Сообщения с разным приоритетом могут менять свой порядок, однако изменение порядка в наборе сообщений одной транзакции нежелательно и следует его избегать.

EM - Execution Mode (2 бита)

Поддерживается три режима выполнения, определённых в параграфе 4.3.1.1.

резерв	(0b00);
execute-all-or-none	(0b01);
execute-until-failure	(0b10);
continue-execute-on-failure	(0b11).

AT - Atomic Transaction (1 бит)

Этот флаг показывает является сообщение автономным или одним из группы сообщений, относящихся к транзакции 2PC (см. параграф 4.3.1.2.2).

автономное сообщение	(0b0);
сообщение из транзакции 2PC	(0b1).

TP - Transaction Phase (2 бита)

Сообщение от CE к FE в транзакции может указывать разные фазы этой транзакции (см. параграф 4.3.1.2.2).

SOT	(0b00)
MOT	(0b01)
EOT	(0b10)
ABT	(0b11)

6.2. Структуры TLV

Структуры TLV[1] широко используются протоколом ForCES. TLV имеют ряд очень удобных свойств, которые делают их подходящими кандидатами для кодирования определений XML модели класса LFB. Эти свойства перечислены ниже.

- Двоичное представление type-value, близкое к схеме представления строк XML tag-value.
- Возможность быстрой генерации функций разбора.
- Совместимость тегов с новыми и старыми версиями. Это эквивалентно подходу XML, т. е. старые приложения могут игнорировать новые TLV, а новые приложения могут игнорировать старые TLV.

[1]Type Length Value - тип, размер, значение.

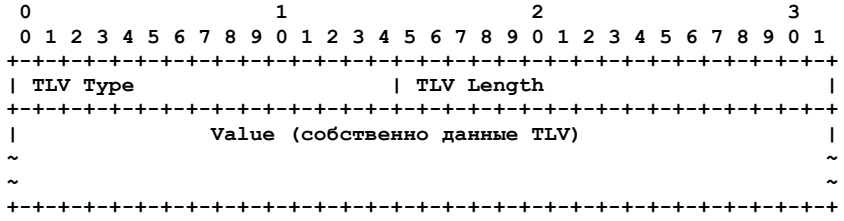

Рисунок 14. Представление TLV.

TLV Type (16)

Поле типа TLV имеет размер 2 октета и семантически показывает тип данных, инкапсулированных в TLV.

TLV Length (16)

Поле размера TLV занимает 2 октета и учитывает размер поля типа TLV (2 октета), TLV Length (2 октета), а также размер данных в поле значения TLV (в октетах). Отметим, что заполнение в поле значения не учитывается в размере.

TLV Value (переменный размер)

Поле значение TLV содержит данные. Для расширяемости значением TLV может быть другой блок TLV. Если размер значения не кратен 32 битам, требуется использовать заполнение минимальным числом октетов, требуемым для выравнивания TLV по 32-битовой границе. Размер значения до заполнения учитывается в поле TLV Length.

Примечание. Поле значения может быть пустым, поэтому минимальный размер TLV составляет 4 октета (размеры полей T и L).

6.2.1. Вложенные TLV

Значениями TLV могут быть другие TLV. Это обеспечивает протоколу гибкость (возможность добавлять расширения путём определения новых TLV). Вложенные TLV также позволяют концептуальную оптимизацию определений XML LFB для двоичного представления PL (использует вложенные TLV).

6.2.2. Область действия типа (T) в TLV

Имеются две глобальных области действия типа в TLV. Первой областью являются OPER-TLV, определённые в Приложении A.4, а вторая область не связана с OPER-TLV и определена в Приложении A.2.

6.3. ILV

ILV представляет собой вариацию TLV. Здесь в качестве типа (T) служит 32-битовый локальный индекс, который указывает идентификатор компоненты ForCES (см. параграф 6.4.1).

Поле ILV представляет собой 4-октетное целое число, которое учитывает размер типа ILV (4 октета), ILV Length (4 октета) и размер данных ILV в поле value (в октетах). Отметим, что как и для TLV, этот размер не учитывает заполнения поля value, если оно применяется.

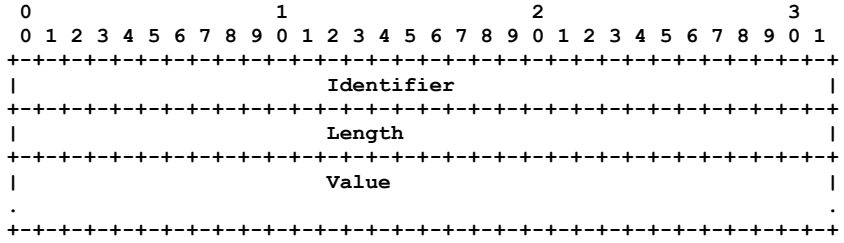

Рисунок 15. Представление ILV.

Следует подчеркнуть, что значение I имеют локальную значимость и определяются декларациями данных из определения LFB. Применение ILV рассматривается в параграфе 7.1.8.

6.4. Вопросы инкапсуляции

В этом параграфе рассматривается несколько концепций инкапсуляции, которые используются протоколом ForCES для своих операций.

Для согласования контекста здесь повторяются концепции для путей и ключей из модели ForCES [RFC5812]. Читателю рекомендуется обратиться за подробностями к [RFC5812].

Для удобочитаемости введены схемы инкапсуляции, которые служат для передачи содержимого протокольных сообщений, а именно, FULLDATA-TLV, SPARSEDATA-TLV и RESULT-TLV.

6.4.1. Пути

Модель ForCES [RFC5812] определяет основанный на XML язык, который позволяет формально определять блоки LFB. Это похоже на взаимоотношения ASN.1 и определений SNMP MIB (MIB является аналогом LFB, ASN.1 - аналогом языка моделирования XML). Любой объект, который CE настраивает в FE, **должен** быть формально определён в LFB. Эти объекты могут быть скалярами (например, 32-битовый адрес IPv4) или векторами (например, таблица nexthop). Каждый объект в LFB задаётся 32-битовым числовым идентификатором, который называют component id. Данная схема позволяет «адресовать» компоненты в протокольной конструкции.

Эти адресуемые объекты могут быть иерархическими (например, колонка или ячейка в строке таблицы). Для адресации иерархических данных в модели [RFC5812] введена концепция пути. Путь представляет собой последовательность 32-битовых идентификаторов компонент, которые обычно разделяются точками (например, 1.2.3.4). Формальное определение использования путей для указания данных, которые будут инкапсулироваться в протокольное сообщение, приведено в разделе 7.

6.4.2. Ключи

Модель ForCES [RFC5812] определяет два способа адресации строк в таблицах. Стандартный (базовый) механизм позволяет представлять строку 32-битовым индексом. Второй механизм, основанный на использовании ключей, позволяет адресовать содержимое. Примером ключа может служить ключ содержимого с множеством полей, который использует адрес IP и размер префикса для однозначного указания строки таблицы маршрутизации IPv4. Сути, базовая схема адресации строк таблиц основана на индексах, а путь строки таблицы может быть выведен из ключа. KEYINFO-TLV (раздел 7) служит для переноса данных, которые используются при поиске.

6.4.3. TLV данных

Данные от элементов FE и к ним передаются в двух типах TLV - FULLDATA-TLV и SPARSEDATA-TLV. Отклики для выполненных FE операций передаются в RESULT-TLV.

Данные в FULLDATA-TLV кодируются таким образом, чтобы получатель, зная путь и определение LFB, мог вывести или сопоставить содержимое поля Value из TLV. Такая оптимизация помогает уменьшить объем описаний, требуемых для транспортируемых данных в грамматике протокола. Примеры FULLDATA-TLV представлены в Приложении C.

Ряду операций ForCES требуется указывать дополнительные данные в более крупных структурах. Кодирование SPARSEDATA-TLV обеспечивает упрощение инкапсуляции необязательных компонент данных. Примеры SPARSEDATA-TLV приведены в Приложении C.

RESULT-TLV передают отклики от FE на основе конфигурации, заданной CE. Схема RESULT-TLV описана в параграфе 7.1.7, а примеры представлены в Приложении C.

6.4.4. Адресация объектов LFB

В параграфах 6.4.1 и 6.4.2 описана адресация объектов внутри LFB. Однако используемый механизм адресации требует сначала выбрать тип и экземпляр LFB. Для такого выбора служит селектор LFB. В разделе 7 это описано более подробно, а здесь для иллюстрации концепции используется рисунок 16. Дополнительные примеры схем приведены ниже (например, рисунок 22).

```
main hdr (Тип сообщения. Например, config)
 |
 |
 |
 +- T = LFBselect
        |
        +-- LFBCLASSID (unique per LFB defined)
        |
        |
        +-- LFBInstance  (рабочая конфигурация)
        |
        +-- T = TLV операции указывает что мы делаем с объектом
            |   //Указывает значения OPER-TLV, перечисленные под
            |   //TLV, которые могут быть использованы для операций
            |
            |
            +--+-- Один или множество путей к цели объекта
               | // Указывает на обсуждение ключей и путей
               |
               |
               +-- Связанные данные для объекта (если они есть)
                 // Указывает на обсуждение FULL/SPARSE DATA TLV
```

Рисунок 16. Адресация объекта.

7. Устройство протокола

Блок PDU протокольного уровня состоит из базового заголовка, определённого в параграфе 6.1, и тела сообщения. После общего заголовка следует тело сообщения, зависящее от типа этого сообщения. Тело каждого сообщения состоит из одного или множества TLV верхнего уровня. Эти блоки TLV могут включать один или множество встроенных TLV, которые указываются в этом документа как OPER-TLV, поскольку они описывают выполняемые операции.

Таблица 1.

Имя сообщения	*TLV верхнего уровня*	*OPER-TLV*	*Параграф*
Association Setup	(LFBselect)*	REPORT	7.5.1
Association Setup Response	ASRresult-TLV	нет	7.5.2
Association Teardown	ASTreason-TLV	нет	7.5.3
Config	(LFBselect)+	(SET | SET-PROP | DEL | COMMIT | TRCOMP)+	7.6.1
Config Response	(LFBselect)+	(SET-RESPONSE | SET-PROP-RESPONSE | DEL-RESPONSE | COMMIT-RESPONSE)+	7.6.2
Query	(LFBselect)+	(GET | GET-PROP)+	7.7.1
Query Response	(LFBselect)+	(GET-RESPONSE | GET-PROP-RESPONSE)+	7.7.2
Event Notification	LFBselect	REPORT	7.8
Packet Redirect	REDIRECT-TLV	нет	7.9
Heartbeat	нет	нет	7.10

Сообщения разных типов перечислены в таблице 1, а числовые значения типов указаны в Приложении A.1. Определения для всех TLV даны в Приложении A.2.

LFBselect TLV (см. параграф 7.1.5) содержит LFB Classid и экземпляр LFB, на который ссылаются, а также OPER-TLV для выполняемых операций.

Каждый тип OPER-TLV ограничен в части описания интересующих путей и селекторов. Приведённая ниже форма BNF описывает базовую структуру OPER-TLV, а в таблице 2 приведены подробности для каждого типа.

```
OPER-TLV := 1*PATH-DATA-TLV
PATH-DATA-TLV := PATH  [DATA]
PATH := flags IDcount IDs [SELECTOR]
SELECTOR :=  KEYINFO-TLV
DATA := FULLDATA-TLV / SPARSEDATA-TLV / RESULT-TLV / 1*PATH-DATA-TLV
KEYINFO-TLV, V= {1, FULLDATA-TLV V={100}}
FULLDATA-TLV := кодированная компонента данных, которая может включать вложенные FULLDATA-TLV
SPARSEDATA-TLV :=  кодированная компонента данных (может включать необязательные компоненты)
RESULT-TLV := код результата и необязательный FULLDATA-TLV
```

Рисунок 17. Форма BNF для OPER-TLV.

- PATH-DATA-TLV указывает целевую компоненту и может (но не обязательно) включать связанные с ней пути. Последний блок PATH-DATA-TLV для случая вложенных путей с использованием конструкции DATA в запросах SET и SET-PROP, а также откликах GET-RESPONSE и GET-PROP-RESPONSE завершается представленными данными или откликом в форме FULLDATA-TLV, SPARSEDATA-TLV или RESULT-TLV.
- PATH указывает путь к данным, на которые ссылаются.
  - flags (16 битов) используются для уточнения операции, применяемой на пути (детали приведены ниже).
  - IDcount (16 битов) - счётчик 32-битовых значений ID.
  - IDs - необязательные 32-битовые идентификаторы (число которых указано в Idcount), определяющие основной путь. В зависимости от флагов (flags) идентификаторы могут быть только полями ID или комбинацией полей и динамических ID. Отсутствие идентификаторов является особым случаем использования контекста в качестве результата пути.
- SELECTOR представляет собой необязательную конструкцию, дополняющую PATH. Единственным определенным в настоящее время селектором является KEYINFO-TLV, используемый для выбора элемента массива по значению поля key. Присутствие SELECTOR корректно лишь в тех случаях, когда оно указано полем flags.
- KEYINFO-TLV представляет информацию, используемую для ключа содержимого.
  - 32-битовый идентификатор KeyID используется в KEYINFO-TLV для указания ключа, который будет служить в текущем массиве для выбора элемента.
  - Данные ключа key используются для поиска в массиве среди указанных ключом полей. Информация кодируется в соответствии с правилами для содержимого FULLDATA-TLV и представляет поле или поля, которые образуют ключ, указанный KeyID.
- DATA может включать FULLDATA-TLV, SPARSEDATA-TLV, RESULT-TLV или один или несколько дополнительных выборов PATH-DATA. FULLDATA-TLV и SPARSEDATA-TLV разрешены только в запросах SET или SET-PROP, а также в откликах, возвращающих информацию содержимого (например, GET-RESPONSE). Можно включать также PATH-DATA для расширения пути в любом запросе.
  - Примечание. Вложенные PATH-DATA-TLV поддерживаются в качестве эффективного способа преобразования общих субвыражений.
  - FULLDATA-TLV и SPARSEDATA-TLV содержат данные, путь к которым указан с помощью PATH (см. параграф 7.1.
  - В таблице 2 показана применимость и ограничения для FULLDATA-TLV и SPARSEDATA-TLV, а также RESULT-TLV для OPER-TLV.

Таблица 2.

OPER-TLV	***DATA TLV***	***RESULT-TLV***
SET		нет
SET-PROP	(FULLDATA-TLV | SPARSEDATA-TLV)+	нет
SET-RESPONSE	нет	(RESULT-TLV)+
SET-PROP-RESPONSE	нет	(RESULT-TLV)+
DEL		нет
DEL-RESPONSE	нет	(RESULT-TLV)+
GET	нет	нет
GET-PROP	нет	нет
GET-RESPONSE	(FULLDATA-TLV)+	(RESULT-TLV)*
GET-PROP-RESPONSE	(FULLDATA-TLV)+	(RESULT-TLV)*
REPORT	(FULLDATA-TLV)+	нет
COMMIT	нет	нет
COMMIT-RESPONSE	нет	(RESULT-TLV)+
TRCOMP	нет	нет

- RESULT-TLV содержит индикацию результата отдельной операции SET или SET-PROP. RESULT-TLV включается в предположении, что отдельные части запроса SET могут быть неудачными или успешными независимо от других.

В заключение отметим, характеристики описанного подхода.

