EBU

OPERATING EUROVISION

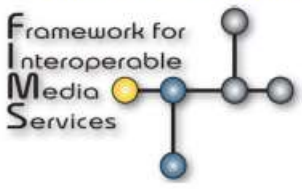
TECH 3356

SPECIFICATION OF THE FIMS MEDIA SOA FRAMEWORK

General Description

VERSION 1.1

Published Jointly With FIMS and AMWA



Geneva
February 2014

EBU

OPERATING EUROVISION AND EURORADIO

TECH 3356

SPECIFICATION OF THE FIMS MEDIA SOA FRAMEWORK

Part 1: General Description

VERSION 1.1

Внимание!
Данный перевод **НЕ** претендует на аутентичность
и может содержать отдельные неточности.
Оригинал документа на сайте https://tech.ebu.ch

СПЕЦИФИКАЦИЯ МЕДИА СТРУКТУРЫ SOA FIMS

Часть 1: Общее описание

ВЕРСИЯ 1.1

Опубликовано совместно с FIMS и AMWA

Framework for Interoperable Media Services

ADVANCED MEDIA WORKFLOW ASSOCIATION

Женева
Февраль 2014

Пояснительная записка

В настоящем документе описана общая коммерчески нейтральная структура для реализации взаимодействующих медиа служб с использованием системы на базе сервисно-ориентированной архитектуры (SOA), поддерживающей взаимодействие, взаимозаменяемость и повторное использование услуг, специфичных для медиа.

Выпуск FIMS 1.1 на момент публикации документа содержит следующее:

- Общее описание FIMS 1.1 (настоящий документ):

  FIMS Schema Spec-1.1.pdf

- Документация API FIMS

  APIDocumentation/START HERE.html

- Схема FIMS:

  В папке *WSDL-SOAP-XSD*

ПРИМЕЧАНИЯ – Просьба обратить внимание на то, что реализация и соответствие данной спецификации могут потребовать использования предметов, ограниченных патентными правами. Публикация данной спецификации не означает никакой позиции в отношении существования или действия какого-либо пункта или патентных прав в связи с вышеизложенным. AMWA, включая Совет директоров AMWA, не несет ответственности за определение патентов, для которых по спецификации AMWA может требоваться лицензия, и за проведение проверки юридической силы или области применения патентов, представленных его вниманию.

Copyright © 2014 AMWA

Copyright © 2014 EBU

Здесь и далее (примечание переводчика):

FIMS - Framework for Interoperable Media Services – Структура взаимодействия медиа сервисов

AMWA - Advanced Media Workflow Association - Ассоциация развития медийных рабочих процессов

Содержание

1. Область действия 4
2. Язык соответствий **4**
3. Справочные документы 5
3.1 FIMS 5
3.2 Общие ссылки..... 5
3.3 Ссылки на стандарты 5
3.3.1 Технология веб и интернет 5
3.3.2 Веб-услуги SOAP 5
3.3.3 Форматы даты и времени 5
3.3.4 Исследования / спецификации EBU и стандарты SMPTE 5
4. Обзор 6
5. Высокоуровневые архитектуры 6
5.1 Что значит быть «FIMS-совместимым»?..... 6
5.2 Опорная архитектура(ы)..... 7
5.3 Таксономия услуг 8
5.4 Сценарии: посредничество, динамическое распределение ресурсов .. 8
5.5 Медиа-центрические вопросы 9
5.5.1 Асинхронные операции для долговыполняемого процесса ... 9
5.5.2 Управление ресурсами 9
5.5.3 Media Bus..... 10
5.5.4 Безопасность 10
6. Управление медиа службами 10
6.1 Жизненный цикл службы 10
6.1.1 Внедрение 10
6.1.2 Вывод из эксплуатации 11
6.1.3 Замена / апгрейд 11
6.1.4 Версии интерфейса FIMS 11
6.2 Управление заданиями 12
6.2.1 Жизненный цикл задания 12
6.2.2 Команды управления 13
6.2.3 Ресурсно-ориентированная модель данных 14
7. Оповещение о медиа службах 15
7.1 Реестр услуг 15
7.1.1 Перечень зарегистрированных услуг 15
7.2 Описание услуги 16
8. Поведение медиа служб 16
8.1 Общее поведение услуг 16
8.1.1 Ресурсно-ориентированный диалог 16
8.1.2 Шаблоны реализации операций 18
8.1.3 Входящие и исходящие медиа 21
8.1.4 Обработка ошибок и исключений 21
8.1.5 Восстановление после отказа 22
8.1.6 Очередь заданий 22
8.1.7 Приоритет выполнения заданий 23
8.1.8 Ссылки на медиа 24
8.1.9 Задания 24
8.1.10 Ошибки 24
8.2 Обзор сервисного интерфейса 25
8.2.1 Временные ограничения 25
8.2.2 Профили 28
8.3 Сервисные интерфейсы с ограничениями REST 29
8.3.1 Пространства имен для определения служб REST 29
8.3.2 Таблицы определений служб REST 29
8.3.3 XML представление сообщений REST 30
8.3.4 JSON представление сообщений REST 30
Приложение 1: Будущие перспективы (информативно) 34
A1. Конвейерные медиа службы 34

Структура для взаимодействующих медиа служб FIMS Media SOA Framework 1.1

Общее описание

Комитет EBU	*Первый выпуск*	*Переработка*	*Переиздание*
TC	2014		

1. Область действия

Настоящий документ описывает общую коммерчески нейтральную структуру для реализации взаимодействующих медиа служб с использованием системы на базе сервисно-ориентированной архитектуры (SOA) для использования в вещании, производстве, постпроизводстве, медиа распространении и приложениях медиа архива. Структура поддерживает взаимодействие, взаимозаменяемость и повторное использование услуг, специфичных для медиа.

Описана высокоуровневая архитектура и структура. Эта структура включает следующие требования к системе и управлению: управление услугами, оповещение, поведение и связь, оповещение о контенте и времени.

Эта первая версия спецификации адресована трем базовым медиа службам: захват, преобразование и переадресация.

Настоящий документ следует читать вместе со следующими документами:

- Файлы со схемами FIMS, содержащими определения WSDL и XSD
- Документация API FIMS

Для управления выпусками спецификации используется стандартная система нумерации версий высшего, низшего порядка и ревизий. Выпуски 1.1.x должны содержать небольшие поправки несоответствий и дефектов в базовом выпуске 1.1. Дополнительные свойства выпуска 1.1 будут добавлены только в будущем документе 1.x.

2. Язык соответствий

Нормативный текст – это текст, который описывает элементы дизайна, обязательные или содержащие ключевые слова языка соответствий: «должен», «следует» или «может». Информативный текст – это текст, потенциально полезный для пользователя, но не обязательный, и может быть удален, изменен или дополнен, не влияя на возможность взаимодействия. Информативный текст не содержит ключевых слов соответствия.

Весь текст в этом документе по умолчанию является нормативным, кроме: введения, разделов, помеченных «информативно» или отдельных абзацев, начинающихся со слова «Примечание».

Ключевые слова «должен» и «не должен» означают требования, которые нужно строго соблюдать в целях соответствия документу и отклонения от которых не допускаются.

Ключевые слова, «следует» и «не следует» означают, что из нескольких вариантов один рекомендован как особенно подходящий, не упоминая и не исключая других; или что определенный ход действий предпочтителен, но не обязателен; или что (в отрицательной форме) определенный вариант или ход действий не рекомендуется, но и не запрещается.

Ключевые слова «можно» и «не нужно» означают ход действий, допустимый в рамках документа.

Ключевое слово «в резерве» означает условие, не определенное в данный момент, не должно использоваться и может быть определено в будущем. Ключевое слово «запрещено» означает «в резерве» и, кроме того, означает, что условие никогда не будет определено в будущем.

Совместимая реализация согласно данному документу включает все обязательные условия («должен») и все рекомендуемые условия («следует») в случае их реализации. Совместимая реализация не требует реализации условий («можно») и их реализации согласно описанию.

Если не указано иначе, порядок приоритетности типов нормативной информации в данном документе должен быть следующим: нормативная проза должна быть авторитетным определением; далее таблицы; затем формальные языки; затем рисунки; а затем любые другие формы языка.

3. Справочные документы

3.1 FIMS

1) Файлы схемы FIMS 1.1 в папке *WSDL-REST-XSD*
2) Базовая схема и документация API, специфическая для услуг (формат HTML) в папке *APIDocumentation/START HERE.html*

N.B. См. http://wiki.amwa.tv/ebu/ - последние версии файлов и документации.

3.2 Общие ссылки

3) World Wide Web Consortium (W3C), http://www.w3.org
4) Organization for the Advancement of Structured Information Standards (OASIS), http://www.oasis-open.org
5) Web Services Interoperability Organisation (WS-I), http://www.ws-i.org
6) Advanced Media Workflow Association (AMWA), http://www.amwa.tv
7) Society of Motion Picture and Television Engineers (SMPTE), http://www.smpte.org
8) European Broadcast Union, EBU Technical References, http://www.ebu.ch

3.3 Ссылки на стандарты

Реализаторы должны иметь общее представление о следующих технологиях:

3.3.1 Технология веб и интернет

9) RFC 1738: Uniform Resource Locators (URL)
10) RFC 1521: Mechanisms for Specifying and Describing the Format of Internet Message Bodies

3.3.2 Веб-услуги SOAP

11) WSDL 1.1
12) Web Services Description Language (WSDL) Version 2.0 Part 0: Primer - W3C Recommendation 26 June 2007. Available at http://www.w3.org/TR/wsdl20-primer .

3.3.3 Форматы даты и времени

13) EBU Tech 3295 P_META Metadata Library
14) ISO 8601:2004 Data elements and interchange formats -- Information interchange -- Representation of dates and times
15) IETF RFC 3339 Date and Time on the Internet: Timestamps

3.3.4 Исследования / спецификации EBU и стандарты SMPTE

16) EBU Tech 3293: EBU Core Metadata Set (EBUCore)
17) EBU Tech 3295: P_META Metadata Library
18) SMPTE ST 12M-1:2008 Television – Time and Control Code
19) SMPTE RP 224v11:2011: SMPTE Labels Register
20) SMPTE ST 258:2004 Television - Transfer of Edit Decision Lists
21) SMPTE ST 291:2010 Television - Ancillary Data Packet and Space Formatting
22) SMPTE ST 377-1:2009 Material Exchange Format (MXF) – File Format Specification
23) SMPTE ST 330:2004 Television - Unique Material Identifier (UMID)

24) SMPTE ST 434: 2006 Material Exchange Format - XML Encoding for Metadata and File Structure Information

25) SMPTE ST 436: 2006 Television - MXF Mappings for VBI Lines and Ancillary Data Packets

26) SMPTE ST 2032-1:2007 Media Dispatch Protocol (MDP) - Protocol Specification

4. Обзор

Этот раздел информативный.

Настоящая спецификация определяет общий подход к интеграции программных компонентов в современное оборудование медиа производства, которая считается главным требованием всей медиа индустрии.

Спецификация основана на общей структуре для интеграции компонентов многократного использования для производства мультимедийного контента, который поддерживает коммерческие функции профессиональной медиа индустрии. Эта структура для медиа служб включает специфические детальные определения общих интерфейсов медиа служб.

Спецификация сфокусирована на сервисно-ориентированной архитектуре (SOA) и отражает переход медиа индустрии к системам, использующим данный подход. Медиа компании переходят к более быстрому внедрению этих систем из-за необходимости увеличения подвижности на рынке, где пользовательские требования быстро меняются. Кроме подвижности, этим компаниям требуется увеличение эксплуатационной технологичности, масштабируемости и расширяемости, что обеспечивают стандартизированные сервисные интерфейсы.

Для правильной эксплуатации этой технологии должна быть внедрена общая структура, помогающая обеспечить взаимодействие интеграции, взаимозаменяемость и повторное использование услуг. Это существенно снизит стоимость интеграции, даст пользователям более свободный выбор наиболее подходящих продуктов на рынке в любое время, повышение эксплуатационной технологичности и помощь во внедрении новых технологий.

В **Разделе 5** дан обзор высокоуровневой архитектуры и общая опорная модель структуры FIMS.

В **Разделе 6** описано управление медиа службами и определен жизненный цикл услуги и задания.

В **Разделе 7** описано оповещение о медиа службах, включая реестр услуг, и описание услуг.

Поведение медиа служб описано в **Разделе 8**. Он включает поведение, общее для всех категорий услуг. Также описано поведение, специфичное для каждой категории услуг.

Коммуникация медиа служб определена в **Части 2, Описание схемы**. Объектная модель FIMS описана набором схем XML, которые дают представление объектной модели для общих объектов и расширений для различных классов услуг. **Часть 2** Раздел S0 определяет базовую схему, а сервисные интерфейсы определены в последующих разделах.

Предполагается, что в будущих версиях спецификации FIMS будет расширено число поддерживаемых интерфейсов медиа служб.

5. Высокоуровневые архитектуры

5.1 Что значит быть «FIMS-совместимым»?

Соответствие FIMS 1.0 требует выполнения следующих условий:

1) Сообщения должны быть реализованы согласно структуре описания услуг FIMS

2) Имена сообщений или определенные схемы не должны изменяться.

3) Сообщения не должны обрабатываться средствами, зависящими от расширений схемы.

4) В случае реализаций SOAP каждый сервисный интерфейс должен соответствовать FIMS WSDL. Для реализаций REST сообщения должны соответствовать преобразованиям заголовка и тела REST, определенным спецификацией для каждой службы. Полезная нагрузка сообщений должна быть согласована со схемой FIMS.

5) В случае реализаций SOAP каждая услуга должна реализовывать как минимум обязательные части определения FIMS WSDL. Аналогично, в реализациях REST должны быть реализованы как минимум обязательные части, определенные в общих схемах FIMS. и не должна их расширять, за исключением полей, предусматривающих расширения производителя.

5.2 Опорная архитектура(ы)

На Рис. 1 показана общая опорная модель структуры FIMS, включая области, не определенные в FIMS 1.1.