- В сообщении может быть одна или несколько комбинаций идентификаторов класса и экземпляра LFB (пакет).
- Может быть одна или множество операций, указанных комбинацией идентификаторов класса и экземпляра LFB (пакет).
- Может быть одна или множество целей путей на операцию (пакет).
- Путь может иметь множество (возможно пустое) связанных с ним значений данных (гибкость и привязка к операциям).

Следует отметить, что приведённое выше описание оптимизировано для случая одного идентификатора класса и экземпляра LFB. Для множества экземпляров одного класса требуется множество LFBselect.

7.1. Кодирование

В параграфе 6.4.3 рассмотрены два типа кодирования DATA (FULLDATA-TLV и SPARSEDATA-TLV), а также дано обоснование их наличия. Здесь кодирование описывается более подробно.

7.1.1. Правила упаковки данных

Ниже приведены правила, которые используются в принятой здесь схеме кодирования данных.

- Поле Value (V в TLV) в FULLDATA-TLV будет содержать передаваемые данные, как описано в определении LFB.
- Данные переменного размера в FULLDATA-TLV будут инкапсулироваться внутри другого FULLDATA-TLV внутри V внешнего TLV. Примеры такого кодирования представлены в Приложениях C и D.
- В случае FULLDATA-TLV.
  - Когда таблица указана в PATH (IDs) блока PATH-DATA-TLV, значение FULLDATA-TLV (V) будет включать содержимое строки этой таблицы с 32-битовым префиксом index/subscript. С другой стороны, PATH может содержать индекс, указывающий строку таблицы, и в этом случае значение FULLDATA-TLV (V) будет содержать только индекс для предотвращения неоднозначности.

7.1.2. Флаги пути

В поле flags для пути в настоящее время используется только бит 0 (SELECTOR), как показано на рисунке 18.

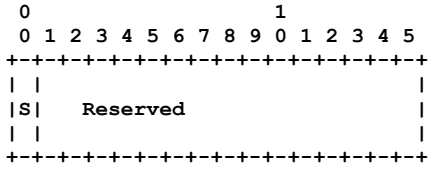

Рисунок 18. Флаги пути.

Семантика флага определена ниже.

- Флаг SELECTOR - F_SELKEY(1) показывает присутствие KEY Selector вслед за этой информацией пути и его следует учитывать при определении содержимого пути.

7.1.3. Связь операционных флагов с глобальными флагами сообщения

Глобальные флаги типа индикаторов режима выполнения и неделимости, определённые в заголовке, применяются ко всем операциям в сообщении. Эти флаги обеспечивают семантику, которая ортогональная семантике операционных флагов типа тех, что определены в path-data. Область действия операционных флагов ограничена одной операцией.

7.1.4. Выбор пути к содержимому

KEYINFO-TLV описывает ключ KEY и связанные с ним данные. Ключи служат для поиска содержимого, ограничены и описываются в определении LFB.

7.1.5. LFBselect-TLV

LFBselect TLV является экземпляром TLV в соответствии с определением параграфа 6.2. Схема показана на рисунке 19.

Type
Тип TLV - LFBselect.

Length
Размер TLV в октетах с учётом полей T и L.

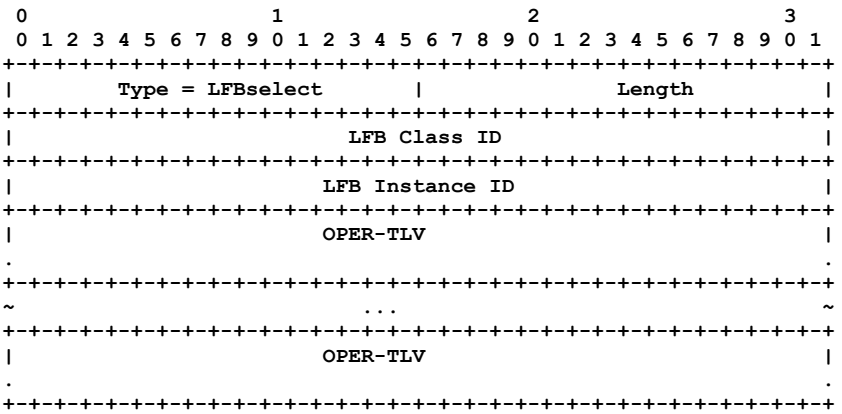

Рисунок 19. Схема PL PDU.

LFB Class ID
Это поле однозначно указывает класс/тип LFB.

LFB Instance ID
Это поле однозначно указывает экземпляр LFB.

OPER-TLV

Описывает операции, встроенные в LFBselect TLV. Отметим, что обычно **следует** включать не менее одного OPER-TLV для TLV выбора LFB.

7.1.6. OPER-TLV

OPER-TLV представляет TLV, которые определяют операции. Типы операций показаны в таблице 3.

Таблица 3.

OPER-TLV	*Тип TLV*	*Комментарии*
SET	0x0001	От CE к FE. Служит для создания, добавления или обновления компонент.
SET-PROP	0x0002	От CE к FE. Служит для создания, добавления или обновления свойств компонент.
SET-RESPONSE	0x0003	От FE к CE. Служит для передачи отклика SET.
SET-PROP-RESPONSE	0x0004	От FE к CE. Служит для передачи отклика SET-PROP.
DEL	0x0005	От CE к FE. Служит для удаления или исключения компонент.
DEL-RESPONSE	0x0006	От FE к CE. Служит для передачи отклика DEL
GET	0x0007	От CE к FE. Служит для получения компонент.
GET-PROP	0x0008	От CE к FE. Служит для получения свойств компонент.
GET-RESPONSE	0x0009	От FE к CE. Служит для передачи отклика GET.
GET-PROP-RESPONSE	0x000A	От FE к CE. Служит для передачи отклика GET-PROP.
REPORT	0x000B	От FE к CE. Служит для передачи асинхронных событий.
COMMIT	0x000C	От CE к FE. Служит для подачи на исполнение транзакции 2PC.
COMMIT-RESPONSE	0x000D	От FE к CE. Служит для подтверждения подачи транзакции 2PC.
TRCOMP	0x000E	От CE к FE. Служит для индикации успешной транзакции 2PC в масштабе NE.

OPER-TLV используется в разных сообщениях, как указано в таблицах 1 и 2.

Запросы SET, SET-PROP и GET/GET-PROP подаются элементом CE и не включают RESULT-TLV. С другой стороны, отклики SET-RESPONSE, SET-PROP-RESPONSE и GET-RESPONSE/GET-PROP-RESPONSE содержат RESULT-TLV.

GET-RESPONSE в отклике на успешный запрос GET будет иметь FULLDATA-TLV, добавленные к путям листьев для передачи запрошенных данных. Для отказавших операций GET вместо FULLDATA-TLV будет RESULT-TLV.

Для SET-RESPONSE/SET-PROP-RESPONSE каждый FULLDATA-TLV или SPARSEDATA-TLV в исходном запросе будет заменяться в отклике на RESULT-TLV. Если в запросе установлен флаг FailureACK, в отклике будут указаны только отказавшие элементы. При установленном флаге AlwaysACK все компоненты запроса будут в отклике с RESULT-TLV.

Отметим, что при запросе SET/SET-PROP со структурой в FULLDATA-TLV, имеющей непригодные поля, FE не будет пытаться указать такие поля, а просто покажет отказ операции. При наличии множества ошибок в одном листе PATH-DATA/FULLDATA-TLV элемент FE может выбирать ошибку для возврата. Если FULLDATA-TLV для SET/SET-PROP в структуре пытается записать поле, доступное только для чтения, а в другое поле записать непригодное значение, FE может выбрать для возврата любую из этих ошибок.

Операция SET/SET-PROP для компоненты переменного размера с length = 0 не то же самое, что её удаление. Если CE хочет удалить компоненту, следует применять операцию DEL с путём, указывающим компоненту массива или необязательную компоненту структуры.

7.1.7. RESULT TLV

RESULT-TLV является экземпляром TLV в соответствии с определением параграфа 6.2. Схема показана на рисунке.

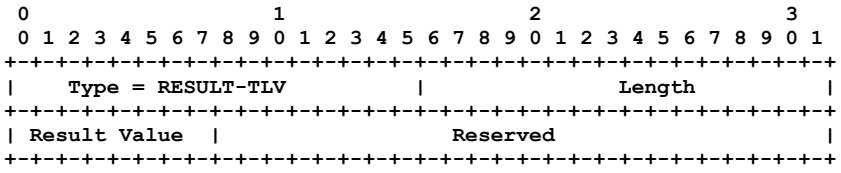

Рисунок 20. RESULT-TLV.

Таблица 4. Определённые значения результатов.

Результат	*Значение*	*Определение*
E_SUCCESS	0x00	Успех.
E_INVALID_HEADER	0x01	Неуказанная ошибка заголовка.
E_LENGTH_MISMATCH	0x02	Размер заголовка не соответствует реальному размеру пакета.
E_VERSION_MISMATCH	0x03	Непреодолимое несоответствие версий.
E_INVALID_DESTINATION_PID	0x04	Destination PID не действителен для получателя сообщения.
E_LFB_UNKNOWN	0x05	LFB Class ID не известен получателю.
E_LFB_NOT_FOUND	0x06	LFB Class ID известен получателю но сейчас не используется.
E_LFB_INSTANCE_ID_NOT_FOUND	0x07	LFB Class ID известен, но указанного экземпляра класса не существует.
E_INVALID_PATH	0x08	Заданный путь не возможен.
E_COMPONENT_DOES_NOT_EXIST	0x09	Заданный путь возможен, но компонента не существует (например, попытка изменить не созданную строку таблицы).
E_EXISTS	0x0A	Заданный объект существует, но его не должно быть для успеха операции (например, попытка добавить имеющийся экземпляр LFB или индекс массива).
E_NOT_FOUND	0x0B	Заданного объекта нет, но он **должен** быть для успеха операции (например, попытка удалить отсутствующий экземпляр LFB или индекс массива).
E_READ_ONLY	0x0C	Попытка изменить значение, открытое только для чтения.
E_INVALID_ARRAY_CREATION	0x0D	Попытка создать массив с неразрешенный индексом (subscript).
E_VALUE_OUT_OF_RANGE	0x0E	Попытка установить значение параметра за пределами диапазона.

E_CONTENTS_TOO_LONG	0x0D	Попытка записи содержимого, превышающего размер целевого объекта (выход за пределы буфера).
E_INVALID_PARAMETERS	0x10	Любая другая ошибка в параметрах данных.
E_INVALID_MESSAGE_TYPE	0x11	Тип сообщения не приемлем.
E_INVALID_FLAGS	0x12	Неприемлемые для данного типа сообщения флаги.
E_INVALID_TLV	0x13	Блок TLV не приемлем для данного типа сообщения.
E_EVENT_ERROR	0x14	Неуказанная ошибка при обработке события.
E_NOT_SUPPORTED	0x15	Попытка выполнить пригодную операцию ForCES, которую получатель не поддерживает.
E_MEMORY_ERROR	0x16	Ошибка памяти при обработке сообщения (в сообщении ошибок нет).
E_INTERNAL_ERROR	0x17	Неуказанная ошибка при обработке сообщения (в сообщении ошибок нет).
-	0x18-0xFE	Резерв.
E_UNSPECIFIED_ERROR	0xFF	Неуказанная ошибка (FE не понимает, что неверно).

7.1.8. DATA TLV

FULLDATA-TLV имеет T=FULLDATA-TLV, 16 битов поля размера и данные (содержимое). Аналогично, SPARSEDATA-TLV имеет T=SPARSEDATA-TLV, 16-битовое поле length и данные (содержимое). В случае SPARSEDATA-TLV каждая компонента Value будет дополнительно инкапсулирована в ILV.

Ниже приведены правила кодирования FULLDATA-TLV и SPARSEDATA-TLV, примеры правил даны в Приложении C:

1. ILV и TLV **должны** выравниваться по 32-битовым границам. Байты заполнения **должны** устанавливаться в 0 при передаче, а на приёмной стороне **должны** игнорироваться.
2. FULLDATA-TLV могут применяться в качестве определённого пути лишь в том случае, когда присутствует каждая компонента этого уровня пути. В примере 1(c) приложения C эта концепция иллюстрируется наличием всех компонент структуры S в представлении FULLDATA-TLV. Это требование сохраняется для всех полей фиксированного или переменного размера, как обязательных, так и необязательных.
   - При использовании FULLDATA-TLV кодировщик **должен** разместить данные для каждой компоненты в том же порядке, как они были определены в спецификации LFB. Это позволит извлечь данные при декодировании. Для примера 1 в Приложении C это предполагает, что кодер и декодер знают как была определена структура S.
   - В случае SPARSEDATA-TLV упорядочивать данные не требуется, поскольку I в ILV однозначно указывает компоненты. Примеры 1(a) и 1(b) в Приложении C иллюстрируют возможности кодирования SPARSEDATA-TLV.
3. Внутри FULLDATA-TLV.
   - Значения для неделимых полей фиксированного размера задаются без инкапсуляции TLV.
   - Значения для неделимых полей переменного размера задаются внутри FULLDATA-TLV.
   - Значения для массивов задаются в форме index/subscript, за которой следует значение, как сказано в параграфе 7.1.1. Правила упаковки данных и продемонстрировано примерами в приложениях.
4. Внутри SPARSEDATA-TLV.
   - Значения всех полей **должны** быть в форме ILV (32-битовый индекс, 32-битовое поле размера).
5. FULLDATA-TLV не могут включать ILV.
6. Блоки FULLDATA-TLV могут включать в себя другие FULLDATA-TLV для компонент переменного размера. Однозначность декодирования обеспечивается приведённым выше правилом 3.

7.1.9. Взаимосвязи SET и GET

Предполагается, что GET-RESPONSE будет удовлетворять приведённым ниже требованиям.

- Это будет в точности то же определение пути, которое было передано в GET. Единственное различие заключается в том, что GET-RESPONSE будет содержать FULLDATA-TLV.
- Должно быть возможно взять тот же GET-RESPONSE и преобразовать его в SET, просто сменив T в TLV операции.
- Из этого правила имеется два исключения.
  1. При использовании селектора KEY с путём в операции GET этот селектор не возвращается в GET-RESPONSE и вместо него возвращается результат (см. примеры использования KEY).
  2. При выгрузке целой таблицы в GET отклик GET-RESPONSE, который просто меняет T, чтобы стать SET, будет завершать переписывание таблицы.

7.2. Визуализация протокольного кодирования

На рисунке 21 показана общая схема PL PDU. За главным заголовком (main hdr) следует один или множество селекторов LFB, каждый из которых может содержать одну или множество операций.

```
main hdr (Config в данном случае)
      |
      |
      +--- T = LFBselect
      |        |
      |        +-- LFBCLASSID
      |        |
      |        |
      |        +-- LFBInstance
      |        |
      |        +-- T = SET
      |        |   |
      |        |   +--  // одна или множество целей путей
      |        |        // с добавлением здесь их данных
      |        |
      |        +-- T  = DEL
      |        .   |
      |        .   +--  // одна или множество целей путей для удаления
      |
      |
      +--- T = LFBselect
      |        |
      |        +-- LFBCLASSID
      |        |
      |        |
      |        +-- LFBInstance
      |        |
      |        + -- T= SET
      |        |    .
      |        |    .
      |        + -- T= DEL
      |        |    .
      |        |    .
      |        |
      |        + -- T= SET
      |        |    .
      |        |    .
      |
      |
      +--- T = LFBselect
               |
               +-- LFBCLASSID
               |
               +-- LFBInstance
               .
               .
               .
```

Рисунок 21. Логическая схема PL PDU.