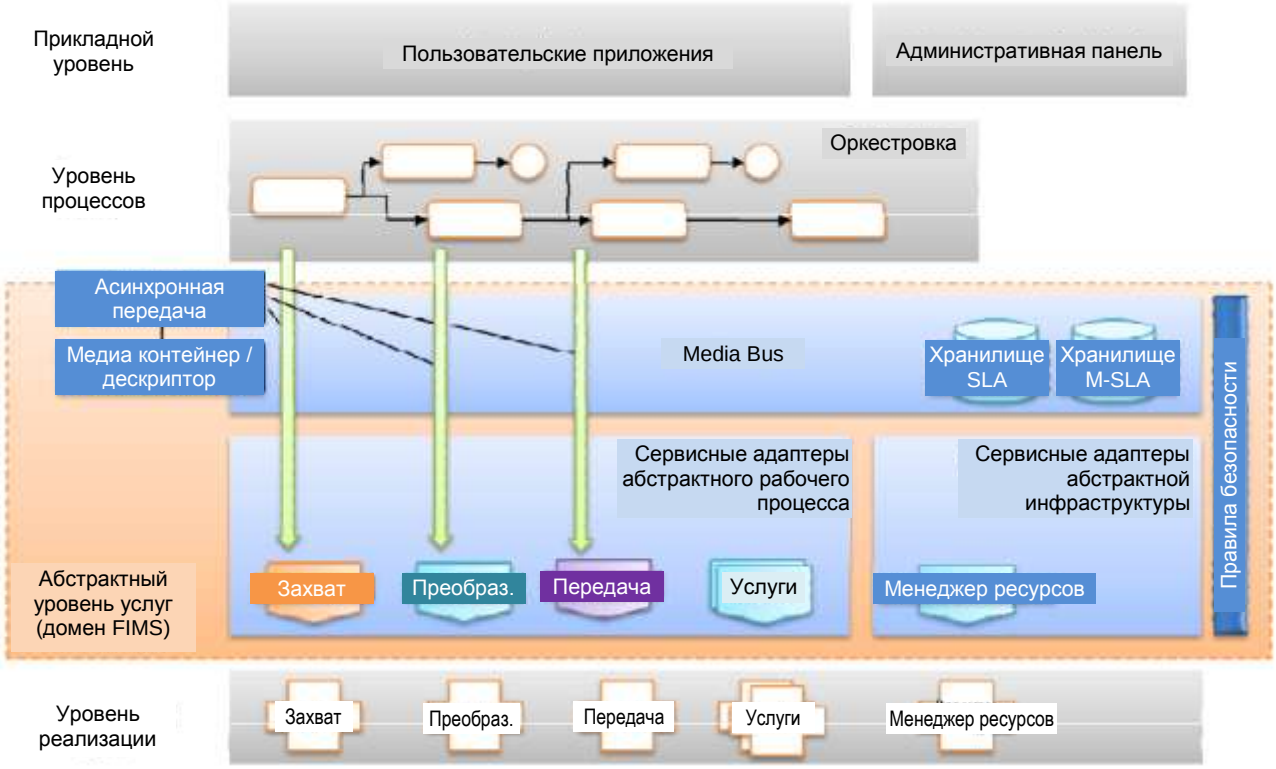


Рис. 1: Опорная модель структуры FIMS

Для учета потребностей аудиовизуальных медиа структура FIMS должна учитывать аспекты медиа служб, которые могут отличаться от традиционной SOA на базе IT, такие как:

1) Асинхронная передача для обработки очень больших объемов данных, связанных с AV медиа, в установленные сроки.
2) Медиа контейнер / медиа дескриптор для связи AV метаданных с AV сущностью.
3) Служба Media Infrastructure (менеджер ресурсов) для соответствующей обработки медиа.
4) Media Bus для обмена AV данными, в дополнение к традиционной ESB для обмена XML сообщениями.
5) Media Bus M-SLA (Media-Service Level Agreement), в дополнение к традиционной SLA в ESB.
6) Правила безопасности для безопасной обработки медиа.

Не все эти аспекты рассмотрены в данной версии спецификации. Решение всех аспектов в будущих версиях – задача FIMS Task Force.

Медиа-ориентированные вопросы описаны в п.5.5.

На Рис. 2 проиллюстрирована область действия данной версии структуры FIMS:

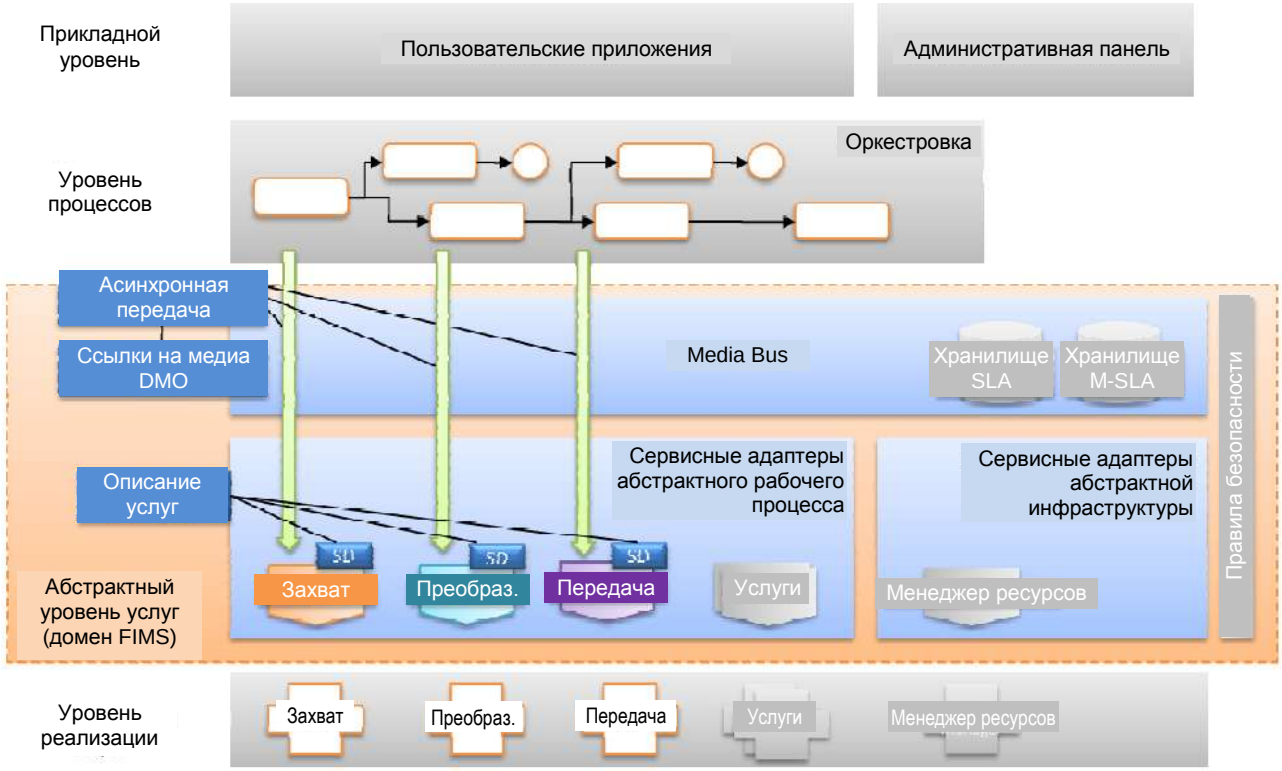


Рис. 2: Структура FIMS, рассматриваемая в спецификации 1.0

Опорная архитектура может работать с другими архитектурными подходами SOA, поддерживающими стили взаимодействия SOAP/WSDL и RESTful.

5.3 Таксономия услуг

Система с компонентами FIMS содержит две широкие категории услуг: (1) службы рабочих процессов, способные реализовать данную бизнес-цель (службы Media Workflow), (2) инфраструктурные сервисы, необходимые компоненты системы Media SOA (службы Media Infrastructure).

Инфраструктурные сервисы, используемые для создания системы Media SOA, называются службами Media Infrastructure (MI). Службы MI могут (в будущем) включать менеджер ресурсов, который производит распределение ресурсов, а также другие общие услуги типа планировщика заданий и организации очередей.

В зависимости от типа деловых рабочих процессов может меняться число и тип медиа служб.

5.4 Сценарии: Посредничество, Динамическое распределение ресурсов

Структура FIMS была задумана таким образом, чтобы позволять реализацию систем с разным уровнем сложности, в зависимости от задач, операционной среды и пользовательских требований.

На самом простом уровне услуги могут инициироваться непосредственно потребителями (клиентами пользовательских интерфейсов или оркестраторами) без промежуточного программного уровня. В этом случае оркестратор рабочих процессов берет на себя всю ответственность за любые поддерживающие операции, например, обнаружение услуг, баланс ресурсов и т.д. Это может подойти для среды малого и среднего масштаба. В более сложных инсталляциях, где несколько процессов должны идти параллельно и конкурировать за ресурсы, может быть удобно централизовать операции управления в промежуточном программном уровне. Это обычно называется сервисной шиной предприятия (Enterprise Service Bus – ESB). Функции, связанные с ESB, включают централизованное выполнение мер безопасности, динамическое распределение ресурсов, управление реестром услуг, баланс нагрузки, устойчивость к отказам и посреднические операции, такие как перевод протоколов (например, из SOAP в REST).

Определение функций, которые должен иметь такой промежуточный уровень, не входит в рамки данной версии спецификации.

5.5 Медиа-ориентированные вопросы

5.5.1 Асинхронные операции для долговыполняемых процессов

Медиа процессы на базе SOA часто долговыполняемы, иногда активны в течение часов, дней или даже недель. Это налагает определенные требования к устойчивости платформы SOA BPM. Серверы могут остановиться или перезапуститься во время работы процессов. Система должна уметь перезапускаться в той же точке рабочего процесса и управлять процессами без потери данных.

Многие медиа службы на базе SOA работают на внешнем оборудовании или являются системами на базе программного обеспечения, которые работают а слабосвязанной асинхронной среде. Если система оркестровки остановится, эти службы могут продолжать работать, а состояния заданий нужно восстанавливать после перезапуска системы.

Обновления статуса асинхронно операции можно реализовать либо путем обратного вызова, либо упорядоченного опроса, в зависимости от архитектуры платформы SOA.

Детали асинхронной связи описаны в Разделе 8.

5.5.2 Управление ресурсами

5.5.2.1 Планирование процессов и управление ресурсами

Вследствие долгого времени обработки в результате большого размера AV данных планирование процессов и управление ресурсами могут быть критичны для медиа систем на базе SOA.

Хотя концепция службы менеджера ресурсов (см. Рис. 3) включает планирование процессов и управление ресурсами, планирование процессов может инициировать только система оркестровки. Служба менеджера ресурсов может быть инициирована системой оркестровки, посредником в связующем программном обеспечении или любой службой рабочих процессов.

Интерфейс для службы планирования процессов может быть скрыт, если она инициируется только в закрытой системе оркестровке, т.к. это задача разработчика системы оркестровки. Однако, если она также используется службами рабочих процессов, интерфейс менеджера ресурсов должен быть определен в составе структуры FIMS. Это не рассматривается в данной версии спецификации.

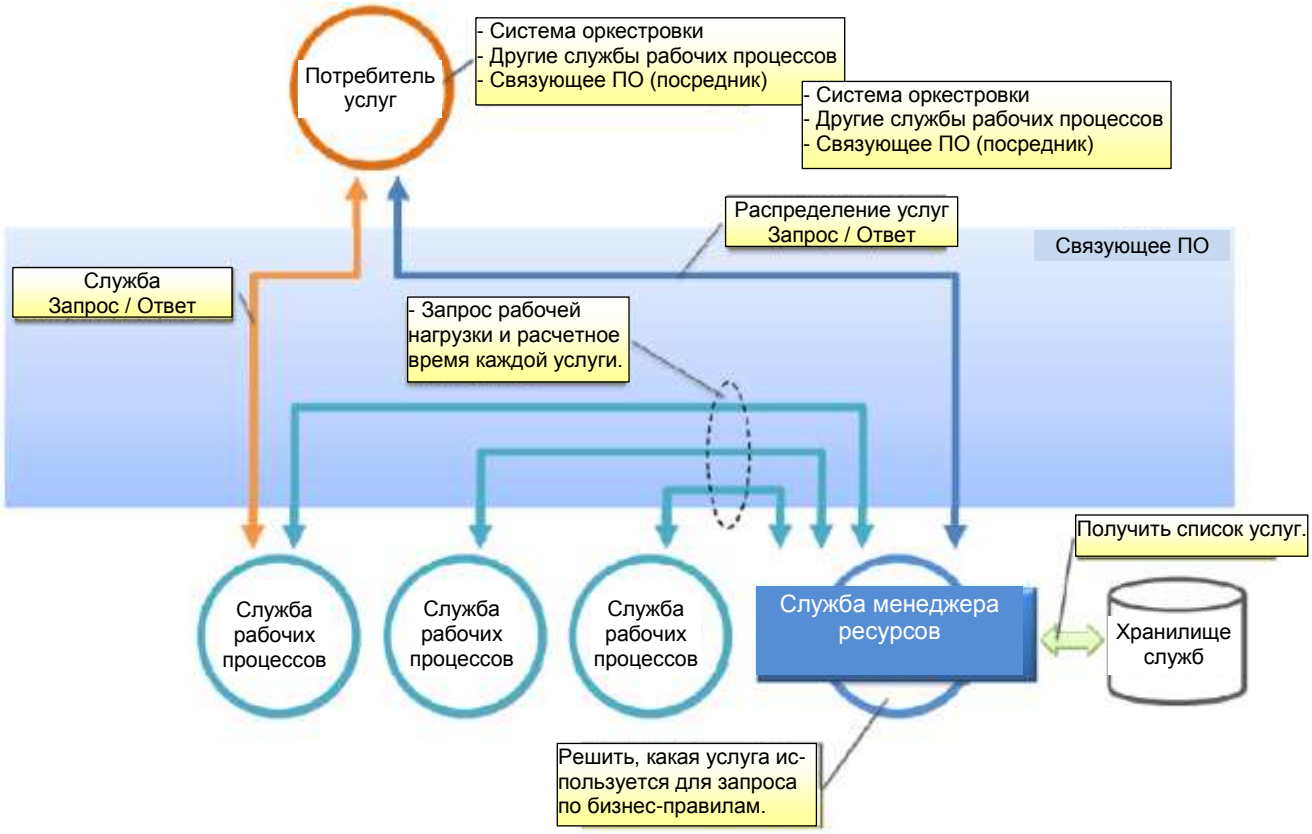


Рис. 3: Концепция менеджера ресурсов

5.5.3 Media Bus

Расширение SOA, часто называемое “Media Bus”, может облегчить операции с хранением и медиа файлами. Расширение Media Bus похоже на Enterprise Service Bus (ESB), оптимизированное для больших файлов.

Поскольку система управляет хранением и перемещением очень большого объема больших файлов (иногда миллионы файлов на проект), важно, чтобы хранилище данных было в базовых функциях системы или в виде внешней (сторонней) услуги.

Может быть много копий экземпляров файлов и много версий, находящихся в разных зонах хранения. Также может быть много файлов с видеокопиями в низком разрешении в системе, представляющей файлы «негатива с камеры» в оригинальном высоком разрешении или последовательности файлов. Необходима возможность отслеживания всех этих прокси-файлов, включая разные версии.

5.5.3.1 M-SLA

Характеристики отдельных компонентов Media Bus (главным образом накопителей), например, полоса пропускания, скорость передачи, емкость, могут требовать организации очереди с использованием общего формата сообщений.

На основании этого клиент, например, система оркестровки и/или служба медиа инфраструктуры, может выбрать накопитель, наиболее подходящий для клиента, и/или более точно оценить время выполнения.

Это соответствует SLA в традиционной системе SOA на базе IT и называется M-SLA. Поскольку детальные спецификации M-SLA требуют гармонизации с SLA, это будет обсуждаться FIMS Task Force позже в соответствии с развитием SLA.

5.5.4 Безопасность

Вследствие высокой ценности интеллектуальной собственности, передаваемой через медиа системы, критично поддерживать безопасность и давать доступ только тем, у кого есть надлежащая авторизация. Есть несколько типов безопасности, которые могут быть реализованы с SOA: безопасность на базе агентов, безопасность на базе сообщений, водяные знаки и управление цифровыми правами. В типичных медиа предприятиях требуется большинство этих мер безопасности, если не все. Безопасность на базе агентов означает отслеживание различных участников медиа системы на базе SOA-, и правильное выполнение этого, несомненно, потребует некой инфраструктуры управления идентичностью.

Технология управления идентичностью хорошо развита в IT и может успешно применяться в связующем программном обеспечении SOA. Вместо использования разрозненных хранилищ и специализированных методов аутентификации пользователей и систем безопасности инструменты управления идентичностью позволяют интегратору объединить все предприятие в одном хранилище и системе управления пользовательскими данными. Это позволяет легко производить изменения в информации о пользователях и быстро добавлять новых. В интегрированной SOA решения по управлению идентичностью также позволяют обзор данных на основе ролей.

В будущей версии спецификации FIMS будут даны правила выбора соответствующих технологий из существующих стандартов безопасности.

6. Управление медиа службами

6.1 Жизненный цикл службы

Основные состояния жизненного цикла следующие:

- Внедрение услуги
- Обновление услуги
- Замена услуги
- Вывод услуги из эксплуатации

6.1.1 Внедрение

Служба разворачивается, когда новая служба или экземпляр службы зарегистрирован в реестре услуг. Информация в реестре включает:

- Сервисные интерфейсы.
- Информация об оконечной точке услуги.
- Метаданные описания услуги.
- Политика обслуживания.

6.1.2 Вывод из эксплуатации

Вывод услуг из среды поддерживается реестром услуг, который позволяет системным администраторам сначала исключить существующие реализации, чтобы потенциальные новые потребители не видели данной реализации. Затем администраторы могут использовать функции отчета и анализа последствий в реестре услуг, позволяя операционной группе идентифицировать оставшихся пользователей версии услуги и убедиться, что они перешли в корректную альтернативную версию. После перехода всех пользователей старую версию услуги можно вывести из эксплуатации и удалить из среды.

6.1.3 Замена / Апгрейд

Услуга может обновляться или заменяться путем обновления информации в реестре услуг.

Сервисные интерфейсы и схемы согласно данной спецификации FIMS не должны требовать обновления, если услуга заменяется услугой с такими же бизнес-функциями с той же версией интерфейса. Однако по мере развития спецификаций FIMS могут публиковаться новые версии интерфейса и требовать обновления в реестре. Поскольку версии службы заменяются новыми реализациями, которые обеспечивают те же бизнес-функции, жизненный цикл управления услугами должен позволять исключение и в конечном итоге списание старых версий.

6.1.4 Версии интерфейсов FIMS

Проблемы обратной совместимости могут возникать по разным причинам. Данная версия спецификации сфокусирована на двух сценариях:

- Есть новая версия спецификаций FIMS для сервисного интерфейса.
- Реализация услуги изменена для добавления или удаления свойств (например, операций).

6.1.4.1 Новая версия интерфейсов

Время от времени технический совет FIMS будет выпускать новые версии нормативных схем FIMS и связанных с ними спецификаций с общим возрастающим номером версии. Хотя были приложены все усилия для того, чтобы новые дополнительные версии были обратно совместимы с предыдущими, характер FIMS, как структуры двусторонней связи, ведет к неизбежным проблемам у реализаторов, в том числе:

- Несовпадающие версии данных между тем, что передано в службу, и тем, что получено в синхронном ответе;
- Аналогично – для уведомлений от служб;
- Использование подтверждения времени выполнения и/или автоматически сгенерированного кода, специфичного для версии схемы.

Службы и системы оркестровки не должны обеспечивать одновременную поддержку множества версий FIMS. Конечная точка одной службы должна обеспечивать поддержку лишь одной версии схем и спецификаций FIMS. Служба может поддерживать более одной версии спецификации FIMS через дополнительные конечные точки, по одной на версию.

Схема базовых служб FIMS содержит простой тип "CurrentVersion", который должен быть строковым значением, ограниченным включением одного значения, равного номеру версии схемы. Например, постоянное значение для FIMS версии 1.1 - "v1_1_0".

Кроме сообщений FIMS v1.0.7, все сообщения FIMS SOAP с запросом, с ответом, с уведомле6ниями и событиями должны указывать номер версии с помощью обязательного атрибута "version" типа "CurrentVersion". Сообщение с ответом или уведомлением, сгенерированное конечной точкой службы, должно иметь тот же номер версии, что и в конечной точке, гарантируя, что серверы и клиенты поддерживают одинаковую версию.

Поскольку атрибут версии не присутствовал в FIMS версии 1.0.7, любое сообщение SOAP с запросом, ответом или уведомлением, которое не содержит атрибут версии, должно обрабатываться как сообщение FIMS 1.0.7.

Все сообщения FIMS REST, кроме сообщений об отказе, должны содержать обязательный заголовок "X-FIMS-Version" с тем же значением, что и номер версии спецификации, как постоянное значение простого типа "CurrentVersion".

Примечание: Для FIMS 1.0.7 не было определено преобразования FIMS REST, поэтому проблемы отсутствия атрибута версии для услуг REST больше не существует.

Для того, чтобы службы REST и SOAP могли получать сообщения об отказах, связанные с номером версии, эти сообщения не должны иметь номер версии.

Когда номер версии сообщения совпадает с поддерживаемым принимающей конечной точкой, сообщение должно быть принято для обработки. Иначе оно будет отклонено с сообщением об отказе с кодом "SVC_S00_0019", означающим несовпадение версий. Для того, чтобы пользователь конечной точки мог изменить версию своего запроса или направить сообщение в соответствующую конечную точку, сообщение об отказе должно содержать детали текущей версии, поддерживаемой конечной точкой.

Примечание: Ожидается реализация услуг перевода в конечных точках, чтобы помочь пользователям с процессом перевода из одной версии в следующую.

6.1.4.2 Реализация новой услуги

В этом сценарии нет изменения версии интерфейса, но услуга заменяется другой реализацией, в которой реализованы лишь некоторые свойства (например, операции) или добавлены новые. Во втором случае услуга обратно совместима. В первом случае есть проблема оповещения структуры о том, что в услуге не реализованы некоторые свойства4.

Необходимо следовать следующей практике:

- Описание услуг для каждой реализации услуги содержит список имеющихся операций и свойств (например, операция преобразования дает список поддерживаемых форматов).
- Новая услуга может быть инициирована через посредничество в ESB или через систему оркестровки, в случае чего можно использовать поиск в реестре услуг для нахождения экземпляра(ов) услуги, где реализована нужная операция.
- Если ни одного экземпляра с нужной операцией не найдено, должна быть показана ошибка (например, услуга с запрашиваемым свойством не найдена).

6.2 Управление заданиями

6.2.1 Жизненный цикл задания

На Рис. 4 показаны состояния, связанные с долговыполняемым заданием от его запроса до выполнения, отмены или отказа. Также показаны команды или действия, инициирующие переход в новое состояние.

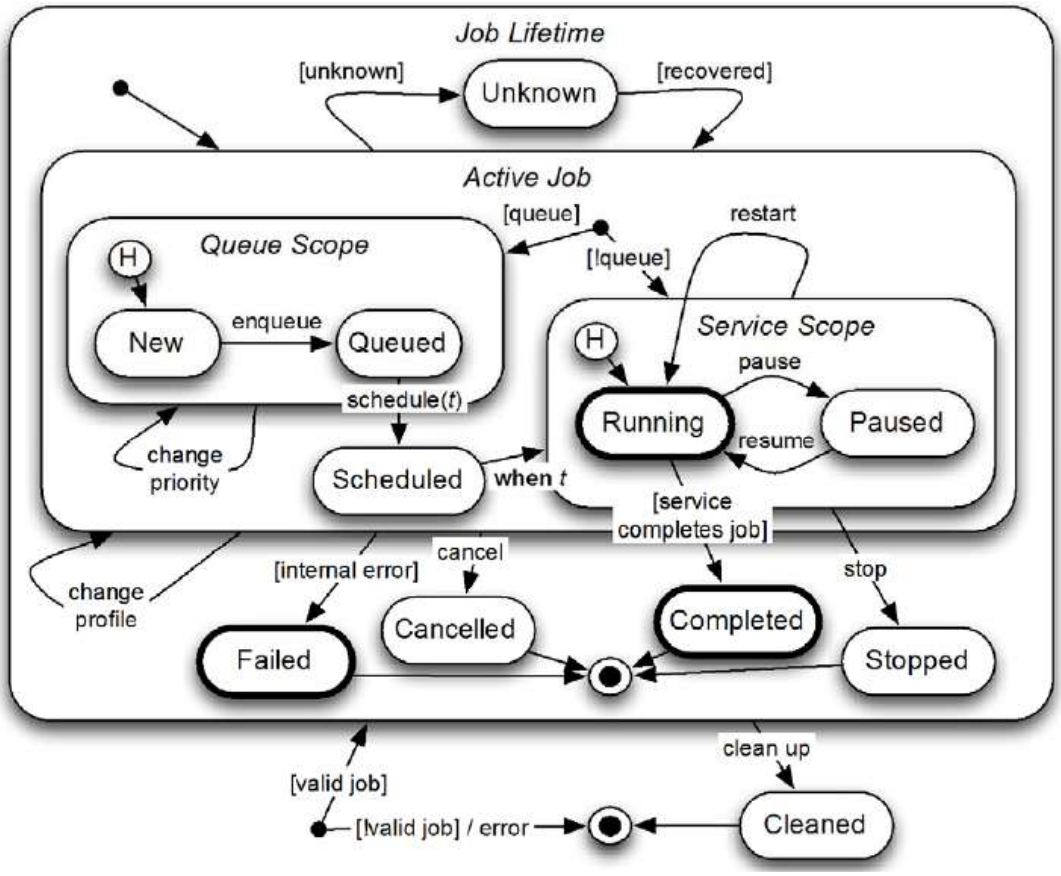


Рис. 4: Диаграмма состояний для долговыполняемого задания

Толстые круги на Рис. 4 означают обязательные состояния, требующие реализации.

Когда служба получает запрос на задание, его первичное состояние либо *New,* либо *Running*. В состоянии *New* или *Queued* задание может изменить приоритет до начала выполнения через внешние команды управления заданиями (если услуга поддерживает эти функции). Задание также может быть отменено внешней командой. После удаления задания из очереди на выполнение оно переходит в состояние *Scheduled,* и когда наступает запланированное время, переходит в состояние *Running.* Если в услуге не реализована очередь заданий, задание переходит прямо в состояние *Running* по прибытии запроса.

Состояние *Running* означает, что задание выполняется. Внешняя команда может поставить задание на паузу (и возобновить его позже). Запрос также может перезапускаться или отменяться внешними командами. Перезапуск означает начало выполнения заново. Если задание выполнено до завершения, оно переходит в состояние *Completed*. В случае ошибки во время выполнения оно переходит в состояние *Failed*. Внешняя команда может вызвать преждевременное окончание выполнения задания. Остановка задания означает принудительное завершение. Это не является ошибочной ситуацией, и результат выполнения задания до этого момента считается результатом выполнения задания.

Медиа службы часто создают большие медиа файлы, которые должны быть доступны до запроса клиентом или другими службами. После завершения (или остановки) задания результирующий продукт операции должен сохраняться этой службой. Внешняя (или внутренняя) команда (cleanup) используется для индикации службе, что результат больше не нужен, и задание переходит в состояние *Cleaned*. Службы могут независимо перемещать задания в состояние *Cleaned* по истечении времени, продолжительность которого определяется службой. Службы должны нести ответственность за очистку заданий в случае остановки или сбоя услуги.

Переход в состояния *Cancelled*, *Completed*, *Stopped* и *Failed* должен создавать уведомление в конечные точки, указанные в параметре *notifyAt* сообщения запроса, если уведомление ожидается в конце выполнения задания или при его отмене. *Completed*, *Stopped* и *Cancelled* должны создавать ответное сообщение в конечную точку *replyTo*, а *Failed* – сообщение об ошибке в конечную точку *faultTo*.

Состояние *Unknown* означает, что задание в данный момент выполняется, но его состояние нельзя определить.

6.2.2 Команды управления

Интерфейсы услуг FIMS обеспечивают три операции управления заданиями. Это manageJob, manageQueue и queryJob, как показано на Рис. 5.