На рисунке 22 представлен более подробный пример общей схемы операции внутри целевого выбора LFB. Идея состоит в демонстрации различных уровней вложенности, которые можно пройти для получения целевого пути.

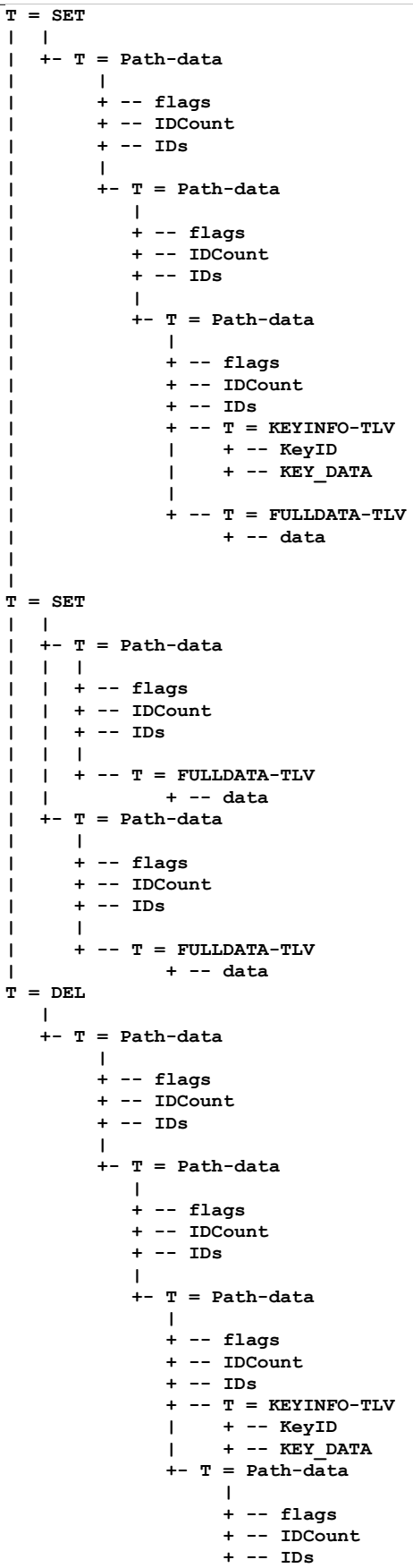

Рисунок 22.Образец схемы операции.

В приложении D показан более выразительный набор примеров кодирования данных.

7.3. Основные LFB протокола ForCES

Имеется два блока LFB, которые применяются для управления работой протокола ForCES, а также взаимодействия с FE и CE:

- FE Protocol LFB;
- FE Object LFB.

Хотя эти LFB имеют такую же форму и интерфейс, как другие LFB, они выделяются во многих отношениях. Они имеют фиксированный общеизвестных LFB Class и Instance ID. Эти блоки определены статически (динамическое создание экземпляров не разрешено), а их статус не может быть изменён протоколом - любые операции по изменению состояния этих LFB (например, запрет LFB) **должны** приводить к ошибке. Кроме того, эти LFB **должны** существовать до того, как будет передано или принято первое сообщение ForCES. Все компоненты этих LFB **должны** иметь предопределённые значения для использования по умолчанию. Наконец, эти LFB не имеют входных и выходных портов и не интегрируются в топологию intra-FE LFB.

7.3.1. FE Protocol LFB

FE Protocol LFB является логическим объектом в каждом FE, который служит для управления протоколом ForCES. Идентификатор класса для FE Protocol LFB имеет значение 0x2, а идентификатор экземпляра - 0x1. В FE **должен** быть один и только один экземпляр FE Protocol LFB. Компоненты FE Protocol LFB имеют предопределённые значения, принятые по умолчанию, которые указаны здесь. Пока значения не заданы явно с помощью сообщений Config от CE, **должны** применяться заданные по умолчанию значения для корректной работы протокола.

Формальное определение FE Protocol Object LFB приведено в Приложении B.

7.3.1.1. Возможности FE Protocol

Возможности FE Protocol доступны только для чтения.

7.3.1.1.1. SupportableVersions

Версии протокола ForCES, поддерживаемые FE.

7.3.1.1.2. Компоненты FE Protocol

Компоненты FE Protocol (могут считываться и устанавливаться).

7.3.1.1.2.1. CurrentRunningVersion

Текущая работающая версия протокола ForCES.

7.3.1.1.2.2. FEID

Индивидуальный идентификатор FE.

7.3.1.1.2.3. MulticastFEIDs

Список групповых идентификаторов FE - идентификаторы групп, в которые входит FE. Идентификаторы задаются CE.

7.3.1.1.2.4. CEHBPolicy

Политика CE для heartbeat. Эта политика вместе с параметром CE Heartbeat Dead Interval (CE HDI), как описано ниже, определяет для FE рабочие параметры проверки живучести CE. Смысл значений политики описан ниже.

- 0 (по умолчанию) - CE будет передавать сообщения Heartbeat элементам FE всякий раз по прошествии интервала в течение которого от CE к FE не было передано других сообщения PL (см. параграфы 4.3.3 и 7.10). Описанная ниже компонента CE HDI привязана к этой политике.
- 1 - CE не будет генерировать сообщения HB. Это означает, что CE не желает, чтобы FE проверяли его живучесть.
- прочие - резерв.

7.3.1.1.2.5. CEHDI

CE Heartbeat Dead Interval (CE HDI) - интервал, используемый FE для проверки живучести CE. Если FE не получил от CE сообщений в течение этого интервала, он считает, что связь потеряна в результате «умирания» CE или его выхода из ассоциации (по умолчанию 30 секунд).

7.3.1.1.2.6. FEHBPolicy

Политика FE для heartbeat, которая вместе с параметром FE Heartbeat Interval (FE HI) определяет рабочие параметры поведения FE, по которым CE может оценить его живучесть. Смысл значений политики описан ниже.

- 0 (по умолчанию) - FE не следует генерировать сообщений Heartbeat. В этом варианте CE отвечает за проверку живучести FE путём установки в заголовке PL флага ACK = AlwaysACK. FE будет отвечать CE при получении от того запросов Heartbeat Request. (см. параграфы 4.3.3 и 7.10).
- 1 - FE **должен** активно передавать сообщения Heartbeat по истечении интервала, заданного FE HI, если не было других сообщений от FE в течение этого интервала, как описано в параграфе 4.3.3.
- прочие - резерв.

7.3.1.1.2.7. FEHI

FE Heartbeat Interval (FE HI) - интервал, по истечении которого FE следует передать сообщение HB, если в течение этого интервала не было передано других сообщений от FE к CE, как описано в параграфе 4.3.3 (по умолчанию 500 с).

7.3.1.1.2.8. CEID

Primary CEID - идентификатор CE, с которым элемент FE связан.

7.3.1.1.2.9. LastCEID

Последний Primary CEID - CEID идентификатор последнего CE, с которым был FE связан. Этот идентификатор CE передаётся новому Primary CEID.

7.3.1.1.2.10. BackupCE

Список резервных CE, которые FE может использовать (см. раздел 8).

7.3.1.1.2.11. CEFailoverPolicy

Политика CE для восстановления при отказах - задаёт поведение FE при потере связи с CE. Политика переключения CE при отказе тесно связана с параграфами 7.3.1.1.2.8, 7.3.1.1.2.10, 7.3.1.1.2.12 и разделом 8. При потере связи активизируется одно из перечисленных ниже правил в зависимости от настройки.

- 0 (по умолчанию) - FE следует незамедлительно прервать работу и перейти в состояние FE OperDisable.
- 1 - FE следует продолжать работу и делать то, что возможно без связи с CE. Обычно это требует от FE поддержки изящного перезапуска CE (и указания такой поддержки в своих возможностях). Если интервал CEFTI истекает до того, как FE восстановит связь с основным CEID (параграф 7.3.1.1.2.8) или одним из резервных CE (параграф 7.3.1.1.2.10), FE будет прекращать работу.
- прочие - резерв.

7.3.1.1.2.12. CEFTI

Интервал ожидания при отказе CE (CEFTI[1]) - время ожидания, связанное с политикой при отказе CE (CEFailoverPolicy) 0 или 1. По умолчанию время ожидания составляет 300 секунд. Отметим, что для CEFTI следует устанавливать значение существенно больше CEHDI, поскольку при этом тайм-ауте FE прекращает работу.

7.3.1.1.2.13. FERestartPolicy

Политика перезапуска FE - задаёт поведение элемента FE при его перезапуске в результате отказа или по иным причинам, требующим перезагрузки. Возможные значения указаны ниже.

- 0 (по умолчанию) - FE перезапускается «с нуля» и в этом случае ему следует начинать с фазы pre-association.
- прочие - резерв на будущее.

7.3.2. FE Object LFB

FE Object LFB представляет собой логический объект, содержащийся в каждом FE и включающий компоненты, относящиеся к самому FE, а не к работе протокола ForCES.

Формальное определение FE Object LFB дано в [RFC5812]. Модель отражает свойства верхнего уровня элемента FE, которые CE нужно знать для начала работы с FE. Идентификатор класса для этого класса LFB также задан в [RFC5812]. Единственный экземпляр этого класса существует всегда и всегда имеет внутри этого класса идентификатор экземпляра 0x1. Общепринято, хотя и не обязательно, получать большую часть информации о компонентах и возможностях из этого экземпляра LFB для CE, когда тот начинает операции управления элементом FE.

7.4. Семантика направления передачи сообщений

Напомним, что PL работает в режиме «ведущий(CE)-ведомый(FE)». LFB размещаются в FE и контролируются CE.

Когда приходит сообщение от CE, селектор LFB (класс и экземпляр) задаёт выбор LFB размещённого в элементе FE.

Когда сообщение от FE приходит в CE, селектор LFB (класс и экземпляр) указывает LFB-источник, находящийся в FE.

7.5. Сообщения для ассоциаций

Сообщения ForCES Association служат для организации и разрыва ассоциаций между элементами FE и CE.

7.5.1. Сообщение Association Setup

Это сообщение передаётся FE элементу CE для организации между ними связи (ассоциации) ForCES.

Направление передачи

От FE к CE.

Заголовок сообщения

Поле MessageType в заголовке имеет значение AssociationSetup. Флаг ACK в заголовке **должен** игнорироваться и всегда предполагается отклик на сообщение Association Setup от получателя (CE), независимо от результата. Поле correlator в заголовке устанавливается так, чтобы FE мог сопоставить сообщение с полученным откликом от CE. FE может установить в заголовке 0 для идентификатора источника, запрашивая тем самым у CE назначение для него FE ID в сообщении Setup Response.

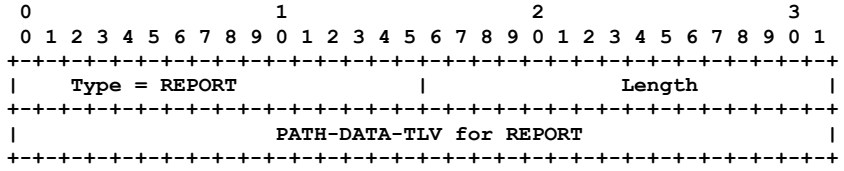

Рисунок 23. OPER-TLV для Association Setup.

Тело сообщения

Тело сообщения Association Setup может включать от 0 до 2 LFBselect TLV, как описано в параграфе 7.1.5. Сообщение Association Setup работает только с FE Object LFB и FE Protocol LFB, поэтому идентификатор класса LFB в LFBselect TLV указывает только эти два типа LFB.

OPER-TLV в LFBselect TLV определяется как операция REPORT. В этом сообщении может быть анонсировано несколько компонент с использованием операции REPORT, чтобы позволить FE заявить свои конфигурационные

[1]CE Failover Timeout Interval.

параметры без запроса. Здесь могут содержаться предлагаемые FE значения типа HB Interval или FEID. Структура OPER-TLV показана ниже.

OPER-TLV для Association Setup

Type

Для сообщений Association Setup определён только один тип операции.

REPORT - задаёт для FE предоставление какого-либо «отчёта» CE.

PATH-DATA-TLV для REPORT

Обычно это формат PATH-DATA-TLV, указанный в разделе 7 в определении PATH-DATA BNF. PATH-DATA-TLV для операции REPORT **может** включать блоки FULLDATA-TLV, **не следует** включать какие-либо RESULT-TLV в данные. RESULT-TLV определён в параграфе 7.1.7, а FULLDATA-TLV - в параграфе 7.1.8.

На рисунке формат PDU показан в виде дерева.

```
main hdr (type =  Association Setup)
     |
     |
     +--- T = LFBselect
     |        |
     |        +-- LFBCLASSID = FE Object
     |        |
     |        |
     |        +-- LFBInstance = 0x1
     |
     +--- T = LFBselect
              |
              +-- LFBCLASSID = FE Protocol Object
              |
              |
              +-- LFBInstance = 0x1
                    |
                    +---OPER-TLV = REPORT
                        |
                        +-- Path-data для одной или множества компонент
```

Рисунок 24. Формат PDU для сообщения Association Setup.

7.5.2. Сообщение Association Setup Response

Это сообщение CE передаёт элементу FE в ответ на сообщение Setup. Оно показывает FE результат попытки создания ассоциации.

Направление передачи

От CE к FE.

Заголовок сообщения

MessageType в заголовке имеет значение AssociationSetupResponse. Флаг ACK в заголовке **должен** игнорироваться и всегда предполагается, что сообщение Setup Response не будет возвращать каких-либо откликов от получателя (FE). Для идентификатора получателя в заголовке устанавливается значение идентификатора источника из соответствующего сообщения Association Setup, если этот идентификатор не был нулевым. В последнем случае CE будет назначать FE ID и указывать его в качестве идентификатора получателя.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Type = ASRresult       |               Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Association Setup Result                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рисунок 25. OPER-TLV для ASResult.

Type (16 битов)

TLV имеет тип ASResult.

Length (16 битов)

Размер TLV с учётом полей T и L (в октетах).

Association Setup result (32 бита)

Это поле показывает результат сообщения Setup - успех или отказ CE. Значения приведены ниже.

0 = успех;

1 = непригодный FE ID;

2 = доступ отвергнут.

Формат PDU показан на рисунке в виде дерева.

```
main hdr (type =  Association Setup Response)
 |
 |
 +--- T = ASResult-TLV
```

Рисунок 26. Формат PDU для сообщения Association Setup Response.

7.5.3. Сообщение Association Teardown

Это сообщение может быть передано FE или CE любому элементу ForCES для завершения ассоциации ForCES с ним.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Type = ASTreason       |               Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Teardown Reason                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Рисунок 27. OPER-TLV для ASTreason.

Направление передачи

От CE к FE, от FE к CE или от CE к CE.

Заголовок сообщения

MessageType в заголовке имеет значение AssociationTeardown. Флаг ACK **должен** игнорироваться. Поле correlator в заголовке **должно** быть установлено в 0 при передаче и **должно** игнорироваться получателем.

Type (16 битов)

TLV имеет тип ASTreason.

Length (16 битов)

Размер TLV с учётом полей T и L (в октетах).