MediaServiceStatus		
manageJob		
input	in	manageJobRequest
output	out	manageJobResponse
fault	fault	fault
manageQueue		
input	in	manageQueueRequest
output	out	manageQueueResponse
fault	fault	fault
queryJob		
input	in	queryJobRequest
output	out	queryJobResponse
fault	fault	fault

Рис. 5: Операции управления заданиями

Операция *manageJob* позволяет инициатору запроса передать рабочие команды для изменения состояния задания согласно предыдущей главе.

Если в службе реализована очередь заданий, ей можно управлять через операцию *manageQueue*. Команды очереди и соответствующие переходы состояний описаны в п.6.2.1.

Операция *queryJob* возвращает информацию о заданиях, переданных в службу. Инициатор запроса может дать список с ID заданий, по которым ему нужна информация.

Альтернативно он может запросить информацию о заданиях, соответствующих требованиям, указанным объектом фильтра.

Список ответов содержит спецификацию заданий и детальную информацию о каждом задании, например, *estimatedCompletionDuration* и *status*.

6.2.3 Ресурсно-ориентированная модель данных

Сообщения о заданиях, выполняемых услугах, их профилях, очередях и объектах, объектах, с которыми они работают, соответствуют модели данных, ориентированной на ресурсы. Классы этой модели данных и их отношения показаны в схеме классов UML на Рис. 6. Нереферентные свойства каждого класса не показаны.

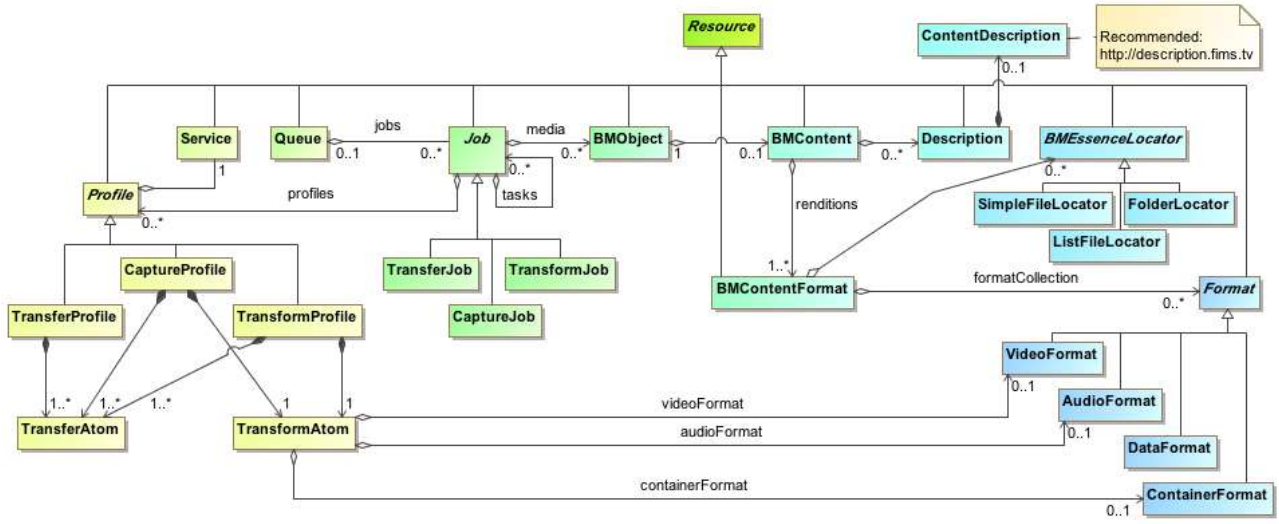


Рис. 6: Модель данных ресурсов FIMS

Будущее расширение структуры FIMS ожидается через расширение этой модели данных. Например, BMObject может быть расширено для разного вида коллекций BMObjects, которые используются в качестве входов или выходов услуг.

Каждый ресурс имеет свойство уникального идентификатора “resourceID”, которое может использоваться для ссылки, типа гиперссылки. В сочетании со средствами для организации ссылок ресурсно-ориентированный подход позволяет следующее:

- ссылка может быть дана на совместно используемые и повторяемые ресурсы для обмена сообщениями меньшего размера. Например, можно определить общий профиль для регулярно повторяемой операции преобразования и хранить его в центральном месте. Его не нужно повторять в каждом запросе на задании, ответе или запросе статуса.
- местонахождение ресурса можно отделить от его использования, что упрощает географическое масштабирование и повышает возможность устранения отказов за счет применения технологий с ресурсным разрешением, например, DNS.
- поддержка внедрения услуг авторизации данных, обеспечивающих единую точку авторизации классов метаданных. Например, система управления цифровыми фондами используется как авторизация данных для информации о контенте. Вместо копирования информации о контенте, которая сохраняется каждой системой, участвующей в задании, со всей нагрузкой хранения копии, можно сделать ссылку на авторитетную версию в системе управления фондами.
- большая степень слабой связанности, в т.ч. больше гибкости для внедрения слабосвязанных систем мониторинга и приборных панелей.
- привязки для ресурсов на базе REST.

Опциональное свойство “revisionID” может использоваться для отслеживания номеров ревизий ресурса, поддерживая системы с согласованным подходом.

Опциональное свойство URI “location” может использоваться для обеспечения определенного места ресурса (URL), на которое можно ссылаться без необходимости разрешения идентификаторов ресурсов.

В терминах нормативного представления модели данных как схемы XML, все свойства ресурсов помечены как опциональные с точки зрения XML (minOccurs=”0”), за исключением свойства “resourceID”. Этот механизм позволяет включать ресурс по ссылке вместо встраивания.

Для кодирования ссылки в ресурс вместо вставки уберите все свойства, кроме “resourceID”, “revisionID” (если оно используется) и “location” (если оно используется). Декодер сообщения либо разрешит ссылку на ресурс, либо сообщит об отказе.

Для кодирования ресурса путем вставки его значения включите значение минимум одного свойства, кроме “resourceID”, “revisionID” и “location”. Вообще, все известные значения свойств для ресурса кодируются для минимизации необходимости объединения версий ресурса. Декодер не будет разрешать ресурс по внешней ссылке и может обновить внутреннюю копию ресурса на основании предоставленной информации.

Использование необязательности XML (minOccurs=”0”) для свойства в схеме XML не подразумевает, что свойство опционально в реализации FIMS. Свойства, определенные как обязательные через спецификацию, либо в сообщении с запросом, либо с ответом, и минимум в одном состоянии задания, должны поддерживаться всеми реализациями FIMS.

*Примечание: Обязательные свойства в сообщениях с запросом или ответом определяются для каждого свойства в **Части 2** данной спецификации и в аннотациях схемы XML с источником "urn:x-fims:inclusionInRequest" и "urn:x-fims:inclusionInResponse".*

Информативная и детальная модель данных UML, показывающая большую часть модели данных FIMS на одной странице, находится в пакете со схемами XML.

FIMS определяет и нормативные технические метаданные (например, Format, VideoFormat, AudioFormat), и рекомендованные описательные метаданные (ContentDescription). Призываем пользователей использовать базовые описательные метаданные, содержащиеся в схеме FIMS, для улучшения взаимодействия. И технические, т описательные метаданные основаны на EBUCore, расширении Dublin Core от EBU для медиа.

7. Оповещение о медиа службах

7.1 Реестр услуг

Реестр полезен для отслеживания услуг во всей инфраструктуре. Когда этот реестр машиночитаемый и четко определенный, общая система может принимать умные решения. Механизм регистрации не входит в рамки 1-й версии спецификации FIMS, Это может быть сложный механизм обмена данными или человек, редактирующий файл.

7.1.1 Перечень зарегистрированных услуг

Система FIMS может обеспечивать реестр, и в таком случае она должна содержать список URL без отграничения числа строк. Этот список должен быть доступен через HTTP GET.

Каждый URL в списке должен возвращать ответ, соответствующий документу описания услуги, определенному в п.7.2.

URL в регистрационном файле может давать подсказку в виде строки запроса ‘Service’, указывая описание услуги, которое будет обнаружено при запросе данного URL.

Например, реестр может содержать URL типа следующих, включая, например, указание, что запрос вернет детали для обнаружения услуги Capture:

http://some.dns.entry/
http://some.dns.entry/AnyURL
http://192.168.1.1:8888/SomeURL?Service=Capture
http://192.168.1.1:8888/SomeOtherURL?Service=Transform

URL обнаружения одного описания должен описывать конечную точку только одной услуги и одну функцию.

Например, следующее недействительно:

http://192.168.1.1:1234/YetAnotherURL?Service=Transform,Capture

7.2 Описание услуги

Службы должны поддерживать механизм для публикации своих функций и конфигурации для систем (и людей). Без понимания того, что служба может принять или обработать, , варианты оркестровки ограничиваются и требуются внеполосные решения. При выведении информации из полосы ухудшается способность системы автоматически реагировать. Либо система должна ждать человеческого вмешательства, либо эта информация должна быть возвращена в область системы через расширение.

По этой причине спецификация FIMS определяет машиночитаемую модель для обеспечения информации об операциях, которые будет принимать услуга, вместе с текущей конфигурацией службы. Эта модель ссылается на EBU Core Metadata.

Реестр услуг (п.7.1.1) определяет URL, которые должны использоваться для возврата XML документа с описанием услуг. При запросе служба должна вернуть документ, описывающий возможности и конфигурацию данной услуги. Такие атрибуты услуги могут меняться со временем, и документ может меняться в течение жизненного цикла услуги, например, вследствие отказов системы, лицензионных ограничений, апгрейда программного обеспечения или ограничения операций со стороны системной администрации.

В качестве альтернативы можно использовать QueryCapabilityRequest для получения местонахождения описания услуг.

Один экземпляр услуги может иметь множество классов операции (например, услуга, которая может выполнять преобразование, а также открытые службы переадресации), услуга не должна регистрировать их в одной конечной точке. Такая услуга должна регистрировать множество конечных точек.

8. Поведение медиа служб

Этот раздел адресован поведению, общему для всех категорий услуг. Также описано поведение, специфичное для каждой категории услуг.

8.1 Общее поведение услуг

8.1.1 Ресурсно-ориентированный диалог

Связь в FIMS состоит из диалога сообщений о FIMS-определенных ресурсах (задание, очередь, услуга, профиль, BMObject и т.д.) между службой и ее клиентом. Операция медиа службы выполняется через диалог о ресурсе задания между провайдером услуг и инициатором запроса операции.

FIMS определяет набор известных шаблонов выполнения операций, которые поддерживаются через определения SOAP/WSDL, согласно п.8.1.2. В соответствующих сообщениях встроены представления ресурсов и/или ссылки на ресурсы посредством идентификаторов как параметров для операций. Определения услуг WSDL предусматривают синхронные и асинхронные запросы, ответы, отказы и уведомления.

Подход к FIMS на базе REST может использовать стандартные команды HTTP с URI маршрутами для получения того же диалога, с описанием ресурсов, вставленным прямо в тело сообщения. Спецификация привязок между командами HTTP и FIMS-определенными шаблонами операций и механизмом событий для уведомлений на базе REST не входит в рамки данной версии спецификации FIMS.

Обратите внимание, что один провайдер услуг может одновременно поддерживать стили взаимодействия SOAP/WSDL и RESTful для одних и тех же ресурсов.

На Рис. 7 показан пример диаграммы последовательностей Capture Request/Ack с подходом SOAP.

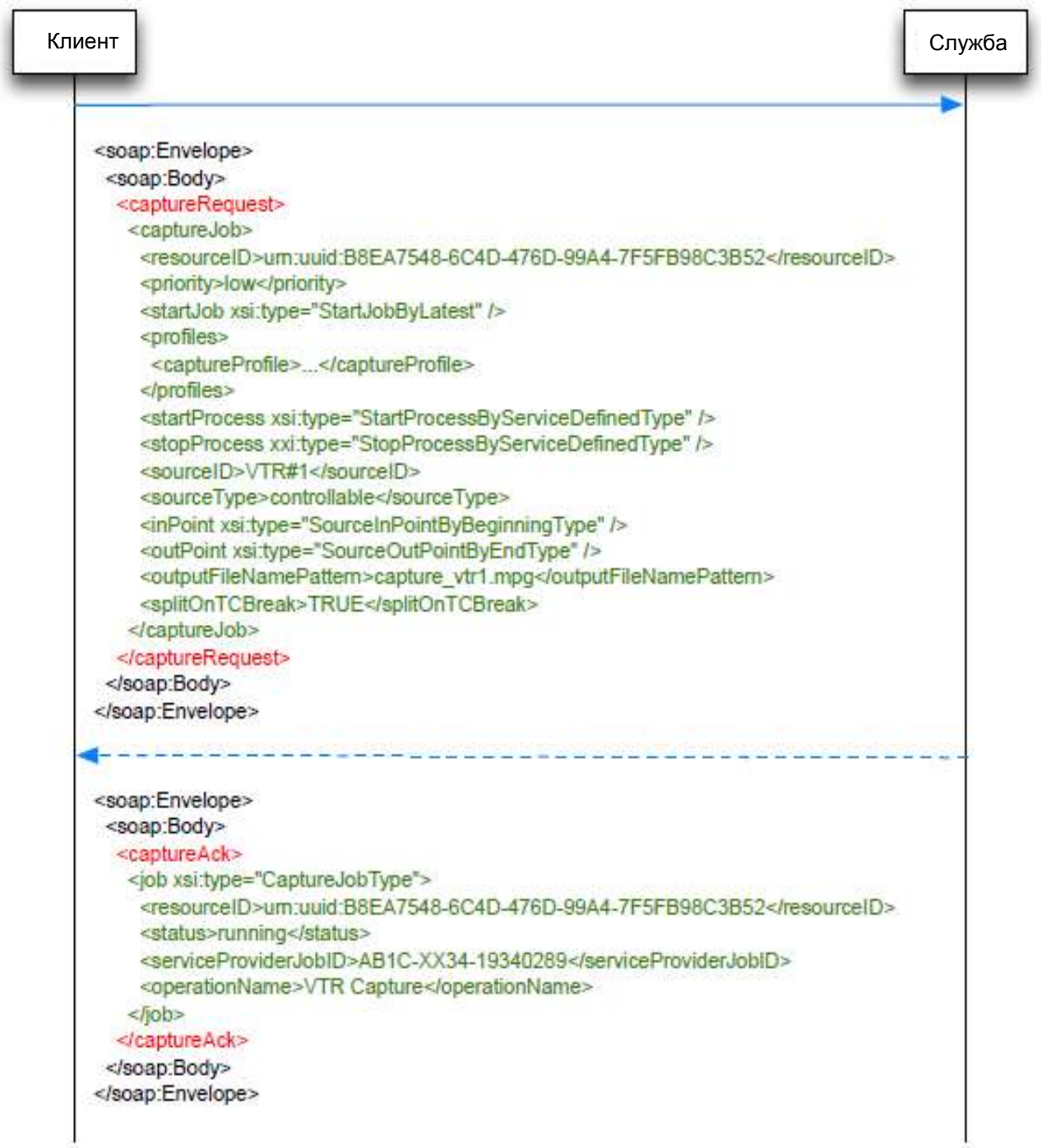


Рис. 7: Service Request/Ack (Capture) - SOAP-RPC

Эквивалент RESTful передает часть сообщения Capture Request, выделенную зеленым цветом как тело сообщения, переданного в службу. В случае успеха синхронный ответ HTTP имеет код состояния 201 Created с телом, аналогичным зеленой части сообщения Capture Ack.

Тип содержания тела сообщения RESTful FIMS должен быть представлен в формате XML или JSON. Определение преобразования определений XML схемы FIMS в сообщения в формате JSON дано в п.8.3.

Пример тела сообщения JSON Capture Request показан на Рис. 8.

Рис. 8: Service Request (Capture) – REST JSON

8.1.2 Шаблоны реализации операций

Сервисные операции, определенные интерфейсами FIMS, обеспечивают различные типы взаимодействия между провайдером и инициатором запроса услуг. Каждый определяет, как результат операции делается доступным инициатору запроса. Определение интерфейса вместе с входными параметрами операции определяет, как услуга должна возвращать ответ об операции.

8.1.2.1 Синхронный запрос / ответ

В этом режиме взаимодействия клиент услуги (например, бизнес-процесс) инициирует услугу к выполнению операции передачи входных параметров (*par1, … parN*) и получает ответ в том же сеансе связи, что и запрос. Операции, реализующие этот режим взаимодействия, не должны быть долговыполняемыми процессами во избежание блокировки клиента услуги на длительный период времени и простоев, которые могут случиться в сеансе связи. См. Рис. 9.

Рис. 9: Синхронный запрос / ответ

Примеры операций, использующих данный тип взаимодействия, это операции управления заданиями и очередями.

8.1.2.2 Асинхронный запрос / ответ с уведомлением

Этот шаблон взаимодействия следует использовать для долговыполняемых процессов. Запрос и ответ, связанные с операцией, передаются в двух разных сеансах связи. Сеанс запроса включает инициирование клиентом операции передачи входных параметров (*par1, … parN, jobGUID, notifyAt)* и подтверждение службой получение запроса. Служба должна вернуть подтверждение, когда она готова ответить на любые дальнейшие действия для этого запроса от системы оркестровки. См. Рис. 10.

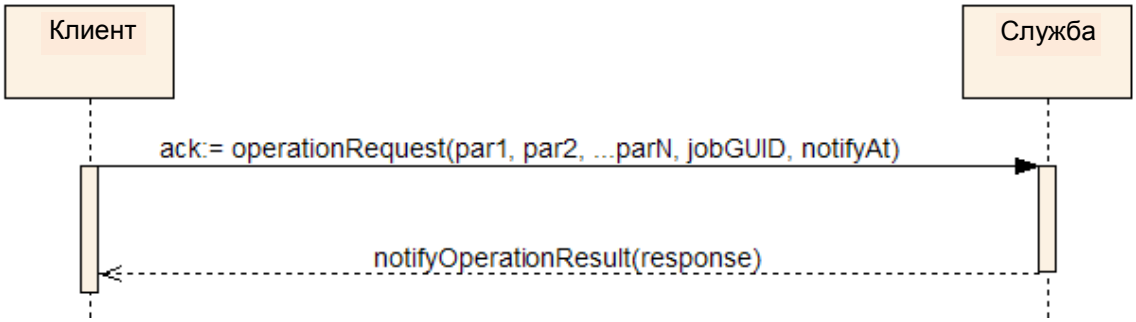


Рис. 10: Асинхронный запрос / ответ с уведомлением

Параметр *jobGUID* уникально идентифицирует запрос на задание с точки зрения бизнес-процесса (клиента услуги). Параметр *notifyAt* указывает провайдеру услуг, куда послать ответное сообщение, когда завершится операция. Он также определяет, куда отправлять сообщение об ошибке в случае отказа службы во время выполнения. Параметр *notifyAt* должен обеспечиваться для того, чтобы служба работала в этом режиме (см. определение AsyncEndpointType).

Отдельный сеанс связи используется для передачи ответного сообщения по адресу, указанному в элементе *replyTo* параметра *notifyAt*. В случае ошибки в процессе операции уведомление об ошибке должно быть передано в конечную точку, определенную элементом *faultTo* параметра *notifyAt*.

Пример операции с данным режимом взаимодействия – операция преобразования службы Transform Media. См. Рис. 11.

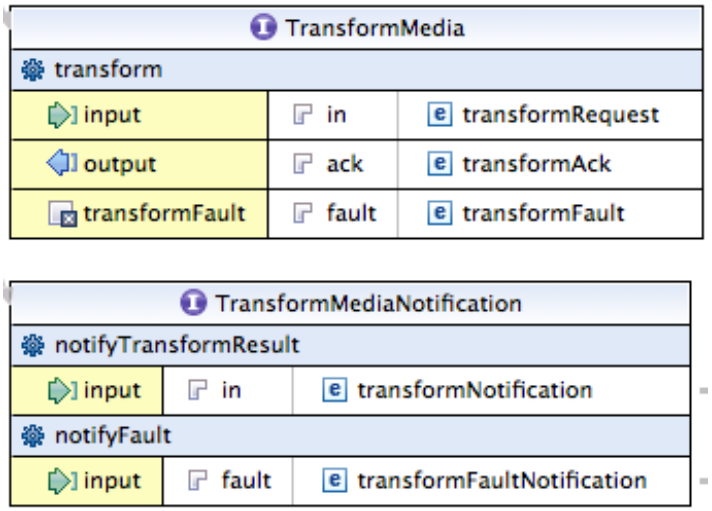


Рис. 11: Пример асинхронного запроса / ответа с уведомлением: служба преобразования

Запросы WSDL/SOAP должны иметь уведомления WSDL/SOAP. Запросы RESTful должны иметь уведомления RESTful.

Для асинхронного запроса RESTful уведомления должны посылаться в конечные точки *replyTo* и *faultTo,* содержащиеся в таблице уведомлений RESTful, предоставляемой с каждой службой.

Тело сообщения RESTful с уведомлением должно иметь специфичный для услуги тип уведомления (например, CaptureNotificationType) для ответа и специфичный для услуги тип отказа (например, CaptureFaultType) для отказа, согласно таблице уведомлений RESTful.

Таблица уведомлений RESTful должна иметь следующие столбцы:

- **Description** – Информативное описание причины уведомления.
- **HTTP method** – Метод HTTP, который должен использоваться для передачи уведомления (обычно POST).
- **Body** – Тип тела уведомления, который должен определяться по ссылке на специфичную для услуги схему XML или базовую схему XML.
- **Generation events** – События, вызывающие передачу уведомления, например, создание или сбой задания.

8.1.2.3 Асинхронный запрос / ответ с опросом – WSDL/SOAP

Это еще один шаблон взаимодействия для долговыполняемых процессов, где предпочтительный шаблон взаимодействия невозможен (например, файрвол, который не дает службе ответить клиенту). Это похоже на режим взаимодействия *Асинхронный запрос / ответ с уведомлением,* за исключением того, что служба не посылает уведомления когда завершит операцию. Параметр *notifyAt* не должен обеспечиваться для того, чтобы служба работала в этом режиме. См. Рис. 12.

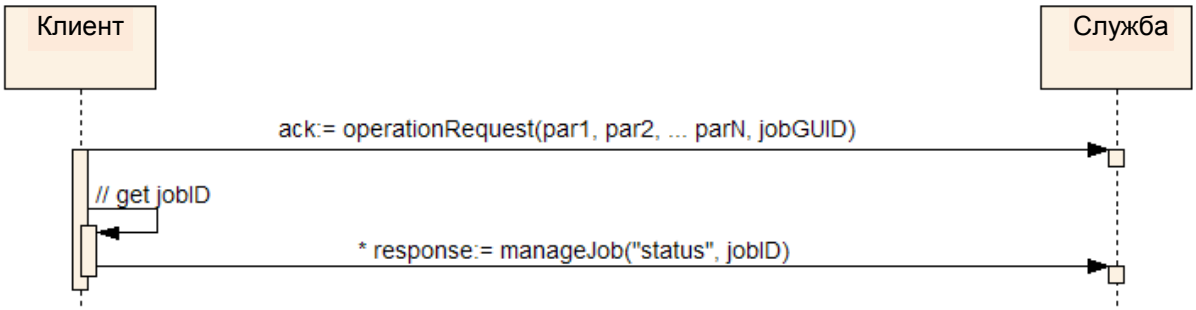


Рис. 11: Асинхронный запрос / ответ с опросом

Когда клиент передает запрос, он получает подтверждение, содержащее ID задания, идентифицированный службой.

В этом режиме взаимодействия клиент несет ответственность за опрос службы, с помощью операции *queryJob*, для получения результата операции по ее завершении. Это делается с помощью параметра *resourceID* задания (в сообщении с подтверждением). После завершения выполнения задания ответное сообщение на запрос о статусе содержит результат операции.

Результат выполнения задания содержится в элементе *jobs*. Этот элемент должен присутствовать при получении информации о задании, когда служба завершит операцию. В этом поле должна присутствовать та же информация, которая будет частью уведомления в шаблоне, указанном в п.8.1.2.2.

Операция преобразования службы Transform Media может быть реализована с помощью этого типа взаимодействия.

Реализация WSDL/SOAP служб FIMS должна поддерживать либо уведомление, либо опрос.

8.1.2.4 Асинхронный запрос / ответ с опросом - RESTful

Каждый ресурс, созданный службой, имеет уникальный идентификатор, через который е нему можно обращаться с URI. Этот URI будет синхронно возвращаться в заголовок *Location* сообщения с ответом после успешно созданного ресурса. В любое время от создания ресурса до удаления, когда есть адресуемая конечная точка, статус ресурса должен быть доступен через запрос HTTP GET в URI для данного ресурса.

Количество деталей, возвращенных запросом GET для ресурса, должно конфигурироваться через параметр запроса *detail* в составе URI. Этот параметр должен поддерживать как минимум следующие три уровня:

- *link* – Только поля *ResourceIDType*, содержащие необходимую информацию для нахождения и идентификации идентификатора ревизии для ресурса.
- *summary* – Важные поля самого ресурса и поля любых типов, которыми расширяется ресурс, согласно реализации услуги. Все коллекции элементов должны быть свернуты , чтобы элементы в коллекции отображались только как ссылки.
- *full* – Полные детали ресурса, включая все поля с данными, независимо от важности, и с полным расширением всех коллекций, чтобы каждый элемент также был представлен со всеми деталями.

Например,, вот URI для запроса GET для получения деталей задания на захват, с возвратом только самых основных деталей:

```
http://capture.device7.acme.com/fims/job/AF7E4D2F-E981-4F66-B157-2026821E3102?detail=link
```

Если задание известно службе, ответ *200 OK* генерируется следующим образом:

```
<cms:captureJob ...>
  <bms:resourceID>urn:uuid:AF7E4D2F-E981-4F66-B157-2026821E3102</bms:resourceID>
  <bms:revisionID>2</bms:revisionID>
</cms:captureJob>
```

Если задание неизвестно службе, генерируется ответ *404 Not Found*, содержащий сообщение об отказе:

```
<cms:captureFault ...>
  <bms:code>DAT_S00_0003</bms:code>
  <bms:description>Invalid jobID - the supplied jobID does not exists.</bms:description>
  <bms:detail>The given job identifier 'AF7E4D2F-E981-4F66-B157-2026821E3102' is not
    known to this device.</bms:detail>
</cms:captureFault>
```

Все задания, известные в данный момент службе, могут быть перечислены с помощью ресурса задания с пропуском идентификатора задания. Параметры запроса skip и limit

Служба должна поддерживать одно или оба преставлений XML и/или JSON. Если поддерживаются оба формата, клиент может выбрать формат с помощью заголовка *Accept* с типом MIME предпочтительного формата ответа. Служба, которая реализует запрашиваемое представление, должна ответить сообщением этого типа, или же ответить отказом.

8.1.3 Входящие и исходящие медиа

Медиа службы часто имеют дело с медиа файлами. Эти службы могут потреблять и/или производить файлы, представляющие медиа сущность. Ссылки на эти медиа файлы передаются во входящих и исходящих сообщениях для этих услуг. Медиа представлены как Business Media Object, согласно Части 2.

8.1.3.1 Обработка входящих медиа

Есть операции, требующие только медиа сущность (или список сущности), например, преобразование медиа контента. В этом случае служба принимает объект контейнера (типа BMObjectType) со списком объектов контента, который расширяет абстрактный тип BMContentFormatType, представляющий медиа сущность(и), с которыми должна работать служба.

8.1.3.2 Производство исходящего контента

Такие службы как Transform Media производят новую медиа сущность (или список медиа сущности). Эти службы в результате операции возвращают один или более объектов контента (типа BMContentFormatType).

8.1.4 Обработка ошибок и исключений

Ошибки и исключения, обнаруженные службой во время выполнения заданий, должны возвращаться инициатору запроса. Для синхронных операций, описанных в п.8.1.2.1, в ответ на запрос должно вернуться сообщение об отказе. См. Рис. 13.

Рис. 13: Обработка сообщения об отказе во время выполнения задания

Сообщение об отказе может содержать детальную информацию об ошибке. См. Рис. 14.

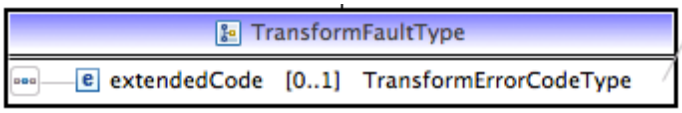


Рис. 14: Пример обработки сообщения об ошибке

Структура FIMS определяет нумерацию кодов ошибок и их описание для каждой категории услуг. Число ошибок общее для всех категорий услуг и представлено в FaultCodeType базовой схемы.