Teardown reason (32 бита)

Указывает причину разрыва ассоциации. Несколько кодов причин показано ниже:

0 - обычное прерывание администратором;
1 - ошибка (потеря heartbeat);
2 - ошибка (нехватка пропускной способности);
3 - ошибка (нехватка памяти);
4 - ошибка (авария приложения);
255 - ошибка (прочее).

Формат PDU в виде дерева представлен на рисунке.

```
main hdr (type =  Association Teardown)
 |
 |
 +--- T = ASTreason-TLV
```

Рисунок 28. Формат PDU для сообщения Association Teardown.

7.6. Конфигурационные сообщения

Сообщения ForCES Configuration служат CE для настройки FE в ForCES NE и возврата результата элементу CE.

7.6.1. Сообщение Config

Это сообщение передаётся от CE к FE для настройки компонент LFB в FE, а также для управления подпиской CE на события LFB.

Как обычно, сообщение Config состоит из общего заголовка, за которым следует тело сообщения с одним или множеством TLV. Подробное описание сообщения приведено ниже.

Направление передачи

От CE к FE.

Заголовок сообщения

MessageType в заголовке имеет значение Config. Флаг ACK в заголовке может иметь любое значение, определённое в параграфе 6.1, и служит для индикации ожидания отклика от FE.

OPER-TLV для Config

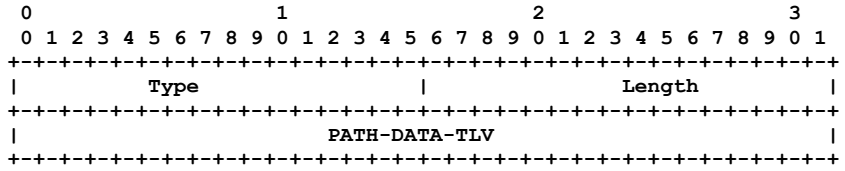

Рисунок 29. OPER-TLV для сообщения Config.

Type

Тип операции для Config. Ниже перечислены пять[1] типов операций, определённых для сообщений Config.

SET - служит для установки компонент LFB.
SET-PROP - служит для установки свойств компонент LFB.
DEL - служит для удаления некоторых компонент LFB.
COMMIT - эта операция передаётся FE для представления транзакции 2PC. COMMIT TLV является пустым (не содержит V) и имеет размер 4 октета (только для заголовка).
TRCOMP - эта операция передаётся FE для указания успеха (с точки зрения NE) транзакции 2PC. TRCOMP TLV является пустым (не содержит V) и имеет размер 4 октета (только для заголовка).

PATH-DATA-TLV

Обычно это формат PATH-DATA-TLV, указанный в разделе 7 в определении PATH-DATA BNF. Ограничение использования PATH-DATA-TLV для операций SET/SET-PROP состоит в том, что этот блок **должен** включать FULLDATA-TLV или SPARSEDATA-TLV, но **недопустимо** включение RESULT-TLV. Ограничение использования PATH-DATA-TLV для операции DEL заключается в том, что блок **может** включать FULLDATA-TLV или SPARSEDATA-TLV, но **недопустимо** включение RESULT-TLV. Блок RESULT-TLV определён в параграфе 7.1.7, а FULLDATA-TLV и SPARSEDATA-TLV - в параграфе 7.1.8.

Примечание. Для подписки на события используются отдельные LFB, определяющие эти события.

Формат PDU в виде дерева представлен на рисунке.

7.6.2. Сообщение Config Response

Это сообщение FE передаёт элементу CE в ответ на сообщение Config. Сообщение показывает результат Config в FE, а также содержит подробные сведения, относящиеся к настройке каждой компоненты.

Направление передачи

От FE к CE.

Заголовок сообщения

MessageType в заголовке имеет значение Config Response. Флаг ACK в заголовке всегда игнорируется и для сообщений Config Response никогда не ожидается отклика со стороны получателя (CE).

OPER-TLV для Config Response

[1] В оригинале ошибочно сказано «два типа». См. https://www.rfc-editor.org/errata/eid5348. *Прим. перев.*

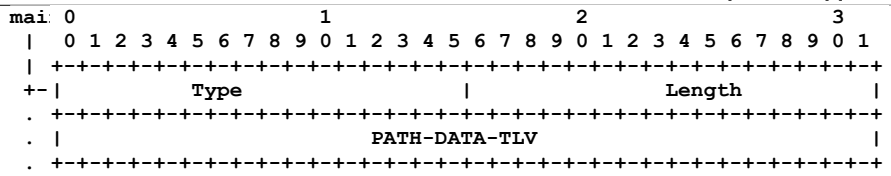

Рисунок 31. OPER-TLV для Config Response.

```
                |
                |
                +-- T = operation { SET }
                |   |
                |   +--  // Одна или множество целей пути
                |        // связанных с FULLDATA-TLV или SPARSEDATA-TLV(s)
                |
                +-- T = operation { DEL }
                |   |
                |   +--  // Одна или множество целей пути
                |
                +-- T = operation { COMMIT } //COMMIT TLV является пустым TLV
                         .
                         .
```

Рисунок 30. Формат PDU для сообщения Configuration.

Type

Тип операции для сообщения Config Response. Ниже перечислены четыре[1] типа операций, определённых для сообщений Config Response.

SET-RESPONSE - для получения отклика от операции SET для компонент LFB.

SET-PROP-RESPONSE - для получения отклика от операции SET SET-PROP для свойств компонент LFB.

DEL-RESPONSE - для получения отклика от операции DELETE для компонент LFB.

COMMIT-RESPONSE - эта операция передаётся элементу CE для подтверждения успешной подачи транзакции 2PC. COMMIT-RESPONSE TLV **должен** включать RESULT-TLV, показывающий успех или отказ.

PATH-DATA-TLV

Обычно это формат PATH-DATA-TLV, указанный в разделе 7 в определении PATH-DATA BNF. Ограничение использования PATH-DATA-TLV для операции SET-RESPONSE состоит в том, что блок **должен** содержать RESULT-TLV. Ограничение использования PATH-DATA-TLV для операции DEL-RESPONSE состоит в том, что блок **должен** содержать RESULT-TLV. Определение RESULT-TLV дано в параграфе 7.1.7.

```
main hdr (type = ConfigResponse)
|
|
+--- T = LFBselect
.         |
.         +-- LFBCLASSID = целевой класс LFB
.         |
          |
          +-- LFBInstance = целевой экземпляр LFB
          |
          |
          +-- T = operation { SET-RESPONSE }
          |   |
          |   +--  // Одна или множество целей пути,
          |        // связанных с FULLDATA-TLV или SPARSEDATA-TLV
          |
          +-- T = operation { DEL-RESPONSE }
          |   |
          |   +--  //  Одна или множество целей пути
          |
          +-- T = operation { COMMIT-RESPONSE }
          |           |
          |           +--  RESULT-TLV
```

Рисунок 32. Формат PDU для сообщения Config Response.

На рисунке 32 формат PDU представлен в виде дерева.

7.7. Запросы

Сообщения ForCES Query используются CE для запроса у LFB в элементе FE информации типа компонент, возможностей, статистики LFB и пр. Группа Query включает сообщения Query и Query Response.

7.7.1. Сообщение Query

Сообщение Query состоит из общего заголовка и тела сообщения, включающего один или множество TLV с данными.

Направление передачи

От CE к FE.

Заголовок сообщения

MessageType в заголовке имеет значение Query. Поле ACK в заголовке игнорируется и для сообщений Query всегда ожидается полный отклик. В заголовке указывается поле Correlator для сопоставления с откликом FE.

OPER-TLV для Query

Type

Один из двух определённых для сообщения типов операции.

GET - запрос компонент LFB.

[1] В оригинале ошибочно сказано «два типа». См. https://www.rfc-editor.org/errata/eid5349. *Прим. перев.*

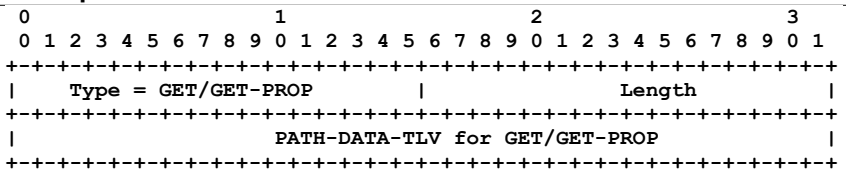

Рисунок 33. OPER-TLV для сообщения Query.

GET-PROP - запрос свойств компонент LFB.

PATH-DATA-TLV для GET/GET-PROP

Обычно это формат PATH-DATA-TLV, указанный в разделе 7 в определении PATH-DATA BNF. Ограничение использования PATH-DATA-TLV для операции GET/GET-PROP состоит в том, что в него **недопустимо** включать SPARSEDATA-TLV или FULLDATA-TLV и RESULT-TLV.

Формат PDU в виде дерева представлен на рисунке.

```
main hdr (type = Query)
 |
 |
 +--- T = LFBselect
 .        |
 .        +-- LFBCLASSID = класс целевого LFB
 .        |
          |
          +-- LFBInstance = экземпляр целевого LFB
          |
          |
          +-- T = operation { GET }
          |   |
          |   +--  // Одна или множество целей пути
          |
          +-- T = operation { GET }
          .   |
          .   +--  // Одна или множество целей пути
          .
```

Рисунок 34. Формат PDU для сообщения Query.

7.7.2. Сообщение Query Response

При получении сообщения Query его следует обработать и определить результат запроса. Получатель сообщения передаёт этот результат отправителю запроса в сообщении Query Response. Результатом может быть запрошенная информация или коды ошибок, если запрос привёл к отказу.

Сообщение Query Response включает общий заголовок и тело сообщения, содержащее один или множество TLV, описывающих результаты запроса.

Направление передачи

От FE к CE.

Заголовок сообщения

MessageType в заголовке имеет значение QueryResponse. Флаг ACK в заголовке игнорируется. Поскольку само сообщение является откликом, оно не требует отклика для себя.

OPER-TLV для Query Response

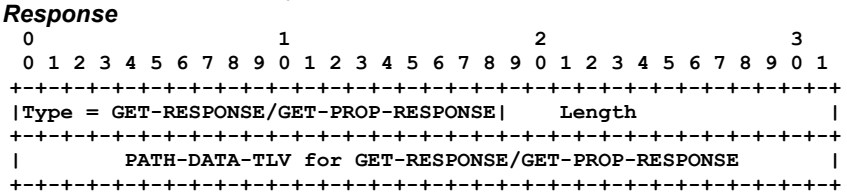

Рисунок 35. OPER-TLV для сообщения Query Response.

Type

Один из двух типов операций, определённых для отклика на запрос:

GET-RESPONSE - отклик на запрос GET для компонент LFB.

GET-PROP-RESPONSE - отклик на запрос GET-PROP для свойств компонент LFB.

PATH-DATA-TLV для GET-RESPONSE/GET-PROP-RESPONSE

Обычно это формат PATH-DATA-TLV, указанный в разделе 7 в определении PATH-DATA BNF. PATH-DATA-TLV для операции GET-RESPONSE **может** содержать SPARSEDATA-TLV, FULLDATA-TLV и/или RESULT-TLV. Блок RESULT-TLV определён в параграфе 7.1.7, а SPARSEDATA-TLV и FULLDATA-TLV - в параграфе 7.1.8.

Формат PDU в виде дерева представлен на рисунке.

7.8. Уведомление о событии

Уведомления о событиях используются FE для асинхронного информирования CE о произошедших в FE событиях.

Все события, которые могут генерироваться в FE являются предметом подписки для CE, который может организовать такую подписку с помощью сообщения Config с операцией SET-PROP, включающей пути, которые указывают события, как определено в библиотеке LFB и описано в модели FE.

Как обычно, сообщение Event Notification состоит из общего заголовка и тела сообщения, включающего один или множество TLV с данными.

Направление передачи

От FE к CE.

Заголовок сообщения

MessageType в заголовке сообщения имеет значение EventNotification. Флаг ACK **должен** игнорироваться CE и для сообщений Event Notification отправитель никогда не ждёт отклика.

```
main hdr (type = QueryResponse)
  |
  |
  +--- T = LFBselect
  .        |
  .        +-- LFBCLASSID = target LFB class
  .        |
           |
           +-- LFBInstance = target LFB instance
           |
           |
           +-- T = operation { GET-RESPONSE }
           |   |
           |   +--  // Одна или множество целей пути
           |
           +-- T = operation { GET-PROP-RESPONSE }
           .   |
           .   +--  // Одна или множество целей пути
           .
```

Рисунок 36. Формат PDU для сообщения Query Response.

OPER-TLV для Event Notification

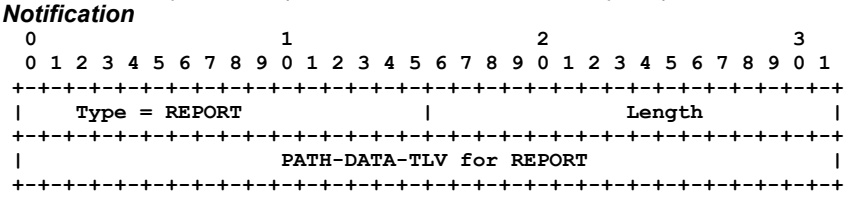

Рисунок 37. TLV для Event Notification.

Type

Для сообщения Event Notification определена единственная операция:

REPORT - для передачи элементом FE информации о том или ином событии.

PATH-DATA-TLV для REPORT

Обычно это формат PATH-DATA-TLV, указанный в разделе 7 в определении PATH-DATA BNF. Блок PATH-DATA-TLV для операции REPORT **может** содержать FULLDATA-TLV или SPARSEDATA-TLV, но **недопустимо** включать в данные какие-либо RESULT-TLV.

Формат PDU в виде дерева представлен на рисунке.

```
main hdr (type = Event Notification)
  |
  |
  +--- T = LFBselect
             |
             +-- LFBCLASSID = target LFB class
             |
             |
             +-- LFBInstance = target LFB instance
             |
             |
             +-- T = operation { REPORT }
             |   |
             |   +--  // Одна или множество целей пути,
             |        // связанных с FULL/SPARSE DATA TLV
             +-- T = operation { REPORT }
             .   |
             .   +--  // Одна или множество целей пути,
             .        // связанных с FULL/SPARSE DATA TLV
```

Рисунок 38. Формат PDU для сообщения Event Notification.

7.9. Сообщение PacketRedirect

Сообщения Redirect используются для передачи пакетов между CE и FE. Обычно эти пакеты содержат данные управления, но они могут быть также пакетами, которым нужна дополнительная обработка (исключение или вышележащие уровни). Возможны также сообщения, содержащие только метаданные.

Направление передачи

От CE к FE или от FE к CE.

Заголовок сообщения

MessageType в заголовке имеет значение PacketRedirect.

Тело сообщения

Сообщение содержит один или множество блоков TLV, которые включают пересылаемый пакет или описывают его. В качестве TLV используются Redirect TLV (Type=Redirect). Формат Redirect TLV показан ниже.