При асинхронной операции согласно п.8.1.2.2 и п.8.1.2.3 ошибки могут случаться на двух этапах. Ошибка или исключение могут быть отброшены во время запроса (например, недействительные параметры Request), и сообщение об отказе должно вернуться мгновенно в ответ на запрос. Поведение аналогично синхронному сценарию.

С другой стороны, если ошибка обнаружена во время выполнения долговыполняемого процесса (например, задание не выполнено) и в сообщении с запросом установлен параметр notifyAt, то уведомление об ошибке должно быть отправлено адресату, указанному в поле *faultTo* в *notifyAt.* См. Рис. 15.

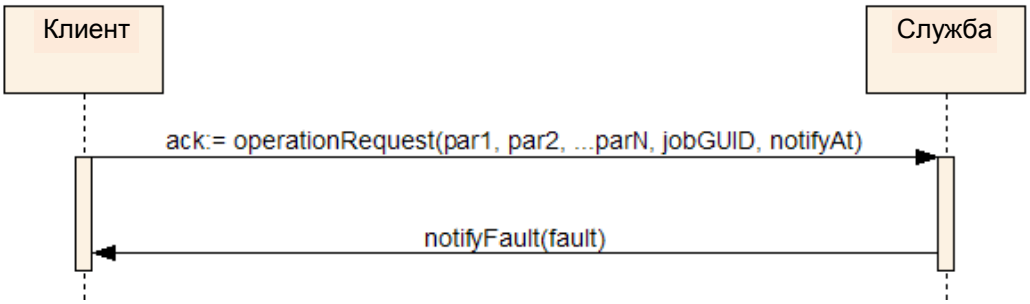


Рис. 15: Обработка сообщения об отказе во время выполнения долговыполняемого процесса

Получатель уведомления об ошибке должен иметь интерфейс, определенный службой. Например, для WSDL/SOAP, в случае службы *Transform Media* получатель реализует операцию *notifyFault,* указанную *TransformMediaNotification*. См. Рис. 16.

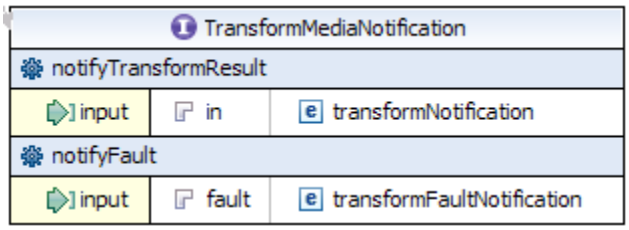


Рис. 16: Пример обработки сообщения об отказе с уведомлением

8.1.5 Восстановление после отказа

Отказ услуги во время выполнения рабочего процесса должен быть восстанавливаемым процессом. Структура предусматривает повторную передачу отказавшей услуги, т.к. во многих случаях причина отказа легко устраняется человеческим вмешательством. В этом сценарии рабочий процесс прерывается по инициативе отказавшей службы и может восстановиться вручную после ее восстановления.

Структура также предусматривает определение механизма компенсации в рабочем процессе для возврата в строго определенное состояние, когда восстановление после отказа не выполняется простым механизмом повторной передачи.

8.1.6 Очередь заданий

Служба может реализовать очередь для поддержки множества одновременных запросов. При реализации запросы на задания ставятся в очередь в порядке приоритета и прибытия. Служба снимает задания с очереди и обрабатывает их по одному. Можно снять с очередь сразу множество заданий, если служба поддерживает выполнение множества одновременных заданий (многопоточность).

На Рис. 17 показаны состояния, связанные с очередью заданий, и переходы, инициированные командами очереди, выданными службе. Эти состояния связаны с процессами, которые ставят запросы в очередь (принимают новые задания) и снимают их с очереди (начинают выполнение). Состояние *Started* означает, что активны процессы и постановки, и снятия с очереди. *Locked* означает, что задания нельзя ставить в очередь, но можно снимать. *Stopped* означает, что задания ни ставятся, ни снимаются с очереди.

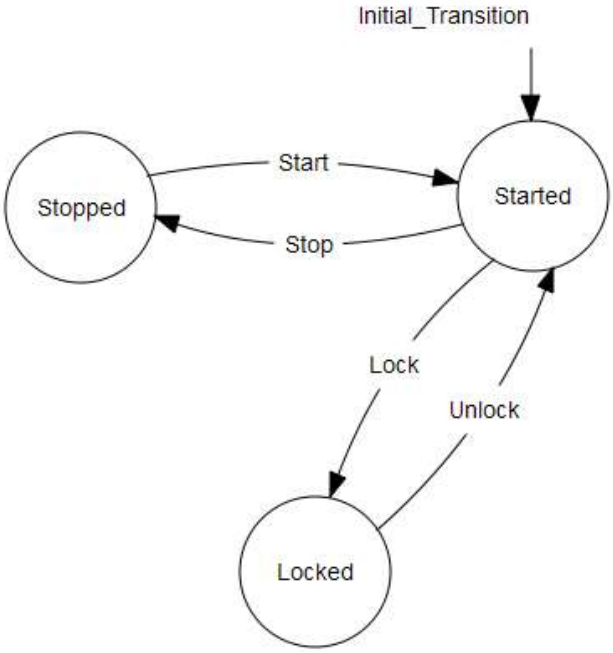


**Рис. 17: Состояния, связанные с очередью заданий,
и переходы, инициированные командами очереди**

Служба должна реализовать операцию *manageQueue* для обработки команд очереди. Если служба не реализует очередь, она должна ответить кодом ошибки “Feature Not Supported” (SVC_S00_0015).

Очередь заданий может иметь максимальный размер, и после его достижения задания больше не принимаются. Отказ, означающий заполнение очереди, должен отбрасываться службой, когда клиент посылает новый запрос на задание (SVC_S00_0008).

8.1.7 Приоритет выполнения заданий

Служба должна выполнять запросы в порядке приоритета. Приоритет задания указан в сообщении с запросом. Задания с более высоким приоритетом имеют преимущество перед заданиями с низким приоритетом.

Структура определяет уровень приоритета запроса. Перечень возможных значений приоритета: «низкий», «средний», «высокий», «срочный» и «немедленный».

- Новое задание с «низким» приоритетом должно помещаться в конец очереди.
- Новое задание со «средним» приоритетом должно помещаться для выполнения перед заданиями с «низким» приоритетом, но после имеющихся заданий со «средним» приоритетом.
- Новое задание с «высоким» приоритетом должно помещаться перед заданиями со «средним» и «низким» приоритетом, но после имеющихся заданий с «высоким» приоритетом.
- Новое задание со «срочным» приоритетом должно помещаться перед заданиями с «высоким», «средним» и «низким» приоритетом, но после имеющихся заданий со «срочным» приоритетом.
- Новое задание с «немедленным» приоритетом должно выполняться, как только получен запрос на задание.

Если служба может обработать лишь одно задание за раз, а выполняемое задание имеет более низкий приоритет, чем новое, то текущее задание должно выполняться до завершения и лишь потом должна начаться обработка нового задания.

За исключением заданий с «немедленным» приоритетом, расстановка приоритетов должна влиять только на задания в очереди. Изменение порядка текущих заданий требует явной операции, которая не входит в рамки данной спецификации.

8.1.8 Ссылки на медиа

Медиа службы FIMS работают с сущностью контента. Для описания возможности взаимодействия сущности нужен уникальный механизм ссылок на сущность и метаданные. Этот раздел описывает объектную модель, которая обеспечивает простой механизм для поддержки требований FIMS: Business Media Object (BMObject). BMObject обеспечивает информацию бизнес-уровня о медиа контенте, который обменивается службами FIMS (например, местонахождение медиа, формат и размер). BMObject обеспечивает видимость медиа контента на уровне бизнес-процессов, т.е. термин «бизнес» означает, что это объект бизнес-уровня, который не пересекается с существующими стандартами для описания медиа и метаданных, а обеспечивает удобный механизм для прозрачного взаимодействия с такими стандартами. BMObject может быть простым как URL для медиа контента и описания, и при этом он очень гибкий, т.к. может представлять сложные модели контента и обеспечивает механизм для транспортировки метаданных при необходимости.

Модель BMObject обеспечивает простую ссылку на файлы сущности контента на основе BMContentType и минимальный набор свойств описательных и технических метаданных на базе EBU Tech 3293 (EBUCore Metadata Set). Это может использоваться структурой FIMS для принятия решений по бизнес-правилам (например, выбор наилучшей службы преобразования медиа для задания на основании формата или размера контента).

Будущие расширения структуры могут расширить BMObjectType другими видами коллекций контента. Например:

- Разные редакторские монтажи одной телепрограммы.
- Монтаж только последовательностей для транскодера.
- Монтажные листы.
- Сложное преобразование отношений между входящим и исходящим контентом.

8.1.8.1 Ссылка на частичные медиа

- Спецификация FIMS 1.0 для передачи и преобразования медиа не требует ссылки на частичные медиа
- Захват обеспечивает InPoint/OutPoint, что позволяет определение частичного захвата медиа с тайм-кодами.
- Дополнительные услуги могут требовать возможности ссылки на частичные медиа.

8.1.9 Задания

Важный аспект управления долговыполняемыми медиа операциями – возможность проверки статуса и взаимодействия с поставленными заданиями. Например, для операций транскодирования могут быть выполнены следующие задачи:

- Проверка статуса задания или списка заданий
- Отмена активного задания
- Пауза активного задания
- Остановка активного задания
- Изменение приоритета задания
- Очистка очереди
- Блокировка / разблокировка очереди

Внедрение общего интерфейса для операций статуса позволяет использовать общие инструментальные средства начальных этапов для управления медиа службами и дает возможность взаимодействия с текущими задачами от клиента или рабочего процесса. Типы управления Queue и Job обеспечивают механизм запроса и управления статусом заданий и их очередей.

8.1.10 Ошибки

Базовая схема определяет тип отказа, который можно расширить классами услуг FIMS для обеспечения кодов ошибок, специфичных для классов услуг. Одно из преимуществ использования абстрактного класса медиа службы – это определение общего набора ошибок. Это определение позволяет клиенту-запросчику реализовать разную логику обработки ошибок для каждого провайдера услуг. При таком подходе клиент может реализовать общую логику компенсации или обработки ошибок для всех провайдеров услуг, поддерживающих спецификацию услуг FIMS.

8.2 Обзор сервисного интерфейса

В этом разделе даны две общие функции сервисных интерфейсов: временные ограничения и профили.

8.2.1 Временные ограничения

8.2.1.1 Концепция

Это временные ограничения в течение выполнения задания, указанные в сообщении с запросом на услугу. Временные ограничения состоят из следующих четырех параметров:

1) startJob

startJob – один из элементов JobType, и применим к любой службе. Он определяет системное время, когда должно начаться задание. Обычно указывается время, когда сообщение с запросом извлекается из очереди и перемещается в состояние выполнения.

Есть три типа указания параметра времени в startJob:

- NoWait: выполнить немедленно
- Time: время начала
- Latest: самое позднее время, когда может начаться процесс, в свойстве startProcess.

Поддержка типов Time и/или Latest означает, что служба поддерживает специальную очередь с возможностью планирования времени начала заданий в соответствии со значением startJob. Служба, которая поддерживает тип Time или Latest, должна декларировать это в описании услуги.

2) startProcess

startProcess – параметр для служб, которые должны управлять функцией передачи потоков в реальном времени, например, Capture или Playout .Он определяет системное время в начале процесса передачи потоков.

Есть четыре типа указания tпараметра времени startProcess, который может требовать покадровой точности:

- NoWait: выполнить немедленно
- Time: время начала
- TimeMark: время, когда обнаружен TimeMark, встроенный в сущность, например, тайм-код
- ServiceDefinedTime: время, определяемое службой

Служба, которая поддерживает тип Time и/или TimeMark, должна декларировать это в описании услуги.

3) stopProcess

stopProcess – параметр для служб, которые должны управлять функцией передачи потоков в реальном времени, например, Capture или Playout Он определяет время, когда процесс передачи потока должен остановиться.

Есть пять типов указания tпараметра времени stopProcess, который может требовать покадровой точности:

- OpenEnd: время, когда получена команда остановки
- Time: время остановки
- TimeMark: время, когда обнаружен TimeMark, встроенный в сущность, например, тайм-код
- Duration: момент, когда прошло определенное время с начала startProcess
- ServiceDefinedTime: время, определяемое службой

Служба, которая поддерживает тип Time, TimeMark /или Duration, должна декларировать это в описании услуги.

4) finishBefore

finishBefore – один из элементов JobType, применимый к любой службе. Он определяет время, когда должно быть выполнено задание.

finishBefore также определяет срок выполнения задания. Например, в случае указания типа TimeMark в stopProcess он может использоваться как время таймаута. Если finishBefore превышено до завершения задания, служба должна уведомить систему оркестровки сообщением "SVC_S00_0016: Deadline passed".

Если служба не может уложиться во временные рамки, она должна уведомить систему оркестровки кодом ошибки SVC_S00_0017: “Time Constraints in request cannot be met”.

Служба, которая может контролировать время выполнения задания с помощью finishBefore, должна декларировать это в описании услуги.. Даже если служба сама не поддерживает функцию контроля времени выполнения задания, система оркестровки должна определять finishBefore как крайний срок.

8.2.1.2 Варианты применения временных ограничений

В таблице 1 показаны возможные варианты применения с комбинацией "startProcess" и "stopProcess".

Таблица 1: Временные ограничения с использованием комбинации "startProcess" и "stopProcess"

startProcess	stopProcess	Описание
NoWait	OpenEnd	Начать как можно скорее, остановить при получении команды стоп (manageJobRequest).
	TimeMark	Начать как можно скорее, остановить при обнаружении определенного timeMark.
	Time	Начать как можно скорее, остановить в указанное время.
	ServiceDefined Time	Начать как можно скорее, остановить во время, определенное службой.
	Duration	Начать как можно скорее, остановить, когда пройдет указанное время.
TimeMark	OpenEnd	Начать при обнаружении указанного timeMark, остановить при получении команды стоп (manageJobRequest).
	TimeMark	Начать при обнаружении указанного timeMark, остановить при обнаружении указанного timeMark.
	Time	Начать при обнаружении указанного timeMark, остановить в указанное время.
	ServiceDefined Time	Начать при обнаружении указанного timeMark, остановить во время, определенное службой.
	Duration	Начать при обнаружении указанного timeMark, остановить, когда пройдет указанное время.
Time	OpenEnd	Начать в указанное время, остановить при получении команды стоп (manageJobRequest).
	TimeMark	Начать в указанное время, остановить при обнаружении указанного timeMark.
	Time	Начать в указанное время, остановить в указанное время.
	ServiceDefined Time	Начать в указанное время, остановить во время, определенное службой
	Duration	Начать в указанное время, остановить, когда пройдет указанное время.
ServiceDefined Time	OpenEnd	Начать во время, определенное службой, остановить при получении команды стоп (manageJobRequest).
	TimeMark	Начать во время, определенное службой, остановить при обнаружении указанного timeMark.
	Time	Начать во время, определенное службой, остановить в указанное время.
	ServiceDefined Time	Начать во время, определенное службой, остановить во время, определенное службой.
	Duration	Начать во время, определенное службой, остановить, когда пройдет указанное время.

8.2.1.3 Примеры диаграмм последовательности

На Рис. 18 показан пример диаграммы последовательности службы захвата в терминах временных ограничений. Тип источника в этом примере – VTR, а inPoint и outPoint – заданный тайм-код.

Служба получает сообщение с запросом от системы оркестровки. Служба начинает выполнять задание в startJob и выполняет некоторые действия, например, контрольные сигналы в inPoint, начало воспроизведения и обнаружение заданного тайм-кода. Важный момент – инструкция о том, как и когда управлять устройством, не входит в сообщение с запросом. Служба сама управляет устройством в соответствии с такой информацией как startProcess/stopProcess, inPoint/outPoint в сообщении с запросом. Процесс захвата начинается с serviceDefinedTime; он остановится в другом serviceDefinedTime. По окончании процесса завершения задания в систему оркестровки передается CaptureNortification для отчета о завершении.

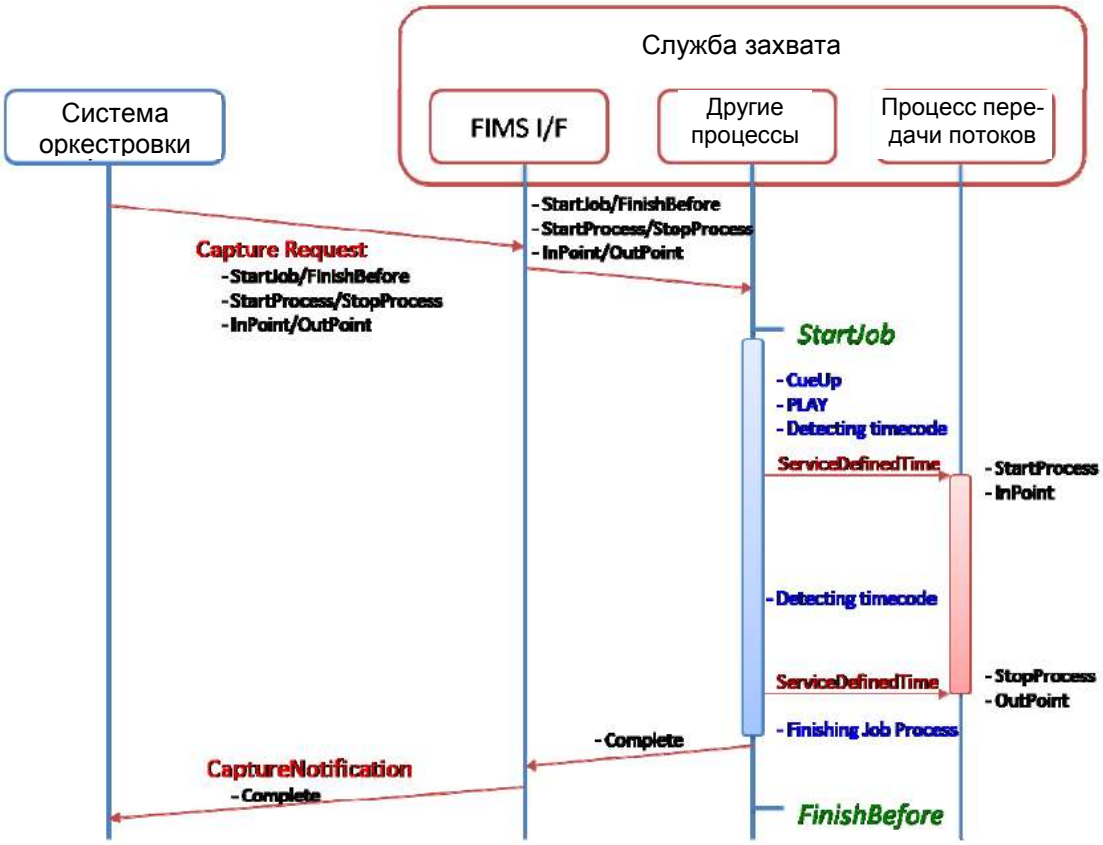


Рис. 18: Пример диаграммы последовательности службы захвата в терминах временных ограничений

На Рис. 19 показан случай, когда временные ограничения в сообщении с запросом не могут быть выполнены по какой-то причине. Это бывает, когда служба получает запрос, во время постановки в очередь или даже во время выполнения. В этом случае служба должна передать в систему оркестровки CaptureFault с кодом ошибки “SVC_S00_0017: Time Constraints in request cannot be met".

Рис. 19: Пример, когда временные ограничения не могут быть выполнены

На Рис. 20 показан случай, когда время, указанное в finishBefore, истекает до завершения процесса захвата. В этом случае служба должна передать в систему оркестровки CaptureFaultNotification с кодом ошибки "SVC_S00_0016 Deadline passed". Передав CaptureFaultNotification, служба должна ждать следующей инструкции от оркестровки, не останавливая процесс.

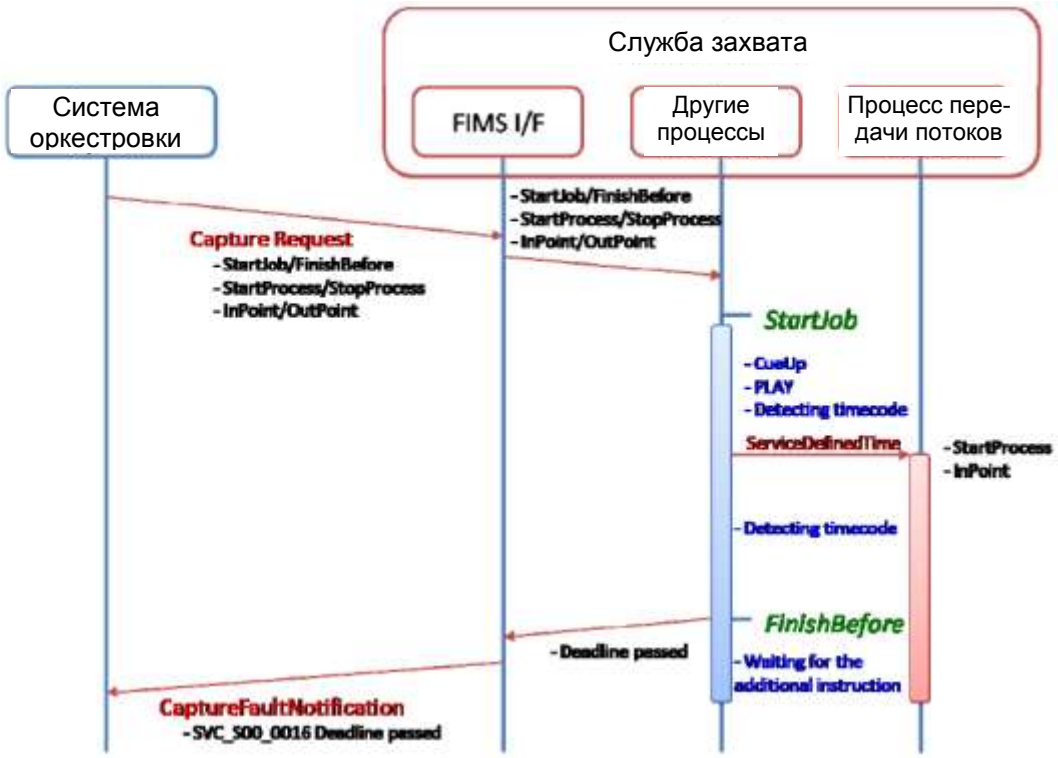


Рис. 20: Пример, когда время, указанное в finishBefore, истекает до завершения

8.2.2 Профили

Запросы в любой сервисный интерфейс FIMS должны включать профиль, который описывает определенные операции, выполняемые в составе данного запроса. Сами профили состоят из параметров, специфичных для данного сервисного интерфейса, и общие элементы, доступные для повторного использования между интерфейсами. Эти многоразовые элементы комбинируются в группы, которые называются Атомами.

Структуры профилей определяются для каждого сервисного интерфейса, определенного FIMS. Профили наследуются из *ProfileType*, который используется для общей основы для всех профилей (Рис. 21).

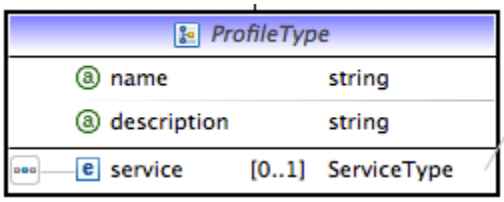


Рис. 21: ProfileType

Профили могут совместно использоваться между заданиями и включаться по ссылке. См. п.6.2.3.

По возможности реализации должны использовать указанный профиль.

На Рис. 22 показан пример запроса с двумя профилями Capture, используемыми для создания основной потоковой сущности (J2K + MXF) и прокси-сущности (AVC + MP4) и выводящими через различные атомы передачи (возможно, в разные места через разный транспорт).

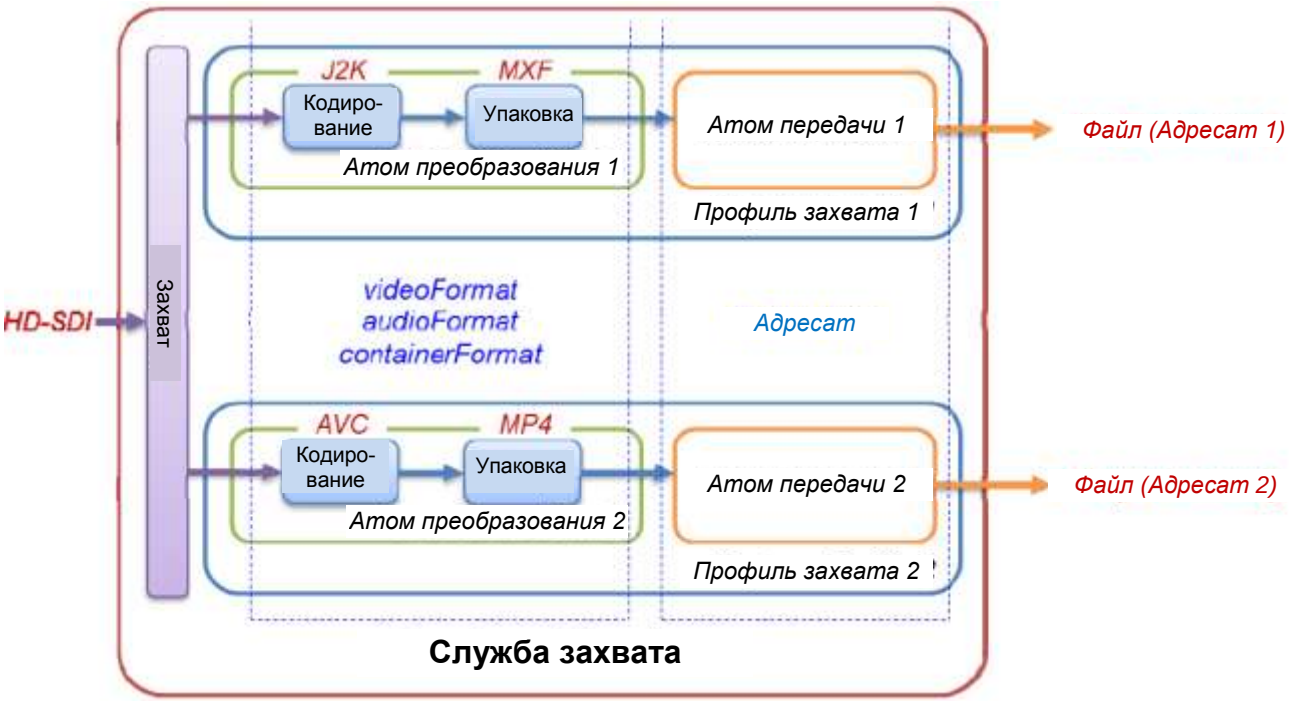


Рис. 22: Пример запроса Capture со множеством профилей

8.3 Сервисные интерфейсы на базе REST

Этот раздел определяет представление REST-значений FIMS в сообщениях REST и определение служб на базе REST. Правила этого раздела – для служб REST, какие файлы WSDL являются службами WSDL/SOAP.

Таблицы определений служб REST следует использовать для определения служб FIMS на базе REST. Эти таблицы должны быть включены в документацию с документацией схемы XML для каждой службы. Каждая таблица ссылается на элементы схем, которые определяют XML или JSON представление тела сообщения REST.

8.3.1 Пространства имен для определения служб REST

Таблица 2 определяет пространства имен и префиксы, которые должны использоваться для определения служб REST в схеме XML.

Таблица 2: Пространства имен для определения служб REST

Название схемы	Префикс	Пространство имен
Base media service	bms	http://base.fims.tv
Base time values	tim	http://baseTime.fims.tv
Transfer media service	tms	http://transfermedia.fims.tv
Transform media service	tfms	http://transformmedia.fims.tv
Capture media service	cms	http://capturemedia.fims.tv
Repository service	rps	http://repository.fims.tv

Пространства имен, перечисленные в таблице, предназначены для служб, определенных с данной версией спецификации. Любые службы, использующие эту структуру, но не являющиеся частью данной версии FIMS, должны указывать пространство имен и префикс, используемые для определения преобразования службы на базе REST.