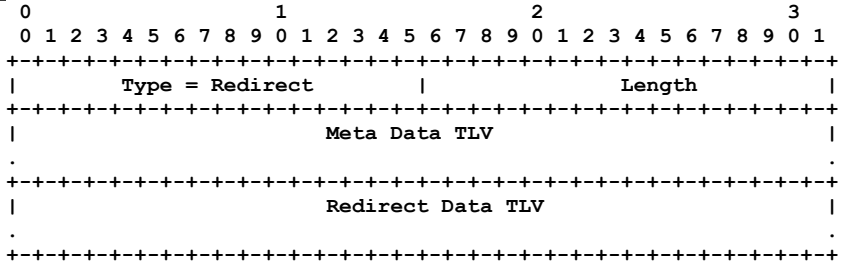

Рисунок 39. Redirect_Data TLV.

TLV метаданных

Этот TLV задаёт метаданные, связанные с расположенными вслед за ними пересылаемыми данными.

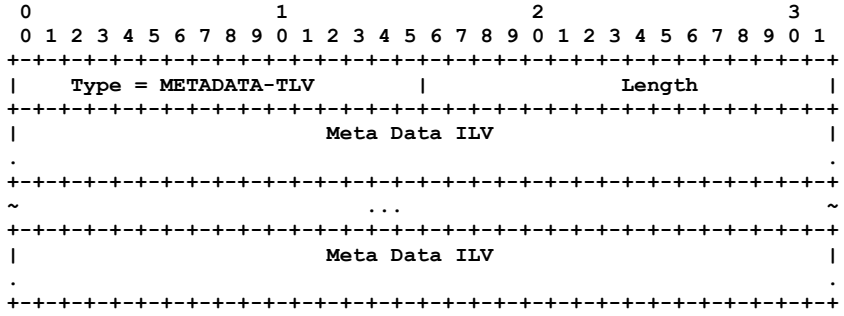

Рисунок 40. METADATA-TLV.

ILV метаданных

Этот формат Identifier-Length-Value используется для описания одного элемента метаданных.

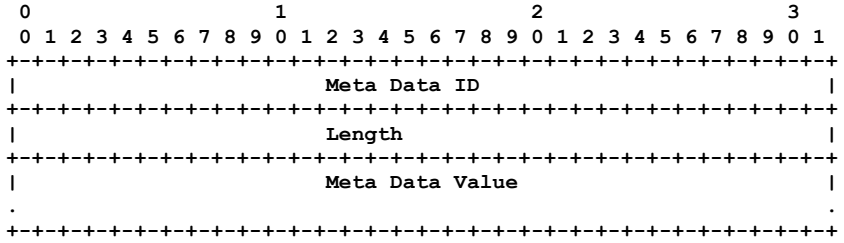

Рисунок 41. ILV для метаданных.

Meta Data ID представляет собой идентификатор метаданных, который статически задаётся в определении LFB.

TLV пересылаемых данных

Этот блок TLV описывает один пакет данных, пересылаемый с помощью операции Redirect.

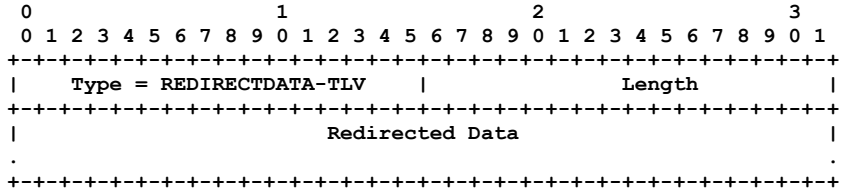

Рисунок 42. REDIRECTDATA-TLV.

Redirected Data

Это поле содержит пакет, который пересылается с использованием сетевого порядка байтов. Пакет следует выравнивать по 32-битовой границе, как и все прочие данные в TLV. Метаданные указывают тип пересылаемого пакета, что упрощает декодирование инкапсулированных данных.

Формат PDU в виде дерева представлен на рисунке.

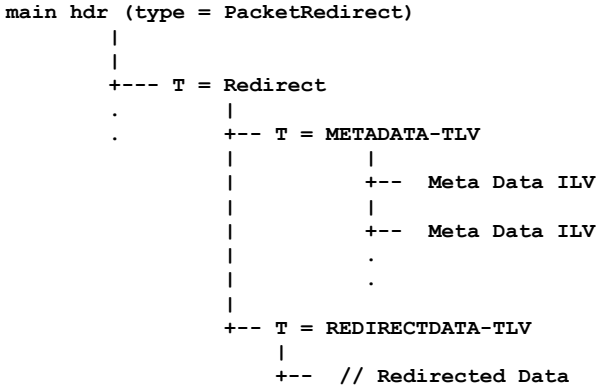

Рисунок 43.Формат PDU для сообщения Packet Redirect.

7.10. Сообщение Heartbeat

Сообщение Heartbeat (HB) используется одним из элементов ForCES (FE или CE) для асинхронного уведомления одного или множества других элементов ForCES в том же ForCES NE о своей живучести. Используемый подход с учётом трафика описан в параграфе 4.3.3.

Сообщения Heartbeat передаются элементами ForCES периодически. Параметризация и правила передачи сообщений heartbeat для FE определяются как компоненты FE Protocol Object LFB и могут быть установлены элементами CE с помощью сообщений Config. Heartbeat слегка отличается от остальных сообщений протокола тем, что оно включает лишь базовый заголовок без тела сообщения.

Направление передачи

От FE к CE или от CE к FE.

Заголовок сообщения

MessageType в заголовке сообщения имеет значение Heartbeat. Используемые механизмы HB описаны в параграфе 4.3.3. Для флага ACK в заголовке **должно** устанавливаться значение NoACK или AlwaysACK.

- NoACK означает, что HB не запрашивает отклик.
- AlwaysACK указывает, что отправитель сообщения HB всегда ждёт отклика от получателя. По правилам, приведённым в параграфе 7.3.1, только CE может передавать сообщения HB для проверки живучести FE. Для простоты и по причине минималистической природы HB откликов на сообщение HB служит другое сообщение HB, т. е. специального сообщения HB Response не определено. Когда FE получает сообщение HB с флагом AlwaysACK от элемента CE, этот FE **должен** незамедлительно передать сообщение HB. В сообщении HB, переданном FE в качестве отклика на AlwaysACK, идентификаторы отправителя и получателя **должны** поменяться местами, чтобы идентификатор FE указывал источник, а CE ID - получателя, а флаг ACK должен быть заменён на NoACK. Элементам CE **недопустимо** отвечать на сообщения HB с флагом AlwaysACK.
- При установке флагов, отличных от NoACK и AlwaysACK, будет считаться, что HB имеет флаг NoACK.

Поле correlator в заголовке HB **следует** устанавливать так, чтобы отправитель мог сопоставить отклик с запросом. Поле correlator **можно** игнорировать, если отклика не ожидается.

Тело сообщения

Тело сообщения Heartbeat пусто.

8. Поддержка высокого уровня доступности

Протокол ForCES обеспечивает механизмы резервирования CE и восстановления при отказах для поддержки высокого уровня доступности (High Availability или HA) в соответствии с [RFC3654]. Резервирование FE и взаимодействие между элементами FE выходит за рамки этого документа. В одном элементе ForCES NE может присутствовать множество избыточных CE и FE. Однако в любой момент только один первичный элемент CE может управлять FE, хотя допускается множество вторичных CE. Уровень протокола (PL) в FE и CE знает о первичном и вторичных CE. Эта информация (первичный и вторичные CE) настраивается в PL элементов FE и CE в процессе создания ассоциации (pre-association) с помощью менеджеров FEM и CEM, соответственно. Управляющие FE сообщения передаёт только первичный CE.

8.1. Связь с FE Protocol

Параметризация HA в FE выполняется путём настройки FE Protocol Object LFB (см. Приложение B и параграф 7.3.1). Параметры FE Heartbeat Interval, CE Heartbeat Dead Interval и правила CE Heartbeat помогают обнаруживать проблемы связи между FE и CE. Политика CE для восстановления при отказах определяет реакцию на обнаружение отказа.

На рисунке 44 показана машина состояний с учётом новых состояний, упрощающих восстановление соединений.

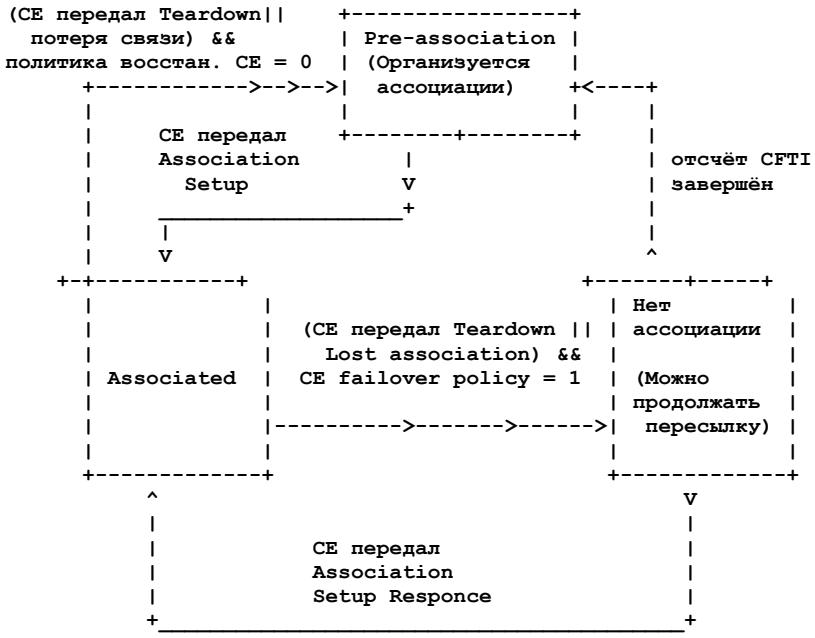

Рисунок 44. Конечный автомат FE с учетом HA.

Переходы между состояниями до ассоциации, а также при наличии и отсутствии связи описаны в параграфе 4.2.

При отказе коммуникаций между FE и CE (который связан с CE или каналом, но не с FE) TML в элементе FE укажет FE PL на этот отказ или проблема будет обнаружена с помощью сообщений HB между FE и CE. Коммуникационный отказ, независимо от способа его обнаружения, **должен** считаться потерей ассоциации между CE и соответствующим FE.

Если для политики восстановления в FE (FEPO) установлено значение 0 (принято по умолчанию), элемент незамедлительно перейдёт в фазу pre-association. Это означает, что при восстановлении связи все состояния FE придётся организовывать заново.

Если для политики восстановления в FE (FEPO) установлено значение 1, это говорит о способности FE к перезапуску в режиму HA. В таком случае FE переходит в состояние отсутствия связи и запускается таймер CEFTI. Элемент FE **может** продолжать пересылку пакетов, находясь в этом состоянии. Он **может** также переключиться на любой из настроенных вторичных CE, перебирая из по кругу. Сначала он добавляет свой первичный CE в конец списка резервных CE и устанавливает основным первый из списка резервных CE. После этого предпринимается попытка организовать ассоциацию с новым первичным CE. Если в течение отсчёта таймера CEFTI не удалось организовать ассоциацию ни с одним из вторичных CE, элемент FE переходит в состояние pre-association.

Если FE, находясь вне ассоциации, удаётся соединиться с новым первичным CE до завершения отсчёта CEFTI, он переходит в связанное состояние. После восстановления связи FE пытается восстановить состояние, которое могло быть потеряно за время отсутствия связи. Способы этого восстановления выходят за рамки документа.

На рисунке 45 показана последовательность сообщений ForCES, применяемых FE при попытке восстановить связь.

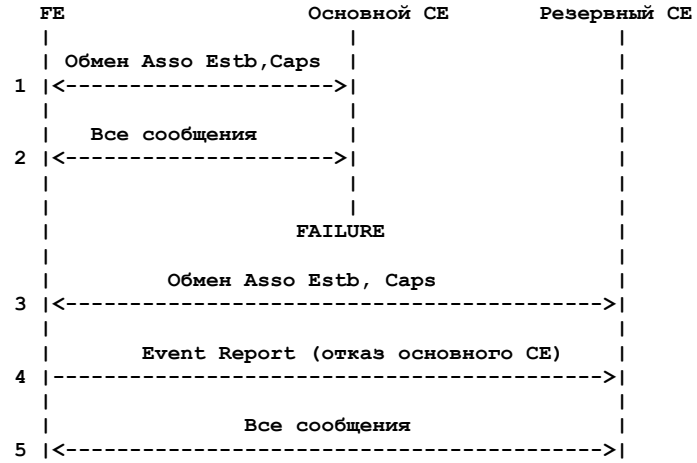

Рисунок 45. Переключение CE в режиме Report Primary.

Требуется протокол синхронизации элементов CE для поддержки быстрого восстановления при отказах, а также для решения некоторых других вопросов, однако это не входит в сферу протокола ForCES и не рассматривается в данном документе.

Явное сообщение (Config с установкой компоненты первичного CE в FE Protocol Object) от первичного CE также может служить для смены первичного CE в процессе обычной работы FE.

Отметим, что элементы FE в ForCES NE могут также использовать групповой идентификатор CE ID, т. е. они могут быть связаны с группой CE (это предполагает использование протокола синхронизации между CE, который выходит за рамки этой спецификации). В таких случаях потеря ассоциации будет означать потерю связи со всей группой CE. Описанные выше механизмы также применимы для этого случая. Однако, если вторичный CE тоже использует групповой идентификатор потерянной связи, FE придётся создавать новую ассоциацию с другим CE ID. При наличии такой возможности FE **может** сначала предпринять попытку создать новую ассоциацию с исходным первичным CE, используя другой (не групповой) идентификатор CE ID.

8.2. Ответственность за HA

Уровень TML

1. TML контролирует доступность логический соединений и восстановление при отказах.
2. TML также контролирует управление партнерским HA.

На этом уровне роль TML заключается в контроле всех нижележащих уровней (например, адресов IP и MAC), а также соответствующих каналов.

Уровень PL

Все остальные функции, включая настройку поведения HA в процессе установки, идентификаторы CE для указания первичных и вторичных элементов CE, протокольные сообщения для уведомления об отказе (Event Report), сообщения Heartbeat для обнаружения отказов, сообщения для смены первичного CE (Config) и другие операции, связанные с HA, лежат в сфере ответственности уровня PL.

При недоступности пути к первичному CE уровень TML будет принимать меры по переходу на резервный путь, если это возможно. Если CE совсем не доступен, уровень PL будет уведомлен об этом и предпримет подобающие действия, описанные выше.

9. Вопросы безопасности

Раздел 8 рамочного документа ForCES [RFC3746] включает подробные сведения о разных угрозах безопасности, возможном влиянии этих угроз на протокол и возможных ответах на угрозы. Данный документ не повторяет этого рассмотрения, а просто рекомендует читателю обратиться к описанию архитектуры ForCES [RFC3746].

Уровень ForCES PL использует услуги защиты, предоставляемые уровнем ForCES TML, который обеспечивает проверку подлинности конечных точек и сообщений, а также защиту конфиденциальности. Служба аутентификации конечных точек вызывается на этапе организации соединения (фаза pre-association), тогда как аутентификация сообщений используется при получении элементом FE или CE каждого сообщения от своего партнёра.

Ниже перечислены базовые механизмы защиты, которые требуются для ForCES PL.

- Управление безопасностью на уровне сессии - после организации выбранного уровня защиты (No Security, Authentication, Confidentiality) это будет действовать в течение всего сеанса работы.
- Оператору следует настраивать одинаковые правила безопасности для основных и резервных элементов FE и CE (если они есть). Это обеспечит однородность операций и избавит от неоправданных сложностей.