8.3.2 Таблицы определений служб REST

Документация для каждой службы должна включать *REST service definition table,* которая определяет, как операции службы представлены в виде сообщений REST. Таблица должна состоять из следующих столбцов:

- **Description** – Описание операции.
- **HTTP method** – HTTP метод, который должен использоваться для сообщения с запросом на операцию, например, GET, POST, DELETE.
- **URI** – URI, который должен использоваться для доступа к службе на базе REST, которая указана относительно конечной точки для службы REST, например, "…/job".
- **Request body** – Ссылка на имя элемента XML и тип элемента XML, которая должна использоваться для представления значения FIMS, которое должно быть телом сообщения с запросом.
- **Success body** – Ссылка на имя элемента XML и тип элемента XML, которая должна использоваться для представления значения FIMS, возвращаемого как тело сообщения с ответом для успешной операции.
- **Failure body** - Ссылка на имя элемента XML и тип элемента XML, которая должна использоваться для представления значения FIMS, возвращаемого как тело сообщения с ответом для неудавшейся операции.
- **HTTP request headers** – Кроме требуемого заголовка с номером версии (п.6.1.4.1) и стандартных заголовков HTTP, любые специфичные для FIMS заголовки HTTP, которые должны быть включены с сообщением запроса.
- **HTTP response headers** – Кроме требуемого заголовка с номером версии (п.6.1.4.1) и стандартных заголовков HTTP, любые специфичные для FIMS заголовки HTTP, которые должны быть включены с сообщением ответа.

Код состояния сообщения с ответом должен быть либо кодом переадресации (диапазон 300), либо следующим:

- 200 – OK – Успешная операция, синхронный или асинхронный ответ;
- 201 – Created – Успешная операция, в результате которой создан новый объект;
- Неудачная операция, содержащая сообщение об отказе с кодом состояния, установленным на значение, определенное для данного отказа. См. общую и специфическую документацию FaultType в схемах XML.

Клиенты FIMS REST должны следовать ответам HTTP с переадресацией согласно спецификациям HTTP.

Любые параметры URI, заголовки HTTP и аргументы запросов должны быть описаны, как в Таблице 3. Параметры URI идентифицированы в таблицах определения услуг REST с использованием фигурных скобок {} и требуют замены соответствующими параметрами до использования. Параметры запроса будут обозначены *курсивом*.

Таблица 3: Параметры шаблонов таблиц FIMS REST

Параметр	Описание	Значения
{jobID}	Уникальный идентификатор задания.	bms:ResourceIDType
{urlToJob}	При успешном создании задания это значение – полный путь к заданию	http(s)://{server name}/job/{jobId}
{version}	Определенная версия контента, на которую есть ссылка в хранилище. Отсутствие версии означает ссылку на последнюю версию.	
{repository credentials}	Данные авторизации для хранилища.	http header = X-FIMS-UserName http header = X-FIMS-Password http header = X-FIMS-SessionToken
{lock token}	Блокировка, полученная из хранилища.	HTTP header = X-FIMS-LockTokenId
{super token}	Мастер-блокировка, полученная из хранилища.	HTTP header = X-FIMS-SuperLockTokenId
{*JobFilterCriteria*}	Параметры фильтра заданий см. в Таблице 4.	

Параметры запроса, которые можно использовать для выбора заданий в услугах запроса заданий, указаны в Таблице 4.

Таблица 4: Критерии фильтра заданий

Параметр	Описание	Тип
jobInfoSelectionType	Возврат только обязательных атрибутов или всех атрибутов задания	"all" или "mandatory"
toDate	Перечисляемые задания должны начинаться с или до даты, указанной в этом поле.	dateTime
fromDate	Перечисляемые задания должны начинаться с или после даты, указанной в этом поле.	dateTime
includeQueued	Флажок для индикации задания или заданий в очереди.	Булево "true" или "false"
includeFinished	Флажок для индикации задания или заданий в состоянии 'Completed', 'Stopped' или 'Cleaned'.	Булево "true" или "false"
includeActive	Флажок для индикации задания или заданий в состоянии 'Running', 'Paused' или 'Unknown'.	Булево "true" или "false"
includeFailed	Флажок для индикации задания или заданий в состоянии 'Failed'.	Булево "true" или "false"
maxNumberResults	Максимальное число результатов в списке.	Целое положительное число

8.3.3 XML представление сообщения REST

Таблицы определений услуг REST и таблицы уведомлений на базе REST определяют тип значений FIMS для передачи в теле сообщений по ссылке на элементы и типы схемы XML. XML схема FIMS определяет тип данных значений FIMS и их преобразование в последовательную форму в виде XML. В теле сообщений REST с установкой "Content-Type" на "application/xml" должны быть значения, преобразованные из параллельной формы в последовательную и обратно из XML согласно опорному элементу XML схем FIMS.

8.3.4 JSON представление сообщения REST

8.3.4.1 Преобразование представления JSON в XML и обратно

В теле любых сообщений REST с установкой "Content-Type" на "application/json" должны быть значения FIMS, преобразованные в последовательную форму в JSON в соответствии с первым преобразованием значения в XML согласно п.8.3.3 и затем с применением правил данного раздела. И наоборот, правила данного раздела должны применяться в преобразовании значения JSON обратно в XML и затем в параллельную форму согласно правилу п.8.3.3.

Примечание: Механизм преобразования для JSON в XML и обратно дан как средство для определения представления JSON с использованием существующей схемы XML. Реализации не должны применять этот подход и могут использовать более эффективные средства преобразования в последовательную и параллельную форму.

Примечание: Пример кода, который делает XML в преобразование JSON и обратно, дан в соответствующей опорной реализации.

В этом разделе пара значений имен для объекта в документе JSON называется *полем* JSON.

Для взаимодействия служб FIMS самое главное, чтобы XML схема FIMS использовала пространства имен XML для различия между полями: определенными базовыми схемами; специфичными для текущих служб; специфичными для будущих служб; свойствами расширений. Поскольку JSON не имеет явной концепции пространств имен, JSON документы FIMS должны следовать правилам пространств имен XML в именах полей и передавать все атрибуты "xmlns", ожидаемые в соответствующем документе XML. Имена полей JSON должны по возможности использовать префиксы для однозначной передачи явного пространства имен используемого свойства.

Примечание: Правила пространства имен XML будут соблюдаться в результате применения правил данного раздела. Структуры, которые поддерживают преобразование в последовательную форму в JSON и обратно и могут использовать различные подходы в работе с пространствами имен. Необходимо уделить внимание – и можно написать дополнительный код – для гарантии полного взаимодействия между службами JSON. Например, префикс имени поля может меняться в зависимости от текущей декларации пространства имен "xmlns".

8.3.4.2 Документы

Ровно один XML документ FIMS должен быть представлен ровно одним документом JSON.

Корневой объект документа JSON должен содержать ровно одно поле. Это поле должно представлять значение корневого элемента эквивалентного документа XML через применение элемента *rule*.

8.3.4.3 Правило элементов

Элементы простого типа в XML, ограниченные XSD атрибутом *maxOccurs,* установленным на 0 или 1, должны преобразовываться один в один в поля JSON. Имя поля должно быть tквалифицировано в пространстве имен с префиксом или не квалифицировано в пространстве имен на основе контекста в зависимости от контекстного использования в соответствующем документе XML. Значение поля должно определяться применением *правил преобразования простого типа*.

Элементы сложного типа, ограниченные XSD атрибутом *maxOccurs,* установленным на 0 или 1, должны быть представлены в JSON как поля их родительского объекта объекта type. Путем применения элемента или *правил атрибута* каждый из дочерних узлов элемента, будь то атрибуты или сами элементы, должен преобразовываться с дочерние поля объекта JSON.

Элементы любого типа, ограниченные XSD атрибутом *maxOccurs,* установленным на значение более 1, включая и те, что определены как несвязанные, должны быть представлены как массивы JSON, где каждое значение массива должно быть результатом применения правила элементов. Порядок субэлементов в последовательности XML должен соответствовать порядку эквивалентных значений в массиве JSON.

Примечание: По определению, поля объекта JSON не упорядочены, а субэлементы элемента FIMS имеют определенный порядок. Возможно, что приложению, переводящее JSON документ FIMS в XML документ FIMS, придется переупорядочить поля объектов JSON в соответствии с XML схемой FIMS перед подтверждением.

Например:

1. Элемент XML
```
<Fred>ginger</Fred>
```
представлен в JSON как
```
"Fred":"ginger"
```
.

2. Элемент XML
```
<Fred>2</Fred>
```
представлен в JSON как
```
"Fred":2
```
.

3. Элемент XML
```
<bms:Fred>ginger</bms:Fred>
```
представлен в JSON как
```
"bms:Fred":"ginger"
```
.

4. Последовательность элементов XML
```
<fromage>cheddar</fromage><fromage>Stilton</fromage>
```
представлена в JSON как
```
"fromage":["cheddar","Stilton"]
```
.

5. Элемент XML сложного типа
```
<thing><child>stuff</child></thing>
```
представлен в JSON как
```
"thing":{"child":"stuff"}
```
.

6. Элемент XML сложного типа
```
<thing about="that"/>
```
представлен в JSON как
```
"thing":{"@about":"that"}
```
.

8.3.4.4 Правило атрибутов

Все атрибуты XML должны быть значениями полей JSON, входящими в родительский элемент, который их определяет. Имя поля должно быть таким же, что используется для XML с символом '@' («собака»), добавленным к началу. Если имя атрибута XML является квалифицированным пространством имен, то имя поля JSON должно быть квалифицировано пространством имен. Если имя XML не имеет квалификации пространства имен, то имя JSON не должно иметь квалификации пространства имен. Тип значения должен преобразовываться в соответствии с *правилами преобразования простого типа*.

Например:

1. Атрибут XML
```
<... "fred"="ginger" ...>
```
является полем JSON
```
..., "@fred":"ginger", ...
```
2. Атрибут XML
```
<... "bms:fred"="ginger" ...>
```
является полем JSON
```
..., "@bms:fred":"ginger", ...
```
3. Атрибут XML
```
<... "fred"="3" ...>
```
является полем JSON
```
..., "@fred":3, ...
```

8.3.4.5 Правила преобразования простых типов

Таблица 5 определяет преобразование между простыми типами в схемах XSD и ограниченный набор простых типов данных JSON для значений полей числа, строки и булева значения.

Таблица 5: Преобразование простых типов из типов XSD в типы JSON

Тип XSD	Тип JSON
Булево	JSON булево "true" или "false"
Целое, длинное, двойное, плавающее, короткое, байт и производные, например, неотрицательное целое.	JSON число
строка, NCName	JSON строка
Строка, ограниченная нумерацией или регулярным выражением	JSON строка с тем же ограничением
дата, датаВремя, время, длительность	JSON строка в том же формате.

Примечание: Тип поля в JSON нельзя определить только по его значению. В частности, необходимо соблюдать осторожность при преобразовании числовых значений JSON для получения достаточного числа ,байтов. Некоторые значения FIMS бывают длинными или двойными для обеспечения достаточного места для больших или точных значений, например, преобразования частоты дискретизации звука или частоты кадров.

Приложение 1: Будущие перспективы (информативно)

A1. Конвейерные медиа службы

Конвейерные медиа службы состоят более чем из одной существующей службы для реализации функций и/или характеристик, которых нельзя достичь использованием просто отдельных служб

Как говорилось в п.8.2.2, конвейерная медиа служба внутри службы может быть реализована с помощью профилей. На Рис. 22 показан пример расширенного профиля, где два профиля Capture (Transform) используются для создания основной потоковой сущности (J2K + MXF) и прокси-сущности (AVC + MP4) с AV Process.

AV Process добавляется в transformAtom, чтобы включить обработку A/V сущности во время захвата.

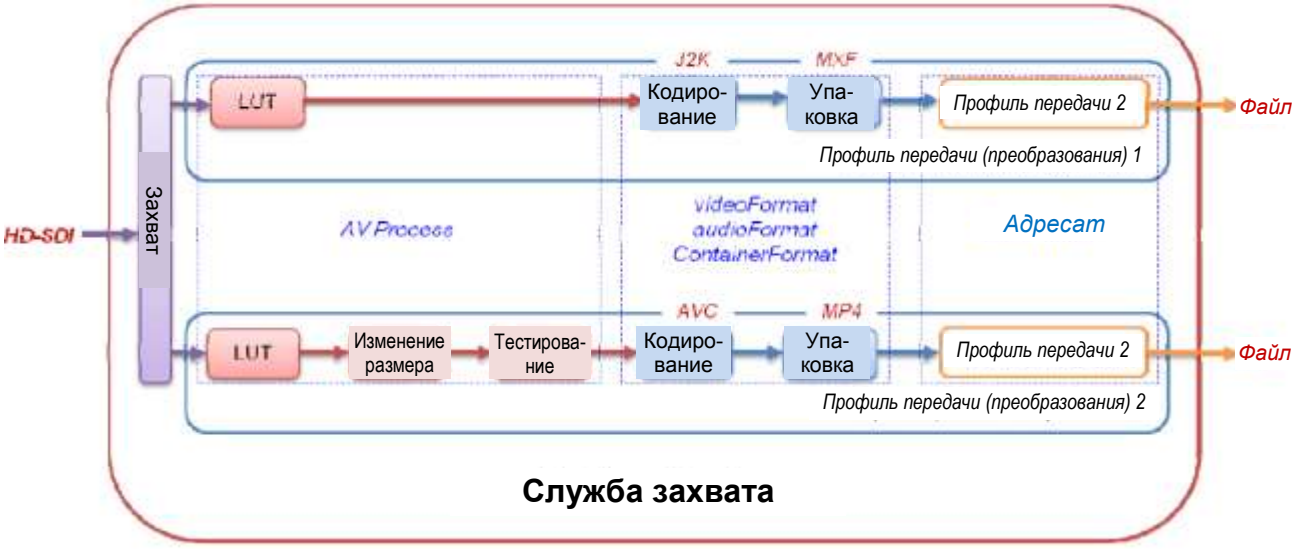


Рис. 22: Пример конвейерной медиа службы