9.1. Без защиты

При выборе уровня No Security (без защиты) для коммуникаций протокола ForCES проверку подлинности конечных точек и сообщений должен выполнять уровень ForCES PL. Его механизмы достаточно слабы и не включают криптографических операций. Оператор может выбрать режим No Security, когда конечные точки протокола размещаются, например, в одном устройстве.

Для обеспечения взаимодействия и однотипной реализации уровней защиты, каждый элемент CE и FE **должен** поддерживать этот уровень.

Описанное в параграфах 9.1.1 и 9.1.2 является лишь проверкой ошибок, а не процедурами защиты. Защита описана в параграфе 9.2.

9.1.1. Проверка подлинности конечных точек

В каждом CE и FE уровень PL поддерживает список ассоциаций как часть своей конфигурации. Это выполняется через интерфейсы CEM и FEM. FE **должен** подключаться только к CE, заданным через FEM, а CE следует принимать соединения и создавать ассоциации с элементами FE, заданными через CEM. Элементу CE следует проверять идентификатор FE прежде, чем принять соединение в фазе pre-association.

9.1.2. Проверка подлинности сообщений

Когда CE или FE передаёт сообщение, принимающая сторона **должна** проверить инициатора по идентификатору CE или FE в базовом заголовке. Это обеспечит корректную работу протокола. Такая дополнительная проверка рекомендуется даже при использовании услуг защиты базового уровня TML.

9.2. ForCES PL и защитные услуги TML

Этот раздел применим в тех случаях, когда оператор решил использовать защитные услуги TML. Уровень ForCES TML **должен** поддерживать одну или множество защитных служб типа проверки подлинности конечных точек и сообщений, а также защиты конфиденциальности, как часть функций защиты уровня TML. Выбор подходящего уровня защиты и настройка политики безопасности определяются оператором. Детали настройки выходят за рамки ForCES PL и зависят от типа транспортного протокола и природы соединений.

Все настройки конфигурации должны быть выполнены до начала работы CE и FE.

При использовании в TML аутентификации на основе сертификатов эти сертификаты могут использовать связанную с ForCES структуру именования для задания имён и соответствующих правил безопасности в настройках CE и FE.

Детали, относящиеся к требованиям защиты TML, читателю следует искать в документах соответствующего TML.

9.2.1. Проверка подлинности конечных точек

При включённой защите TML уровень ForCES TML выполняет аутентификацию конечных точек. Защищённые связи организуются между элементами CE и FE и прозрачны для уровня ForCES PL.

9.2.2. Проверка подлинности сообщений

Относящиеся к TML операции прозрачны для ForCES PL (см. раздел 5).

9.2.3. Конфиденциальность

Относящиеся к TML операции прозрачны для ForCES PL (см. раздел 5).

10. Благодарности

Авторы документа выражают свою признательность и благодарности членам рабочей группы ForCES, особо отметив вклад Furquan Ansari, Alex Audu, Steven Blake, Shuchi Chawla, Alan DeKok, Ellen M. Deleganes, Xiaoyi Guo, Yunfei Guo, Evangelos Haleplidis, Zsolt Haraszti, Fenggen Jia, John C. Lin, Alistair Munro, Jeff Pickering, T. Sridhlar, Guangming Wang, Chaoping Wu и Lily L. Yang. Спасибо также David Putzolu и Patrick Droz за их комментарии и предложения по протоколу, а также за бесконечное терпение. Спасибо Sue Hares и Alia Atlas за рецензирование документа.

Alia Atlas проделала большую работу по улучшению читаемости документа, обеспечив отклики IESG. Ross Callon сыграл важную роль по преодолению основных препятствий для публикации этого документа.

Редакторы использовали инструменты xml2rfc [RFC2629] для подготовки этого документа и очень признательны этим инструментам за их качество. Спасибо также Elwyn Davies за помощь в корректировке XML для этого документа.

11. Литература

11.1. Нормативные документы

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5390] Rosenberg, J., "Requirements for Management of Overload in the Session Initiation Protocol", RFC 5390, December 2008.

[RFC5811] Hadi Salim, J. and K. Ogawa, "SCTP-Based Transport Mapping Layer (TML) for the Forwarding and Control Element Separation (ForCES) Protocol", RFC 5811, March 2010.

[RFC5812] Halpern, J. and J. Hadi Salim, "Forwarding and Control Element Separation (ForCES) Forwarding Element Model", RFC 5812, March 2010.

11.2. Дополнительная литература

[2PCREF] Gray, J., "Notes on database operating systems", in "Operating Systems: An Advanced Course" Lecture Notes in Computer Science, Vol. 60, pp. 394-481, Springer-Verlag, 1978.

[ACID] Haerder, T. and A. Reuter, "Principles of Transaction-Orientated Database Recovery", 1983.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC3654] Khosravi, H. and T. Anderson, "Requirements for Separation of IP Control and Forwarding", RFC 3654, November 2003.

[RFC3746] Yang, L., Dantu, R., Anderson, T., and R. Gopal, "Forwarding and Control Element Separation (ForCES) Framework", RFC 3746, April 2004.

Приложение A. Взаимодействие с IANA

В соответствии с правилами «Guidelines for Writing an IANA Considerations Section in RFCs» (RFC 5226 [RFC5226]) для ForCES определены пространства имён:

- Message Type Namespace, раздел 7;
- Operation Type Namespace, параграф 7.1.6;
- Header Flags, параграф 6.1;
- TLV Type, раздел 7;
- RESULT-TLV[1] Result Values, параграф 7.1.7;
- LFB Class ID, параграф 7.1.5 (см. [RFC5812]);
- Result: Association Setup Response, параграф 7.5.2;
- Reason: Association Teardown Message, параграф 7.5.3.

A.1. Пространство типов сообщений

Для типов сообщений используются 8-битовые значения. Ниже приведены рекомендации по распределению типов.

0x00 - 0x1F

Типы из этого диапазона являются частью базового протокола ForCES и выделяются по согласованию с IETF [RFC5226].

Значения, заданные этой спецификацией

0x00	Reserved
0x01	AssociationSetup
0x02	AssociationTeardown
0x03	Config
0x04	Query
0x05	EventNotification
0x06	PacketRedirect
0x07 - 0x0E	Reserved
0x0F	Hearbeat
0x11	AssociationSetupResponse
0x12	Reserved
0x13	ConfigResponse
0x14	QueryResponse

0x20 - 0x7F

Типы из этого диапазона выделяются по процедуре Specification Required [RFC5226] и **должны** быть документированы в RFC или других доступных для чтения документах.

0x80 - 0xFF

Типы из этого диапазона зарезервированы для фирменных расширений и назначаются производителями. Взаимодействие с IANA для значений типов из этого диапазона не требуется.

A.2. Выбор операции

Для Operation Selection (OPER-TLV) используются 16-битовые значения. Ниже приведены рекомендации по распределению типов OPER-TLV.

0x0000-0x0FF

Типы OPER-TLV из этого диапазона выделяются по согласованию с IETF [RFC5226].

Значения, заданные этой спецификацией

0x0000	Reserved
0x0001	SET

[1] В оригинале допущена ошибка, см. https://www.rfc-editor.org/errata/eid2566. *Прим. перев.*

0x0002	SET-PROP
0x0003	SET-RESPONSE
0x0004	SET-PROP-RESPONSE
0x0005	DEL
0x0006	DEL-RESPONSE
0x0007	GET
0x0008	GET-PROP
0x0009	GET-RESPONSE
0x000A	GET-PROP-RESPONSE
0x000B	REPORT
0x000C	COMMIT
0x000D	COMMIT-RESPONSE
0x000E	TRCOMP

0x0100-0x7FFF

Типы OPER-TLV из этого диапазона **должны** быть документированы в RFC или других доступных для чтения документах [RFC5226].

0x8000-0xFFFF

Типы OPER-TLV из этого диапазона зарезервированы для фирменных расширений и назначаются производителями. Взаимодействие с IANA для значений типов из этого диапазона не требуется.

A.3. Флаги заголовка

Поле флагов в заголовке имеет размер 32 бита. Поле flags является частью базового протокола ForCES. Значения флагов выделяются по согласованию с IETF [RFC5226].

A.4. Типы TLV

Для типов TLV используются 16-битовые значения. Ниже приведены рекомендации по распределению типов TLV.

0x0000-0x01FF

Типы TLV из этого диапазона выделяются по согласованию с IETF [RFC5226].

Значения, заданные этой спецификацией

0x0000	Reserved
0x0001	REDIRECT-TLV
0x0010	ASResult-TLV
0x0011	ASTreason-TLV
0x1000	LFBselect-TLV
0x0110	PATH-DATA-TLV
0x0111	KEYINFO-TLV
0x0112	FULLDATA-TLV
0x0113	SPARSEDATA-TLV
0x0114	RESULT-TLV
0x0115	METADATA-TLV
0x0116	REDIRECTDATA-TLV

0x0200-0x7FFF

Типы TLV из этого диапазона **должны** быть документированы в RFC или других доступных для чтения документах [RFC5226].

0x8000-0xFFFF

Типы TLV из этого диапазона зарезервированы для фирменных расширений и назначаются производителями. Взаимодействие с IANA для значений типов из этого диапазона не требуется.

A.5. Коды результата в RESULT-TLV

Значения RTesult RESULT-TLV являются 8-битовыми.

0x00	E_SUCCESS
0x01	E_INVALID_HEADER
0x02	E_LENGTH_MISMATCH
0x03	E_VERSION_MISMATCH
0x04	E_INVALID_DESTINATION_PID
0x05	E_LFB_UNKNOWN
0x06	E_LFB_NOT_FOUND
0x07	E_LFB_INSTANCE_ID_NOT_FOUND
0x08	E_INVALID_PATH
0x09	E_COMPONENT_DOES_NOT_EXIST
0x0A	E_EXISTS
0x0B	E_NOT_FOUND
0x0C	E_READ_ONLY
0x0D	E_INVALID_ARRAY_CREATION
0x0E	E_VALUE_OUT_OF_RANGE
0x0F	E_CONTENTS_TOO_LONG
0x10	E_INVALID_PARAMETERS
0x11	E_INVALID_MESSAGE_TYPE
0x12	E_INVALID_FLAGS[1]
0x13	E_INVALID_TLV
0x14	E_EVENT_ERROR
0x15	E_NOT_SUPPORTED
0x16	E_MEMORY_ERROR
0x17	E_INTERNAL_ERROR
0x18-0xFE	Reserved
0xFF	E_UNSPECIFIED_ERROR

Все значения, не назначенные этой спецификацией, распределяются по процедуре Expert Review [RFC5226].

[1]В оригинале ошибочно сказано E_E_INVALID_FLAGS. См. https://www.rfc-editor.org/errata/eid4188. *Прим. перев.*

A.6. Отклик Association Setup

Для откликов Association Setup используются 32-битовые значения. Рекомендации по распределению приведены ниже.

0x00000000-0x000000FF[1]

Значения Association Setup Response из этого диапазона выделяются по согласованию с IETF [RFC5226].

Значения, заданные этой спецификацией[1]

```
0x0000000   Success
0x0000001   FE ID Invalid
0x0000002   Permission Denied
```

0x00000100-0x00000FFF[1]

Значения Association Setup Response из этого диапазона выделяются по процедуре Specification Required [RFC5226] и **должны** быть документированы в RFC или других доступных для чтения документах [RFC5226].

0x00001000-0xFFFFFFFF[1]

Значения из этого диапазона зарезервированы для фирменных расширений и назначаются производителями. Взаимодействие с IANA для значений типов из этого диапазона не требуется.

A.7. Сообщение Association Teardown

Для сообщений Association Teardown используются 32-битовые значения. Рекомендации по распределению приведены ниже.

0x00000000-0x0000FFFF

Значения Association Teardown Message из этого диапазона выделяются по согласованию с IETF [RFC5226].

Значения, заданные этой спецификацией

```
0x00000000        Normal - teardown by administrator
0x00000001        Error  - loss of heartbeats
0x00000002        Error  - loss of bandwidth
0x00000003        Error  - out of Memory
0x00000004        Error  - application crash
0x000000FF        Error  - unspecified
```

0x00010000-0x7FFFFFFF

Коды Association Teardown Message из этого диапазона выделяются по процедуре Specification Required [RFC5226] и **должны** быть документированы в RFC или других доступных для чтения документах [RFC5226].

0x80000000-0xFFFFFFFF

Значения из этого диапазона зарезервированы для фирменных расширений и назначаются производителями. Взаимодействие с IANA для значений типов из этого диапазона не требуется.

Приложение B. Схема LFB протокола ForCES

Приведённая ниже схема соответствует схеме LFB, описанной в модели ForCES [RFC5812].

Описания различных компонент этого определения приведены в параграфе 7.3.1.

```
<LFBLibrary xmlns="urn:ietf:params:xml:ns:forces:lfbmodel:1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    provides="FEPO">
<!-- XXX  -->
  <dataTypeDefs>
     <dataTypeDef>
        <name>CEHBPolicyValues</name>
              <synopsis>
                  Возможные значения политики CE для heartbeat
              </synopsis>
           <atomic>
           <baseType>uchar</baseType>
           <specialValues>
              <specialValue value="0">
                <name>CEHBPolicy0</name>
                <synopsis>
                     Политика CE для heartbeat, номер 0
                </synopsis>
                </specialValue>
              <specialValue value="1">
                 <name>CEHBPolicy1</name>
                 <synopsis>
                      Политика CE для heartbeat, номер 1
                 </synopsis>
              </specialValue>
            </specialValues>
            </atomic>
      </dataTypeDef>

      <dataTypeDef>
         <name>FEHBPolicyValues</name>
              <synopsis>
                  Возможные значения политики FE для heartbeat
            </synopsis>
           <atomic>
           <baseType>uchar</baseType>
           <specialValues>
             <specialValue value="0">
               <name>FEHBPolicy0</name>
               <synopsis>
                    Политика FE для heartbeat, номер 0
```

[1] В оригинале ошибочно указаны 16-битовые значения. См. https://www.rfc-editor.org/errata/eid2568. *Прим. перев.*

```
                </synopsis>
              </specialValue>
              <specialValue value="1">
                 <name>FEHBPolicy1</name>
                 <synopsis>
                      Политика FE для heartbeat, номер 1
                 </synopsis>
                </specialValue>
            </specialValues>
            </atomic>
      </dataTypeDef>

      <dataTypeDef>
      <name>FERestartPolicyValues</name>
            <synopsis>
                 Возможные значения политики FE для перезапуска
            </synopsis>
           <atomic>
           <baseType>uchar</baseType>
           <specialValues>
              <specialValue value="0">
                <name>FERestartPolicy0</name>
                <synopsis>
                     Политика FE для перезапуска, номер 0
                </synopsis>
                </specialValue>
            </specialValues>
            </atomic>
      </dataTypeDef>

      <dataTypeDef>
      <name>CEFailoverPolicyValues</name>
            <synopsis>
                Возможные значения политики CE для восстановления при отказах
            </synopsis>
           <atomic>
           <baseType>uchar</baseType>
           <specialValues>
             <specialValue value="0">
                <name>CEFailoverPolicy0</name>
                <synopsis>
                     Политика CE для восстановления при отказах, номер 0
                </synopsis>
              </specialValue>
            <specialValue value="1">
               <name>CEFailoverPolicy1</name>
               <synopsis>
                    Политика CE для восстановления при отказах, номер 1
               </synopsis>
             </specialValue>
            </specialValues>
            </atomic>
      </dataTypeDef>

     <dataTypeDef>
        <name>FEHACapab</name>
               <synopsis>Поддерживаемые функции HA</synopsis>
           <atomic>
           <baseType>uchar</baseType>
           <specialValues>
              <specialValue value="0">
                <name>GracefullRestart</name>
                <synopsis>
                     FE поддерживает Graceful Restart
                </synopsis>
                </specialValue>
              <specialValue value="1">
                 <name>HA</name>
                 <synopsis>
                      FE поддерживает HA
                 </synopsis>
              </specialValue>
            </specialValues>
            </atomic>
      </dataTypeDef>
  </dataTypeDefs>

  <LFBClassDefs>
    <LFBClassDef LFBClassID="2">
      <name>FEPO</name>
      <synopsis>
         FE Protocol Object
      </synopsis>
      <version>1.0</version>

  <components>
```

```
<component componentID="1" access="read-only">
    <name>CurrentRunningVersion</name>
    <synopsis>Версия используемого в данный момент протокола ForCES</synopsis>
    <typeRef>uchar</typeRef>
  </component>
<component componentID="2" access="read-only">
  <name>FEID</name>
  <synopsis>Индивидуальный FEID</synopsis>
  <typeRef>uint32</typeRef>
</component>
<component componentID="3" access="read-write">
   <name>MulticastFEIDs</name>
   <synopsis>
      Таблица всех групповых идентификаторов
   </synopsis>
   <array type="variable-size">
    <typeRef>uint32</typeRef>
   </array>
</component>
<component componentID="4" access="read-write">
  <name>CEHBPolicy</name>
  <synopsis>
   Политика CE для Heartbeat
  </synopsis>
  <typeRef>CEHBPolicyValues</typeRef>
</component>
<component componentID="5" access="read-write">
  <name>CEHDI</name>
  <synopsis>
    Интервал CE Heartbeat Dead в миллисекундах
  </synopsis>
  <typeRef>uint32</typeRef>
</component>
<component componentID="6" access="read-write">
  <name>FEHBPolicy</name>
  <synopsis>
    Политика FE для Heartbeat
  </synopsis>
  <typeRef>FEHBPolicyValues</typeRef>
</component>
<component componentID="7" access="read-write">
  <name>FEHI</name>
  <synopsis>
    Интервал FE Heartbeat в миллисекундах
  </synopsis>
  <typeRef>uint32</typeRef>
</component>
<component componentID="8" access="read-write">
  <name>CEID</name>
  <synopsis>
     Первичный CE, с которым связан этот FE
  </synopsis>
  <typeRef>uint32</typeRef>
</component>
<component componentID="9" access="read-write">
   <name>BackupCEs</name>
   <synopsis>
      Таблица всех резервных CE за исключением первичного
   </synopsis>
   <array type="variable-size">
    <typeRef>uint32</typeRef>
   </array>
</component>
<component componentID="10" access="read-write">
  <name>CEFailoverPolicy</name>
  <synopsis>
    Политика CE для восстановления при отказах
  </synopsis>
  <typeRef>CEFailoverPolicyValues</typeRef>
</component>
<component componentID="11" access="read-write">
  <name>CEFTI</name>
  <synopsis>
    Интервал CE Failover Timeout в миллисекундах
  </synopsis>
  <typeRef>uint32</typeRef>
</component>
<component componentID="12" access="read-write">
  <name>FERestartPolicy</name>
  <synopsis>
     Политика перезапуска FE
  </synopsis>
  <typeRef>FERestartPolicyValues</typeRef>
</component>
<component componentID="13" access="read-write">
  <name>LastCEID</name>
```

```
            <synopsis>
               Первичный CE с которым этот FE был связан в последний раз
            </synopsis>
            <typeRef>uint32</typeRef>
          </component>
        </components>

       <capabilities>
            <capability componentID="30">
               <name>SupportableVersions</name>
               <synopsis>
                  Таблица версий ForCES, поддерживаемых FE
               </synopsis>
               <array type="variable-size">
                <typeRef>uchar</typeRef>
               </array>
             </capability>
          <capability componentID="31">
             <name>HACapabilities</name>
             <synopsis>
                Таблица возможностей HA, поддерживаемых FE
             </synopsis>
             <array type="variable-size">
              <typeRef>FEHACapab</typeRef>
             </array>
          </capability>
        </capabilities>

        <events baseID="61">
          <event eventID="1">
            <name>PrimaryCEDown</name>
            <synopsis>Первичный CE был изменён</synopsis>
            <eventTarget>
                <eventField>LastCEID</eventField>
            </eventTarget>
            <eventChanged/>
            <eventReports>
               <eventReport>
                 <eventField>LastCEID</eventField>
               </eventReport>
            </eventReports>
          </event>
        </events>
      </LFBClassDef>
    </LFBClassDefs>
  </LFBLibrary>
```

B.1. Возможности

SupportableVersions содержит перечисление всех версий ForCES, которые поддерживает элемент FE.

FEHACapab содержит перечисление всех возможностей HA в элементе FE. Если FE не поддерживает изящный перезапуск или HA, он не сможет участвовать в HA, как описано в параграфе 8.1.

B.2. Компоненты

Все компоненты описаны в параграфе 7.3.1.

Приложение C. Примеры кодирования данных

В этом приложении обсуждаются несколько примеров кодирования данных. Заполнение в примерах не показано.

Пример 1

Структура с тремя обязательными полями фиксированного размера

```
struct S {
    uint16 a
    uint16 b
    uint16 c
}
```

(a) Описание всех полей, использующих SPARSEDATA-TLV

```
PATH-DATA-TLV
  Путь к экземпляру структуры S ...
  SPARSEDATA-TLV
    ComponentIDof(a), lengthof(a), valueof(a)
    ComponentIDof(b), lengthof(b), valueof(b)
    ComponentIDof(c), lengthof(c), valueof(c)
```

(b) Описание подмножества полей

```
PATH-DATA-TLV
  Путь к экземпляру структуры S ...
  SPARSEDATA-TLV
    ComponentIDof(a), lengthof(a), valueof(a)
    ComponentIDof(c), lengthof(c), valueof(c)
```

Примечание. Несмотря на присутствие в структуре S обязательных компонент, однозначная идентификация компонент позволяет селективно передавать компоненты структуры S (например, для обновления от CE к FE).

(c) Описание всех полей, использующих FULLDATA-TLV

```
PATH-DATA-TLV
```

```
  Путь к экземпляру структуры S ...
  FULLDATA-TLV
    valueof(a)
    valueof(b)
    valueof(c)
```

Пример 2

Структура с тремя полями фиксированного размера, два из которых не обязательны.

```
struct T {
    uint16 a
    uint16 b (необязательно)
    uint16 c (необязательно)
}
```

Этот пример идентичен примеру 1, как показано ниже.

(a) Описание всех полей, использующих SPARSEDATA-TLV

```
PATH-DATA-TLV
  Путь к экземпляру структуры S ...
  SPARSEDATA-TLV
    ComponentIDof(a), lengthof(a), valueof(a)
    ComponentIDof(b), lengthof(b), valueof(b)
    ComponentIDof(c), lengthof(c), valueof(c)
```

(b) Описание подмножества полей, использующих SPARSEDATA-TLV

```
PATH-DATA-TLV
  Путь к экземпляру структуры S ...
  SPARSEDATA-TLV
    ComponentIDof(a), lengthof(a), valueof(a)
    ComponentIDof(c), lengthof(c), valueof(c)
```

(c) Описание всех полей, использующих FULLDATA-TLV

```
PATH-DATA-TLV
  Путь к экземпляру структуры S ...
  FULLDATA-TLV
    valueof(a)
    valueof(b)
    valueof(c)
```

Примечание. FULLDATA-TLV **не может** использоваться, пока не описаны все поля.

Пример 3

Структура с полями фиксированного и переменного размера, часть которых не обязательна. Переменный размер имеет строка b.

```
struct U {
uint16 a
string b (optional)
uint16 c (optional)
}
```

(a) Описание всех полей, использующих SPARSEDATA-TLV

```
Путь к экземпляру структуры U ...
SPARSEDATA-TLV
  ComponentIDof(a), lengthof(a), valueof(a)
  ComponentIDof(b), lengthof(b), valueof(b)
  ComponentIDof(c), lengthof(c), valueof(c)
```

(b) Описание подмножества полей, использующих SPARSEDATA-TLV

```
Путь к экземпляру структуры U ...
SPARSEDATA-TLV
  ComponentIDof(a), lengthof(a), valueof(a)
  ComponentIDof(c), lengthof(c), valueof(c)
```

(c) Описание всех полей, использующих FULLDATA-TLV

```
Путь к экземпляру структуры U ...
  FULLDATA-TLV
    valueof(a)
    FULLDATA-TLV
      valueof(b)
    valueof(c)
```

Примечание. Поле переменного размера требует добавления FULLDATA-TLV во внешний FULLDATA-TLV, как для компоненты b выше.

Пример 4

Структура, содержащая массив структур другого типа.

```
struct V {
uint32 x
uint32 y
struct U z[]
}
```

(a) Кодирование использует SPARSEDATA-TLV с двумя экземплярами z[], также описываемыми SPARSEDATA-TLV в предположении, что кодируются только элементы массива z[] с индексами 10 и 15.

```
Путь к экземпляру структуры V ...
SPARSEDATA-TLV
 ComponentIDof(x), lengthof(x), valueof(x)
 ComponentIDof(y), lengthof(y), valueof(y)
 ComponentIDof(z), lengthof(all below)
   ComponentID = 10 (De index 10 Drop z[]), lengthof(all below)
       ComponentIDof(a), lengthof(a), valueof(a)
       ComponentIDof(b), lengthof(b), valueof(b)
   ComponentID = 15 (index 15 Drop z[]), lengthof(all below)
       ComponentIDof(a), lengthof(a), valueof(a)
       ComponentIDof(c), lengthof(c), valueof(c)
```

Отметим пропуск в компонентах z (после 10 следует 15), а также разрыв в индексе 15 с присутствием только компонент a и c, но не b.

Приложение D. Варианты использования

Для приведённых ниже вариантов использования рассмотрим LFB с перечисленными ниже компонентами.

```
foo1, type u32, ID = 1
foo2, type u32, ID = 2
table1: type array, ID = 3
        компоненты:
        t1, type u32, ID = 1
        t2, type u32, ID = 2  // индекс в table2
        KEY: nhkey, ID = 1, V = t2
table2: type array, ID = 4
        компоненты:
        j1, type u32, ID = 1
        j2, type u32, ID = 2
        KEY: akey, ID = 1, V = { j1,j2 }
table3: type array, ID = 5
        компоненты:
        someid, type u32, ID = 1
        name, type string variable sized, ID = 2
table4: type array, ID = 6
        компоненты:
        j1, type u32, ID = 1
        j2, type u32, ID = 2
        j3, type u32, ID = 3
        j4, type u32, ID = 4
        KEY: mykey, ID = 1, V = { j1}
table5: type array, ID = 7
        компоненты:
        p1, type u32, ID = 1
        p2, type array, ID = 2, array components of type-X
Type-X:
        x1, ID 1, type u32
        x2, ID2 , type u32
                KEY: tkey, ID = 1, V = { x1}
```

Во всех примерах используется valueof(x) для указания значения упомянутой компоненты x. При отсутствии F_SEL** (биты 00) флаги не будут показывать какой-либо выбор.

Все примеры показывают для кодирования данных только применение FULLDATA-TLV, хотя в некоторых случаях имеет больше смысла SPARSEDATA-TLV. Акцент делается на показе схем сообщений. Примеры использования FULLDATA-TLV и SPARSEDATA-TLV приведены в Приложении C.

1. Получение foo1

```
OPER = GET-TLV
        PATH-DATA-TLV: IDCount = 1, IDs = 1
```

Результат

```
OPER = GET-RESPONSE-TLV
        PATH-DATA-TLV:
                flags=0, IDCount = 1, IDs = 1
                FULLDATA-TLV L = 4+4, V =  valueof(foo1)
```

2. Установка foo2 = 10

```
OPER = SET-TLV
        PATH-DATA-TLV:
                flags = 0,  IDCount = 1, IDs = 2
                FULLDATA-TLV: L = 4+4, V=10
```

Результат

```
OPER = SET-RESPONSE-TLV
        PATH-DATA-TLV:
                flags = 0,  IDCount = 1, IDs = 2
                RESULT-TLV
```

3. Выгрузить (dump) table2

```
    OPER = GET-TLV
        PATH-DATA-TLV:
                IDCount = 1, IDs = 4
```

Результат

```
    OPER = GET-RESPONSE-TLV
        PATH-DATA-TLV:
                flags = 0, IDCount = 1, IDs = 4
                FULLDATA-TLV: L = XXX, V= последовательность index, valueof(j1),
                     valueof(j2), представляющая всю таблицу
```

Примечание. Следует иметь возможность взять GET-RESPONSE-TLV и преобразовать его в SET-TLV. Если результат приведённого выше примера передать обратно в SET-TLV (вместо GET-RESPONSE_TLV), будет заменено все содержимое таблицы.

4. Пример множества операций для создания записей 0-5 в table2 (ошибки игнорируются)

```
OPER = SET-TLV
        PATH-DATA-TLV:
                flags = 0 , IDCount = 1, IDs = 4
                PATH-DATA-TLV
                  flags = 0, IDCount = 1, IDs = 0
                  FULLDATA-TLV valueof(j1), valueof(j2) of entry 0
                PATH-DATA-TLV
                  flags = 0, IDCount = 1, IDs = 1
```

```
                  FULLDATA-TLV valueof(j1), valueof(j2) of entry 1
                PATH-DATA-TLV
                  flags = 0, IDCount = 1, IDs = 2
                  FULLDATA-TLV valueof(j1), valueof(j2) of entry 2
                PATH-DATA-TLV
                  flags = 0, IDCount = 1, IDs = 3
                  FULLDATA-TLV valueof(j1), valueof(j2) of entry 3
                PATH-DATA-TLV
                  flags = 0, IDCount = 1, IDs = 4
                  FULLDATA-TLV valueof(j1), valueof(j2) of entry 4
                PATH-DATA-TLV
                  flags = 0, IDCount = 1, IDs = 5
                  FULLDATA-TLV valueof(j1), valueof(j2) of entry 5
```

Результат

```
OPER = SET-RESPONSE-TLV
        PATH-DATA-TLV:
                flags = 0 , IDCount = 1, IDs = 4
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 0
                    RESULT-TLV
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 1
                    RESULT-TLV
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 2
                    RESULT-TLV
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 3
                    RESULT-TLV
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 4
                    RESULT-TLV
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 5
                    RESULT-TLV
```

5. Пример блочных операций (с пропусками) - замена записей 0,2 в table2.

```
OPER = SET-TLV
        PATH-DATA-TLV:
             flags =  0 , IDCount = 1, IDs = 4
             PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 0
                FULLDATA-TLV c valueof(j1), valueof(j2) равными 0
             PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 2
                FULLDATA-TLV c valueof(j1), valueof(j2) равными 2
```

Результат

```
OPER = SET-TLV
        PATH-DATA-TLV:
             flags =  0 , IDCount = 1, IDs = 4
             PATH-DATA-TLV
                 flags = 0, IDCount = 1, IDs = 0
                 RESULT-TLV
             PATH-DATA-TLV
                 flags = 0, IDCount = 1, IDs = 2
                 RESULT-TLV
```

6. Пример получения первой строки из table2.

```
OPER = GET-TLV
        PATH-DATA-TLV:
                IDCount = 2, IDs = 4.0
```

Результат

```
OPER = GET-RESPONSE-TLV
        PATH-DATA-TLV:
                IDCount = 2, IDs = 4.0
                 FULLDATA-TLV c valueof(j1), valueof(j2)
```

7. Получение записей 0-5 из table2.

```
OPER = GET-TLV
        PATH-DATA-TLV:
                flags = 0, IDCount = 1, IDs = 4
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 0
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 1
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 2
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 3
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 4
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 5
```

Результат

```
OPER = GET-RESPONSE-TLV
        PATH-DATA-TLV:
                flags = 0, IDCount = 1, IDs = 4
                PATH-DATA-TLV
```

```
                    flags = 0, IDCount = 1, IDs = 0
                    FULLDATA-TLV c valueof(j1), valueof(j2)
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 1
                    FULLDATA-TLV c valueof(j1), valueof(j2)
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 2
                    FULLDATA-TLV c valueof(j1), valueof(j2)
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 3
                    FULLDATA-TLV c valueof(j1), valueof(j2)
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 4
                    FULLDATA-TLV c valueof(j1), valueof(j2)
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 5
                    FULLDATA-TLV c valueof(j1), valueof(j2)
```

8. Создание строки в table2, индекс 5.

```
OPER = SET-TLV
        PATH-DATA-TLV:
             flags = 0, IDCount = 2, IDs = 4.5
             FULLDATA-TLV c valueof(j1), valueof(j2)
```

Результат

```
OPER = SET-RESPONSE-TLV
        PATH-DATA-TLV:
             flags = 0, IDCount = 1, IDs = 4.5
             RESULT-TLV
```

9. Выгрузка содержимого table1.

```
OPER = GET-TLV
        PATH-DATA-TLV:
                flags = 0, IDCount = 1, IDs = 3
```

Результат

```
OPER = GET-RESPONSE-TLV
        PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 3
                FULLDATA-TLV, Length = XXXX
                        (зависит от размера table1)
                        index, valueof(t1),valueof(t2)
                        index, valueof(t1),valueof(t2)
                        ...
```

10. Использование ключей для получения строки из table4 с j1=100 (j1 определяет ключ для таблицы и его KeyID = 1).

```
OPER = GET-TLV
        PATH-DATA-TLV:
                flags = F_SELKEY  IDCount = 1, IDs = 6
                KEYINFO-TLV = KeyID=1, KEY_DATA=100
```

Результат

Если j1=100 имела индекс 10

```
OPER = GET-RESPONSE-TLV
        PATH-DATA-TLV:
                flags = 0, IDCount = 1, IDs = 6.10
                FULLDATA-TLV c valueof(j1), valueof(j2),valueof(j3),valueof(j4)
```

11. Удаление строки с KEY (j1=100, j2=200) из table2 (пара j1, j2 определяет ключ для table2).

```
OPER = DEL-TLV
        PATH-DATA-TLV:
                flags = F_SELKEY  IDCount = 1, IDs = 4
                KEYINFO-TLV:  {KeyID =1 KEY_DATA=100,200}
```

Результат

Если пара (j1=100, j2=200) была строкой 15

```
OPER = DELETE-RESPONSE-TLV
        PATH-DATA-TLV:
                flags = 0  IDCount = 2, IDs = 4.15
                RESULT-TLV
```

12. Выгрузка содержимого table3

Следует отметить, что таблица имеет столбец с именем компоненты переменного размера. Целью является демонстрация кодирования таких компонент.

```
OPER = GET-TLV
        PATH-DATA-TLV:
             flags = 0 IDCount = 1, IDs = 5
```

Результат

```
OPER = GET-RESPONSE-TLV
    PATH-DATA-TLV:
       flags = 0  IDCount = 1, IDs = 5
           FULLDATA-TLV, Length = XXXX
            index, someidv, TLV: T=FULLDATA-TLV, L = 4+strlen(namev),
                   V = valueof(v)
            index, someidv, TLV: T=FULLDATA-TLV, L = 4+strlen(namev),
                   V = valueof(v)
            index, someidv, TLV: T=FULLDATA-TLV, L = 4+strlen(namev),
                   V = valueof(v)
            index, someidv, TLV: T=FULLDATA-TLV, L = 4+strlen(namev),
                   V = valueof(v)
               ...
```

13. Множество неделимых операций

Примечание 1. Это эмулирует добавление новой записи nexthop и неделимого обновления записи L3, указывавшей на старых NH, с заменой её на новый[1]. Предполагается, что обе таблицы относятся к одному LFB.

Примечание. Обратите внимание на две операции SET для одного экземпляра LFB.

```
//Операция 1 - добавление новой записи в table2, индекс 20.
OPER = SET-TLV
        Path-TLV:
                flags = 0, IDCount = 2,  IDs = 4.20
                FULLDATA-TLV, V= valueof(j1),valueof(j2)

// Операция 2 - обновление в table1 записи, которая была указана
// t2 = 10 на запись, указанную индексом 20
OPER = SET-TLV
        PATH-DATA-TLV:
                flags = F_SELKEY, IDCount = 1, IDs = 3
                KEYINFO-TLV = KeyID=1 KEY_DATA=10
                PATH-DATA-TLV
                        flags = 0  IDCount = 1, IDs = 2
                        FULLDATA-TLV, V= 20
```

Результат

```
// Первая операция SET
OPER = SET-RESPONSE-TLV
        PATH-DATA-TLV
                flags = 0 IDCount = 3, IDs = 4.20
                RESULT-TLV code = success
                        FULLDATA-TLV, V = valueof(j1),valueof(j2)
// Вторая операция SET (предполагается обновление записи 16)
OPER = SET-RESPONSE-TLV
        PATH-DATA-TLV
                flags = 0 IDCount = 2, IDs = 3.16
                PATH-DATA-TLV
                        flags = 0  IDCount = 1, IDs = 2
                        RESULT-TLV code = success
                                FULLDATA-TLV, Length = XXXX v=20
```

14. Селективная установка

В table4 для индексов 1, 3, 5, 7 и 9 меняется j1 на 100, j2 на 200, j3 на 300. j4 сохраняется.

```
PER = SET-TLV
    PATH-DATA-TLV
        flags = 0, IDCount = 1, IDs = 6
        PATH-DATA-TLV
            flags = 0, IDCount = 1, IDs = 1
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 1
                FULLDATA-TLV, Length = XXXX, V = {100}
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 2
                FULLDATA-TLV, Length = XXXX, V = {200}
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 3
                FULLDATA-TLV, Length = XXXX, V = {300}
        PATH-DATA-TLV
            flags = 0, IDCount = 1, IDs = 3
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 1
                FULLDATA-TLV, Length = XXXX, V = {100}
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 2
                FULLDATA-TLV, Length = XXXX, V = {200}
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 3
                FULLDATA-TLV, Length = XXXX, V = {300}
        PATH-DATA-TLV
            flags = 0, IDCount = 1, IDs = 5
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 1
                FULLDATA-TLV, Length = XXXX, V = {100}
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 2
                FULLDATA-TLV, Length = XXXX, V = {200}
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 3
                FULLDATA-TLV, Length = XXXX, V = {300}
        PATH-DATA-TLV
            flags = 0, IDCount = 1, IDs = 7
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 1
                FULLDATA-TLV, Length = XXXX, V = {100}
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 2
                FULLDATA-TLV, Length = XXXX, V = {200}
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 3
                FULLDATA-TLV, Length = XXXX, V = {300}
```

[1]В оригинале допущена ошибка, см. https://www.rfc-editor.org/errata/eid2574. *Прим. перев.*

```
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 9
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 1
                    FULLDATA-TLV, Length = XXXX, V = {100}
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 2
                    FULLDATA-TLV, Length = XXXX, V = {200}
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 3
                    FULLDATA-TLV, Length = XXXX, V = {300}
```

Отклик

```
    OPER = SET-RESPONSE-TLV
        PATH-DATA-TLV
            flags = 0, IDCount = 1, IDs = 6
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 1
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 1
                    RESULT-TLV
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 2
                    RESULT-TLV
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 3
                    RESULT-TLV
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 3
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 1
                    RESULT-TLV
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 2
                    RESULT-TLV
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 3
                    RESULT-TLV
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 5
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 1
                    RESULT-TLV
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 2
                    RESULT-TLV
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 3
                    RESULT-TLV
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 7
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 1
                    RESULT-TLV
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 2
                    RESULT-TLV
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 3
                    RESULT-TLV
            PATH-DATA-TLV
                flags = 0, IDCount = 1, IDs = 9
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 1
                    RESULT-TLV
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 2
                    RESULT-TLV
                PATH-DATA-TLV
                    flags = 0, IDCount = 1, IDs = 3
                    RESULT-TLV
```

15. Примеры манипуляций с таблицей таблиц - получение x1 из table10, строка 4 внутри table5 строка 10

```
    operation = GET-TLV
            PATH-DATA-TLV
                    flags = 0  IDCount = 5, IDs=7.10.2.4.1
```

Результат

```
    operation = GET-RESPONSE-TLV
            PATH-DATA-TLV
                    flags = 0  IDCount = 5, IDs=7.10.2.4.1
                    FULLDATA-TLV: L=XXXX, V = valueof(x1)
```

16. Получение X2 из строки row 10, table10 в table5 на основе значения x1 = 10 (x1 имеет KeyID 1).

```
    operation = GET-TLV
            PATH-DATA-TLV
                    flag = F_SELKEY, IDCount=3, IDS = 7.10.2
                    KEYINFO-TLV, KeyID = 1, KEY_DATA = 10
                    PATH-DATA-TLV
```

```
                        IDCount = 1, IDS = 2 //select x2
```

Результат

Если x1=10 было в записи 11

```
operation = GET-RESPONSE-TLV
        PATH-DATA-TLV
                flag = 0, IDCount=5, IDS = 7.10.2.11
                PATH-DATA-TLV
                        flags = 0  IDCount = 1, IDS = 2
                        FULLDATA-TLV: L=XXXX, V = valueof(x2)
```

17. Дополнительный пример манипуляций с таблицей таблиц

Рассмотрим table6, определённую ниже.

```
table6: type array, ID = 8
        компоненты:
        p1, type u32, ID = 1
        p2, type array, ID = 2, array components of type type-A
type-A:
        a1, type u32, ID 1,
        a2, type array ID2 ,array components of type type-B
type-B:
        b1, type u32, ID 1
        b2, type u32, ID 2
```

В этом примере проводятся замены

table6.10.p1 на 111;
table6.10.p2.20.a1 на 222;
table6.10.p2.20.a2.30.b1 на 333

в одном сообщении и одной операции.

Это можно сделать двумя способами:

a) используя вложенность;
b) используя плоский путь к данным.

A. Метод использует вложенность в одном сообщении с одной операцией.

```
operation = SET-TLV
        PATH-DATA-TLV
                flags = 0  IDCount = 2, IDs=6.10
                PATH-DATA-TLV
                        flags = 0, IDCount = 1, IDs=1
                        FULLDATA-TLV: L=XXXX,
                                V = {111}
                PATH-DATA-TLV
                        flags = 0  IDCount = 2, IDs=2.20
                        PATH-DATA-TLV
                                flags = 0, IDCount = 1, IDs=1
                                FULLDATA-TLV: L=XXXX,
                                        V = {222}
                        PATH-DATA-TLV :
                                flags = 0, IDCount = 3, IDs=2.30.1
                                FULLDATA-TLV: L=XXXX,
                                        V = {333}
```

Результат

```
operation = SET-RESPONSE-TLV
        PATH-DATA-TLV
                flags = 0  IDCount = 2, IDs=6.10
                PATH-DATA-TLV
                        flags = 0, IDCount = 1, IDs=1
                        RESULT-TLV
                PATH-DATA-TLV
                        flags = 0  IDCount = 2, IDs=2.20
                        PATH-DATA-TLV
                                flags = 0, IDCount = 1, IDs=1
                                RESULT-TLV
                        PATH-DATA-TLV :
                                flags = 0, IDCount = 3, IDs=2.30.1
                                RESULT-TLV
```

B. Метод использует плоский путь к данным в одном сообщении с одной операцией.

```
operation = SET-TLV
        PATH-DATA-TLV :
                flags = 0, IDCount = 3, IDs=6.10.1
                FULLDATA-TLV: L=XXXX,
                        V = {111}
        PATH-DATA-TLV :
                flags = 0, IDCount = 5, IDs=6.10.1.20.1
                FULLDATA-TLV: L=XXXX,
                        V = {222}
        PATH-DATA-TLV :
                flags = 0, IDCount = 7, IDs=6.10.1.20.1.30.1
                FULLDATA-TLV: L=XXXX,
                        V = {333}
```

Результат

```
operation = SET-TLV
        PATH-DATA-TLV :
                flags = 0, IDCount = 3, IDs=6.10.1
                RESULT-TLV
        PATH-DATA-TLV :
                flags = 0, IDCount = 5, IDs=6.10.1.20.1
```

```
                RESULT-TLV
        PATH-DATA-TLV :
                flags = 0, IDCount = 7, IDs=6.10.1.20.1.30.1
                RESULT-TLV
```

18. Получение всего LFB (все его компоненты и пр.)

Например, при старте CE может потребоваться целиком FE Object LFB. Его можно получить с помощью запроса для класса 1 и экземпляра 1.

```
operation = GET-TLV
        PATH-DATA-TLV
                flags = 0  IDCount = 0
```

Результат

```
operation = GET-RESPONSE-TLV
        PATH-DATA-TLV
                flags = 0  IDCount = 0
                FULLDATA-TLV кодирование LFB объекта FE
```

Адреса авторов

Avri Doria (редактор)
Lulea University of Technology
Rainbow Way
Lulea SE-971 87
Sweden
Phone: +46 73 277 1788
EMail: avri@ltu.se

Jamal Hadi Salim (редактор)
Znyx
Ottawa, Ontario
Canada
Phone:
EMail: hadi@mojatatu.com

Robert Haas (редактор)
IBM
Saumerstrasse 4
8803 Ruschlikon
Switzerland
Phone:
EMail: rha@zurich.ibm.com

Hormuzd M Khosravi (редактор)
Intel
2111 NE 25th Avenue
Hillsboro, OR 97124
USA
Phone: +1 503 264 0334
EMail: hormuzd.m.khosravi@intel.com

Weiming Wang (редактор)
Zhejiang Gongshang University
18, Xuezheng Str., Xiasha University Town
Hangzhou 310018
P.R. China
Phone: +86-571-28877721
EMail: wmwang@zjgsu.edu.cn

Ligang Dong
Zhejiang Gongshang University
18, Xuezheng Str., Xiasha University Town
Hangzhou 310018
P.R. China
Phone: +86-571-28877751
EMail: donglg@zjgsu.edu.cn

Ram Gopal
Nokia
5, Wayside Road
Burlington, MA 310035
USA
Phone: +1-781-993-3685
EMail: ram.gopal@nsn.com

Joel Halpern
P.O. Box 6049
Leesburg, VA 20178
USA
Phone: +1-703-371-3043
EMail: jmh@joelhalpern.com

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru