IBM Business Process Manager
версия 8 выпуск 5

Обзор IBM Business Process Manager

Книги PDF и справочная система Information Center

Книги в формате PDF доступны для удобства печати и чтения. Последняя версия информации доступна в справочной системе Information Center.

Книги PDF содержат тот же набор информации, что и справочная система Information Center. Некоторые ссылки в книгах в формате PDF созданы для использования в справочных системах Information Center и могут давать сбои.

Документация PDF становится доступной через квартал после основного выпуска справочной системы Information Center, например версии 7.0 или 7.5.

Документация в формате PDF обновляется с меньшей частотой по сравнению с информацией в справочной системе Information Center, однако чаще чем руководства Redbooks. Книги PDF обновляются по мере накопления достаточного количества изменений.

Содержание

Глава 2. Дополнительные сведения об основных концепциях 181

Глава 1. Начало работы с IBM Business Process Manager

В этом разделе описаны возможности, которые предоставляет IBM® Business Process Manager для управления бизнес-процессами, а также взаимосвязи между различными этапами управления бизнес-процессами, например, между созданием и развертыванием приложений процессов.

Приложение процессов является основным контейнером для процессов и их компонентов в IBM Business Process Manager. Разработчики процессов создают приложения процессов в средах разработки и создания приложений, и в них могут быть включены службы, задачи и артефакты, необходимые для их выполнения.

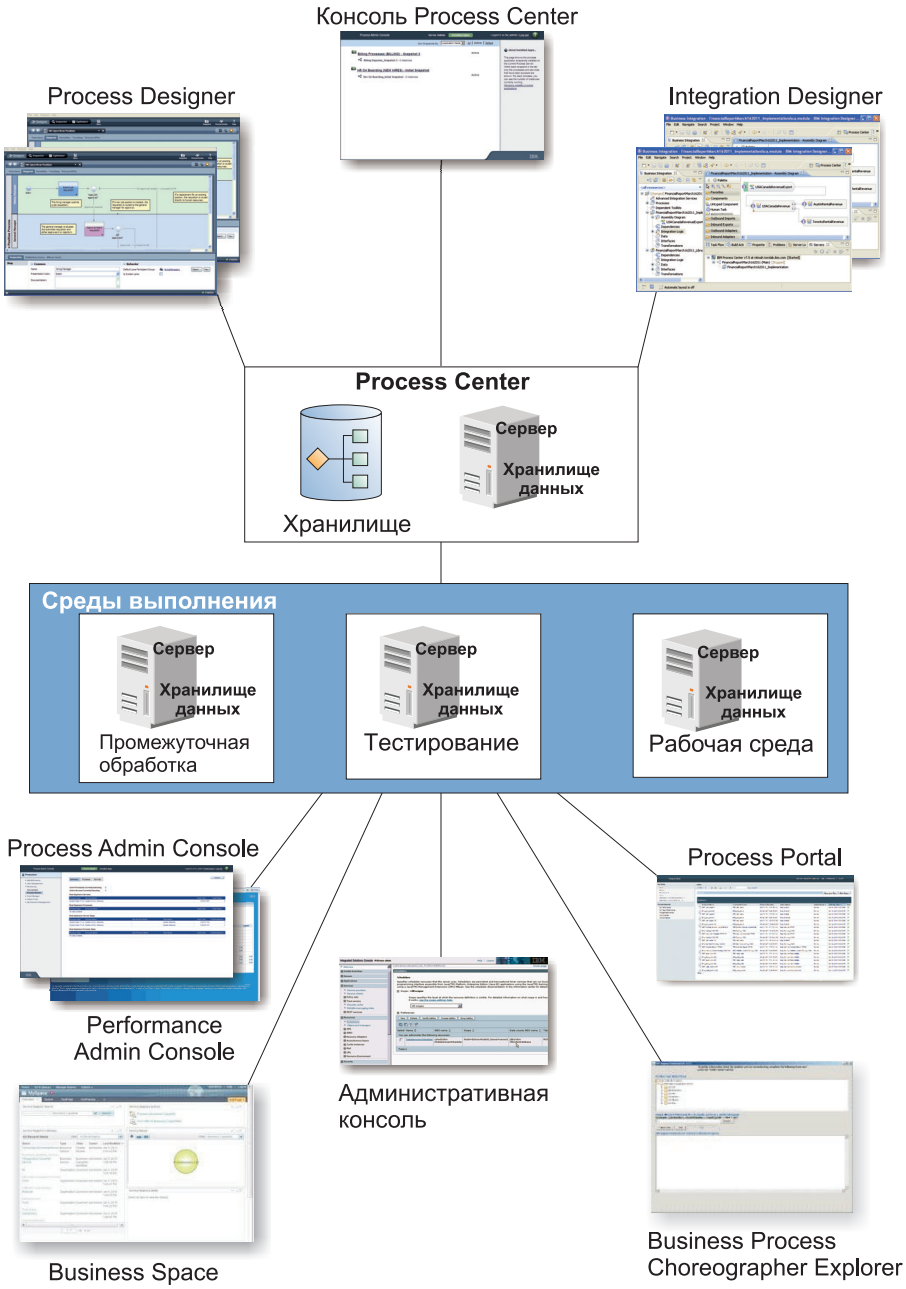
Расширенные службы интеграции реализуются в IBM Integration Designer и связываются с приложениями процессов. Из Process Center приложения процессов развертываются в Process Server, который представляет собой среду выполнения процессов для IBM Business Process Manager.

Точно так же, созданные в Integration Designer автоматизированные процессы могут использовать неавтоматизированные службы, разработанные в IBM Process Designer.

Обзор продукта

IBM Business Process Manager - это платформа комплексного управления бизнес-процессами, предоставляющая общую картину для управления бизнес-процессами. Она предоставляет инструментарий и среду выполнения для проектирования, выполнения, отслеживания и оптимизации процессов, а также базовую поддержку системной интеграции. Продукт может быть настроен для поддержки различных уровней сложности задачи и управления бизнес-процессами.

Компоненты IBM Business Process Manager предоставляют едино хранилище BPM, инструментарий для авторов, администраторов и пользователей, а также платформу для среды выполнения. Продукт может быть настроен для поддержки различных уровней сложности задачи и управления бизнес-процессами. На следующей диаграмме показана типичная конфигурация IBM Business Process Manager:



- Из среды создания IBM Process Designer или IBM Integration Designer разработчики подключаются к IBM Process Center. В одном из этих средств разработки приложений на основе графического пользовательского интерфейса разработки могут создать, протестировать, отладить и развернуть бизнес-приложения. Выберите одно или другое средство в зависимости от типа разрабатываемого приложения. Могут также возникнуть случаи, в которых использование обоих средств приносит значительные преимущества.
- С помощью среды разработки приложений Process Designer и Integration Designer проектировщики процессов и служб создают развертываемые приложения процессов и инструментарии с возможностью повторного использования. Приложения процессов содержат модели и реализацию служб, в том числе все требуемые вспомогательные файлы. Приложения процессов хранятся в хранилище Process Center, поэтому их можно использовать совместно.
- Process Center включает в себя два сервера: сервер Process Center и сервер Performance Data Warehouse. Эти серверы позволяют разработчикам, работающим с Process Designer, выполнять приложения процессов и

сохранять данные статистики для тестирования и воспроизведения во время разработки. Performance Data Warehouse регулярно извлекает отслеживаемые данные из Process Server или сервера Process Center.

- Process Center также поддерживает многочисленные административные функции. Из консоли Process Center администраторы устанавливают приложения процессов, готовые к установке, тестированию и работе на сервере процессов. Администраторы могут также управлять выполняющимися экземплярами приложений процессов в настроенных средах.
- Приложения процессов можно установить на сервере процессов в промежуточном режиме, режиме тестирования или рабочем режиме. Среды выполнения поддерживают процессы BPMN (Business Process Model and Notation) версии 2.0. IBM Business Process Manager Advanced также поддерживает процессы на Языке выполнения бизнес-процессов (BPEL).
- Из консоли администратора процессов и консоли администратора производительности администраторы могут управлять всеми серверами выполнения. Используйте административную консоль процессов для управления сервером Process Center и серверами процессов в средах выполнения. Используйте консоль Performance Admin Console для определения узких мест производительности и захвата вспомогательных данных для дальнейшего анализа.
- Используйте административную консоль для создания и управления объектами, такими как ресурсы, приложения и серверы. Кроме того, с помощью административной консоли можно просмотреть сообщения о продукте.
- Используйте Business Space для создания пользовательских бизнес-пространств, которые предоставляют виджеты для отслеживания и администрирования различных аспектов системы. Например, можно отслеживать бизнес-операции, службы и работоспособность системы или администрировать стратегии передачи и бизнес-календари. Можно также создать бизнес-пространство с виджетами Управление неавтоматизированными задачами и использовать его для участия в бизнес-процессах.
- С помощью Process Portal участники процессов могут подключаться к серверу Process Center или к Process Server в любой настроенной среде выполнения в зависимости от того, находится ли процесс в разработке, тестируется или был установлен рабочей среде.
- Процессами на Языке выполнения бизнес-процессов (BPEL) можно управлять в Business Process Choreographer Explorer или в Business Space.

Конфигурации IBM Business Process Manager

Различные конфигурации IBM Business Process Manager применяются в различных начальных ситуациях или на разных этапах программы управления бизнес-процессами в компании.

Таблица 1. Конфигурации IBM Business Process Manager

Конфигурация	**Этап**
Advanced	**Преобразование** Полный набор функций по управлению бизнес-процессами • Расширенная поддержка средств крупномасштабной автоматизации процессов • Встроенные компоненты SOA для расширенной интеграции служб и организации взаимодействия в масштабах предприятия
Standard	**Программа** Настраивается для типичных проектов управления бизнес-процессами • Для программ, вовлекающих несколько проектов с высокой бизнес-нагрузкой • Базовые возможности системной интеграции • Быстрая отдача и улучшенная эффективность работы пользователей

Таблица 1. Конфигурации IBM Business Process Manager (продолжение)

Конфигурация	**Этап**
Express	**Проект** Настраивается для первого проекта управления бизнес-процессами • Быстрая отдача и улучшенная эффективность работы пользователей • Низкая начальная цена • Простая установка и настройка

Варианты конфигурации IBM Business Process Manager

В этом разделе описаны продукты IBM и их функциональные возможности для управления бизнес-процессами. Выберите продукт, наиболее подходящий для вашего предприятия.

IBM Business Process Manager - это единая платформа BPM, которая объединяет в себе функциональные возможности систем, ориентированных на взаимодействие людей и интеграцию ИТ-систем. Различные конфигурации продукта отвечают различным задачам предприятия, и пользователь может выбрать наиболее подходящую. Конфигурации продукта могут сочетаться для среды коллективной разработки приложений и среды выполнения, развернутой в сети.

Таблица 2. Варианты конфигурации IBM Business Process Manager

Функция	**Расш.**	**Станд.**	**Express**
Выполнение, совместимое с WebSphere Lombardi Edition	X	X	X
Проектировщик процесса (BPMN)	X	X	X
Коллективное редактирование / немедленное воспроизведение	X	X	X
Интерактивные пользовательские интерфейсы "гида процессов"	X	X	X
Правила процесса на основе ODM	X	X	X
Process Portal	X	X	X
Мониторинг и отчеты в реальном времени	X	X	X
Анализ производительности и оптимизация	X	X	X
Performance Data Warehouse	X	X	X
Process Center / единое хранилище ресурсов	X	X	X
Неограниченное число авторов и пользователей	X	X	200 пользователей / 3 автора
Высокая готовность: кластеры и отсутствие ограничений на число ядер	X	X	• 4-ядерный, рабочий • 2-ядерный, разработка • Без кластера
Выполнение, совместимое с WebSphere Process Server	X		
Integration Designer (BPEL / SOA)	X		
Встроенная Enterprise Service Bus (ESB)	X		
Поддержка транзакций	X		
Адаптеры интеграции	X		
Гибкий пользовательский интерфейс бизнес-пространства	X		
Расширенная поддержка платформы (Linux на System z, IBM AIX, Solaris)	X	X	*Примечание

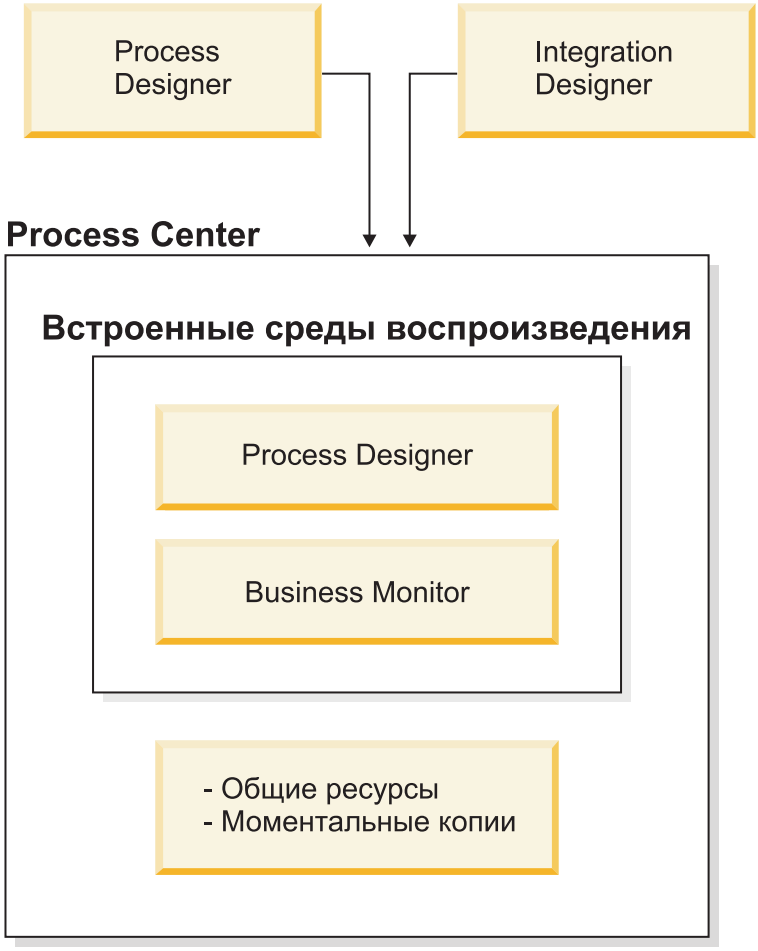
Примечание: IBM BPM Express поддерживается в AIX только для клиентов IBM Master Data Management (MDM).

Хранилище Process Center

В состав Process Center входит хранилище для всех процессов, служб и других ресурсов, создаваемых в средах создания IBM Business Process Manager, Process Designer и Integration Designer.

Продукт Process Center - это компонент программного обеспечения, который выполняется в качестве сервера, где Process Designer и Integration Designer совместно используют ресурсы. Фактически это обеспечивает коллективную разработку бизнес-процессов и высокую степень взаимодействия.

На следующем рисунке показано несколько взаимосвязанных компонентов, используемых при создании сложных бизнес-процессов.



В консоли Process Center доступны инструменты, необходимые при обслуживании хранилища.

- С помощью консоли Process Center можно создавать приложения процессов и комплекты инструментов и предоставлять к ним доступ другим пользователям.
- Пользователи создают среде разработки приложений модели процессов, службы и другие ресурсы в приложениях процессов.
- В состав Process Center входят два сервера - сервер Process Center и сервер Performance Data Warehouse. Process Center Server позволяет разработчикам, работающим в Process Designer, выполнять приложения

процессов и сохранять данные статистики для тестирования и воспроизведения во время разработки. Performance Data Warehouse регулярно извлекает отслеживаемые данные из Process Server и Сервера Process Center.

- С помощью консоли Process Center администраторы устанавливают готовые для тестирования или работы приложения процессов на серверы процессов в этих средах.
- С помощью консоли Process Center администраторы управляют запущенными экземплярами приложений процессов в настроенных средах.

Консоль Process Center предоставляет удобное расположение для создания и управления высокоуровневыми контейнерами, такими как приложения процессов и инструменты. Тем администраторам, которые не работают активно с панелью Designer, консоль Process Center предоставляет структуру, в которой разработчики и аналитики BPM могут компоновать свои процессы и их реализации. Еще одна основная задача администраторов - управление доступом к хранилищу Process Center путем настройки соответствующих прав доступа для пользователей и групп.

Пользователи с подходящими правами доступа могут выполнять некоторые административные задачи непосредственно в Process Designer и Integration Designer. Например, если разработчик хочет зафиксировать состояние всех ресурсов проекта на определенном этапе разработки, и у него есть права доступа для записи, он может создать моментальную копию из панели Проектировщик.

Process Server и среды выполнения

Process Server предоставляет единую среду выполнения, которая поддерживает широкий диапазон бизнес-процессов, организацию взаимодействия служб и функции интеграции.

В среде разработки приложений сервер процессов, встроенный в Process Center, позволяет запускать процессы по мере их компоновки. По мере готовности вы можете устанавливать и запускать одни и те же процессы на серверах процессов в средах выполнения. Компонент Business Performance Data Warehouse собирает данные процессов, работающих на серверах процессов. Эти данные можно использовать для улучшения бизнес-процессов.

Консоль администратора процессов служит для управления серверами процессов в среде выполнения. Она поддерживает промежуточные, тестовые и рабочие серверы, а также серверы процессов, входящие в Process Center.

Среды разработки и создания приложений

В состав IBM Business Process Manager Advanced включены две среды создания приложений. IBM Process Designer позволяет эффективно моделировать бизнес-процессы, включающие неавтоматизированные задачи. IBM Integration Designer применяется для создания служб, работающих автономно или вызывающих другие существующие службы, такие как веб-службы, приложения с ресурсами предприятия или приложения, работающие в CICS и IMS.

- “Process Designer”
- “Integration Designer” на стр. 7

Process Designer

Компонент Process Designer доступен во всех выпусках продукта. Вместе с IBM Business Process Manager Advanced предлагается продукт Integration Designer со всеми его редакторами и адаптерами.

Процесс - это главный логический модуль в IBM Business Process Manager. Он является контейнером для всех компонентов определения процесса, включая службы, действия и шлюзы; хронометр, сообщение и события исключительных ситуаций; линии последовательности, правила и переменные. При моделировании процесса создается многоразовое определение бизнес-процесса (BPD). Можно использовать как Process Designer, так и Integration Designer, для создания моделей процессов, которые могут содержать неавтоматизированные задачи.

С помощью Process Designer можно разрабатывать бизнес-процессы. Удобные графические инструменты помогают создать последовательность действий, которые составляют бизнес-процесс, и если впоследствии обстоятельства изменяются, то этот процесс можно перерисовать. Если какая-либо операция требует доступа к большой базе данных или службам, предоставляющим данные для бизнес-процесса, например, информацию о клиентах, то это можно сделать с помощью Integration Designer. Посредством простого интерфейса операция в Process Designer может вызывать службу, созданную в Integration Designer. Такая служба может использовать поток передачи для преобразования, маршрутизации и расширения данных, чтобы адаптировать их к стандартному виду для многих баз данных. Резюмируя, Process Designer работает в основном с бизнес-процессом, а Integration Designer - автоматизированными службами, дополняющими бизнес-процесс. См. раздел Начало работы с IBM Process Designer.

Все проекты Process Designer содержатся в приложениях процессов. Эти приложения процессов и связанные артефакты содержатся в хранилище Process Center. Приложения процессов могут совместно использовать ресурсы, помещенные в комплекты инструментов.

IBM Business Process Manager поддерживает несколько пользовательских интерфейсов, позволяющих моделировать, реализовывать, имитировать и проверять бизнес-процессы. Консоль Process Center позволяет создавать приложения процессов, комплекты инструментов, дорожки и моментальные копии и управлять ими. Можно создавать модели процессов, отчеты и простые службы в Process Designer. Inspector применяется для запуска и отладки процессов. Optimizer позволяет запускать имитацию.

Приложения процессов, разработанные в Process Designer, могут запускаться в любое время на сервере Process Center. Также можно сохранить их моментальную копию, чтобы развернуть ее на Process Server. Это же относится к службам, разработанным в Integration Designeri связанным с приложениями процессов.

Точка расширения функции идентификации при входе в системе настраивается в Process Designer, чтобы открыть точку настройки для логики входа в систему уровня клиента, позволяющую выполнить специальные требования к идентификации со стороны сервера. В ходе идентификации Process Designer обращается к функции идентификации и вызывает логику входа в систему. Точка расширения идентификации может быть предоставлена в формате модуля Eclipse. Инструкции по работе с модулем идентификации приведены в следующем разделе: Установка IBM Process Designer.

Integration Designer

Integration Designer содержит редакторы и вспомогательные средства для создания сложных автоматизированных процессов и служб (таких как модули SCA, передачи и процессы BPEL). Он доступен в составе IBM Business Process Manager Advanced или как автономный набор инструментов для других целей.

IBM Integration Designer спроектирован как универсальная среда разработки для построения интегрированных приложений. Интегрированные приложения не являются простыми. Они могут вызывать приложения из информационных систем предприятия (EIS), запускать бизнес-процессы других отделов или предприятий, а также приложения, написанные внутри или за пределами организации на различных языках для различных операционных систем. Чтобы создать такие компоненты или собрать из них новое интегрированное приложение (то есть приложение, состоящее из набора компонентов), используются визуальные редакторы, в которых компоненты представлены наглядно и отделены от своей реализации. С помощью различных инструментов разработчик может создавать интегрированные приложения, не зная технических деталей реализации каждого компонента.

Инструменты Integration Designer созданы для архитектуры на основе служб. Компоненты, а также интегрированные приложения, запускающие много компонентов, являются службами. Существующие службы соответствуют общепринятым стандартам. Процессы BPEL, которые становятся компонентами, создаются при помощи тех же визуальных инструментов, соответствующих стандарту BPEL.

В парадигме Integration Designer компоненты собираются в модули. Для обмена данных между модулями используются импорты и экспорты. Модули могут совместно использовать артефакты, помещенные в библиотеку.

Модули и библиотеки могут быть связаны с приложениями процессов для использования в Process Center. Они могут использоваться как службы процессами, созданными в Process Designer. В таких случаях они также развертываются с приложением процесса.

Кроме того, модули и библиотеки могут развертываться прямо в тестовой среде или на Process Server. Модули посредников могут создавать потоки передачи, которые развертываются на WebSphere Enterprise Service Bus или на Process Server.

IBM Integration Designer также поддерживает создание типов данных и карт XML, которые можно развернуть на устройстве WebSphere DataPower. С устройством WebSphere DataPower можно обмениваться файлами.

Инструменты администрирования

IBM Business Process Manager содержит инструменты администрирования для выполнения таких задач, как, например, установка моментальных копий и управление ими и администрирование процессов и работа с ресурсами в среде ИТ.

Инструменты командной строки

IBM Business Process Manager содержит инструменты командной строки, интерфейсы сценариев и программирования для администрирования среды выполнения.

- Инструменты командной строки - это простые программы, запускаемые из командной строки операционной системы для выполнения определенных задач. Эти инструменты позволяют запускать и останавливать серверы приложений, проверять состояние сервера, добавлять или удалять узлы и выполнять прочие задачи.
- Утилита wsadmin работает на основе сценариев и представляет собой интерпретатор команд с текстовым интерфейсом для выполнения административных задач и сценариев (Jython или Jacl). Она позволяет выполнять те же задачи, что и административная консоль, а также многие задачи консоли Process Center. Утилита wsadmin полезна в рабочих системах и автоматизированных операциях.
- Административные интерфейсы программирования - это набор классов Java и методов, соответствующих спецификации Java Management Extensions (JMX) и обеспечивающих поддержку функций администрирования архитектуры компонентов служб (SCA) и бизнес-объектов. Каждый интерфейс программирования содержи описание своего назначения, пример использования интерфейса или класса, а также ссылки на описание отдельных методов.

Консоль Process Center

Консоль Process Center предоставляет удобное для пользователей расположение для создания и управления высокоуровневыми элементами библиотеки, такими как приложения процессов и инструменты. Она предоставляет рабочую среду, в которой разработчики и аналитики BPM могут компоновать процессы и основные реализации. Кроме того, консоль Process Center содержит инструменты для обслуживания хранилища, в том числе для настройки прав доступа пользователей и групп.

Обратитесь к консоли Process Center с помощью веб-браузера (пример: **http://host:9080/ProcessCenter**).

Консоль Process Admin

Консоль Process Admin предназначена для администрирования серверов процессов в среде, в том числе для управления пользователями и установленными моментальными копиями на серверах. Она также содержит инструменты для управления очередями и кэшем.

Консоль Process Admin содержит компонент Process Inspector, позволяющий просматривать экземпляры процессов из приложений процессов, выполняющихся на определенных серверах процессов.

Обратитесь к консоли Process Admin с помощью веб-браузера (например, **http://host:9080/ProcessAdmin**).

Консоль Business Performance Admin

Для управления хранилищами данных о производительности в среде используется консоль Business Performance Admin. Этот инструмент предназначен для управления очередями сервера и контроля за производительностью сервера.

Обратитесь к консоли Business Performance Admin с помощью веб-браузера (например, **http://host:9080/PerformanceAdmin**).

Административная консоль WebSphere Application Server

Административная консоль применяется для администрирования приложений, служб и прочих ресурсов на уровнях ячейки, узла, сервера или кластера. Консоль можно использовать с автономными серверами и администраторами развертывания, которые управляют всеми серверами в сетевой среде.

Если установлен автономный профайл, то работает одиночный узел в собственном административном домене, называемом ячейкой. Административная консоль применяется для управления приложениями, шинами, серверами и ресурсами в этом административном домене. Если же настроена ячейка среды сетевого развертывания, то есть узел администратора развертывания и один или несколько управляемых узлов в одной ячейке. Административная консоль применяется для управления приложениями, настройки управляемых узлов в ячейке и контроля за узлами и их ресурсами.

Обратитесь к этой консоли с помощью веб-браузера (например, **http://host:9060/ibm/console** или **https://host:9043/ibm/console**).

Business Process Choreographer Explorer и Business Process Archive Explorer

В зависимости от роли пользователя можно использовать эти пользовательские интерфейсы для управления процессами BPEL и неавтоматизированными задачами, созданными в IBM Integration Designer, работы с присвоенных задач, просмотра завершенных процессов BPEL и неавтоматизированных задач, архивированных в базе данных, или удаления процессов и задач из архива.

Виджеты администрирования

Виджеты Администрирование позволяют администрировать и отслеживать отдельные компоненты, решений по управлению бизнес-процессами, включая модули и службы Расширенной службы интеграции. Используйте эти виджеты в бизнес-пространстве, для того чтобы обеспечить прозрачность модулей и приложения службы, что поможет ответить на такие вопросы как:

- Какими службами модуль пользуется и какие предоставляет? Каково время ответа и производительность за определенный период времени для этих служб?
- Каково состояние модуля?
- Присутствуют ли в модуле события с ошибками?
- Какие стратегии передачи связаны с модулем?
- Какие процессы BPEL и неавтоматизированные задачи используются в модуле?
- Существуют ли бизнес-календари или бизнес-правила в модуле?

Используйте один или несколько виджетов для получения моментальной копии общей работоспособности системы бизнес-решения, включая И топологии (среды развертывания, кластеры), приложения системы (например, состояние необработанных событий или Business Process Choreographer), источники данных, службы обмена сообщениями и очереди сообщений.

Business Process Rules Manager

Администратор бизнес-правил - это веб-инструмент, с помощью которого бизнес-аналитик может просматривать и изменять бизнес-правила. Это компонент IBM Process Server, который можно выбрать для установки во время создания профайла или после установки сервера.

Специальные возможности в IBM Business Process Manager

Специальные возможности позволяют работать с продуктом пользователям с ограниченными возможностями, в том числе с ограниченной подвижностью или ослабленным зрением.

IBM стремится сделать свои продукты доступными каждому, независимо от возраста или физических возможностей.

Дополнительная информация о специальных возможностях этого продукта приведена в разделе Специальные возможности в IBM Business Process Manager.

Поддержка языков в IBM Business Process Manager

IBM Business Process Manager предлагается на разных языках. В списках описан уровень поддержки для разных языков.

IBM Business Process Manager поддерживает следующие языки. Документация может быть переведена не полностью.

- Упрощенный китайский
- Традиционный китайский
- Чешский
- Английский (США)
- Французский
- Немецкий
- Венгерский
- Итальянский
- Японский
- Корейский
- Польский
- Португальский (Бразилия)
- Русский
- Испанский

IBM Business Process Manager предлагает частичную поддержку следующих языков. Документация может быть переведена не полностью.

- Арабский (переведены виджеты неавтоматизированных задач BPEL, виджеты Business Process Choreographer Explorer, среда Business Space и виджеты монитора Business Space)
- Датский (переведены виджеты монитора Business Space и среда Business Space)
- Голландский (переведены Process Designer, Process Center, BPD Modeler, Service Modeler, JSEditor, Process Designer, среда Business Space и виджеты монитора Business Space)
- Финский (переведены виджеты монитора Business Space, среда Business Space, BPD Modeler, Service Modeler, JSEditor и Process Designer)
- Греческий (переведены Process Designer, Process Center и Business Space)
- Иврит (переведены неавтоматизированные задачи BPEL, Business Process Choreographer Explorer, среда Business Space и виджеты монитора Business Space)
- Норвежский (переведены виджеты монитора Business Space и среда Business Space)
- Португальский (Португалия) (Process Designer, Process Center, BPD Modeler, Service Modeler и JSEditor)
- Румынский (переведены операции среды выполнения)
- Словацкий (переведены Business Space, BPD Modeler, Service Modeler, JSEditor и Process Designer)
- Шведский (переведены виджеты монитора Business Space и среда Business Space)
- Турецкий (переведен Business Space)

  Примечание: Для турецкой локали необходимо записи **case-insensitive-security-cache** в файле `60Database.xml` присвоить значение **false**, чтобы имена пользователей и пароли могли содержать букву *i*.

Файл 60Database.xml расположен в каталоге *корневой-каталог-установки*\profiles*имя-профайла*\config\cells*имя-ячейки*\nodes*имя-узла*\servers*имя-сервера*\process-center\config\system\.

IBM Business Process Manager позволяет пользователям вводить двунаправленные строки в среде Process Designer, Coach и Process Portal. Доступны API JavaScript для работы с двунаправленными языками.

Coach и Process Portal поддерживают арабский и еврейский календари.

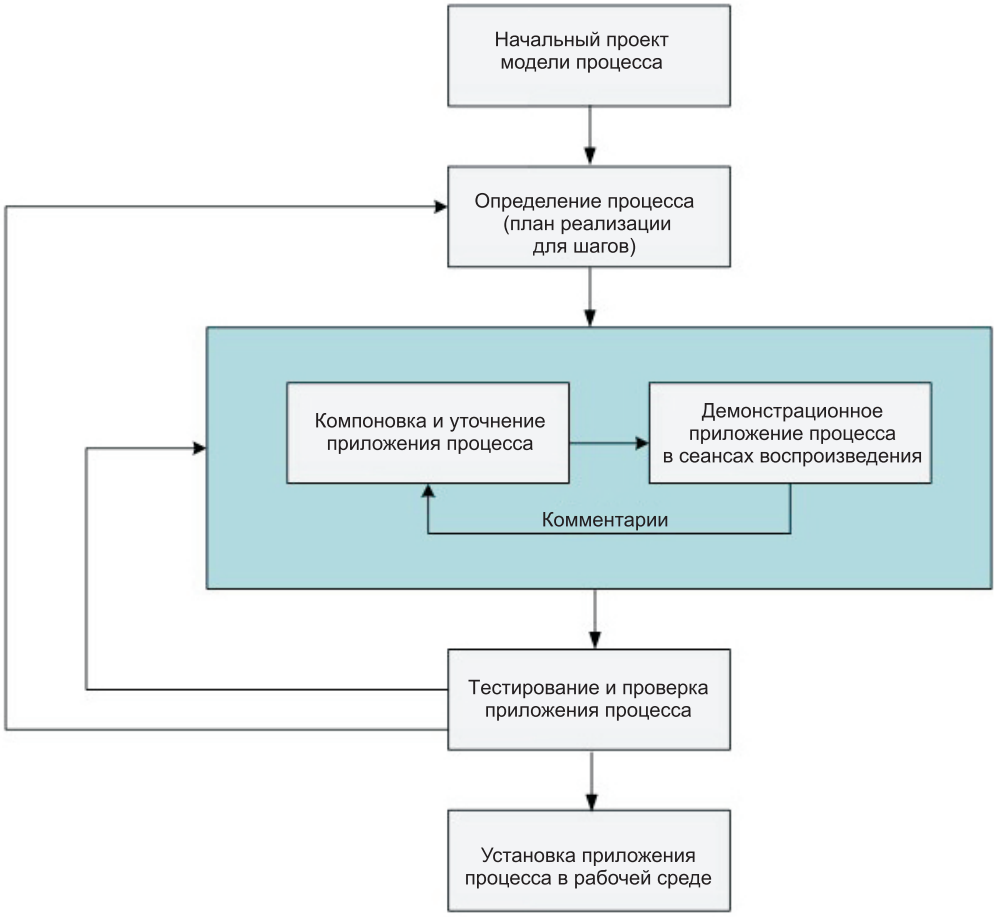
Обзор управления бизнес-процессами

Во время разработки процессов в Process Designer необходимо составить план окончательной установки приложений процессов на серверы в рабочих средах и средах тестирования.

Process Designer находится в IBM Business Process Manager Express, IBM Business Process Manager Standard и IBM Business Process Manager Advanced. В этом разделе мы сфокусируемся на версии Advanced, предназначенной для крупномасштабной автоматизации и использования сложных служб, разработанных в Integration Designer. Она имеет встроенные компоненты SOA, которые могут быть использованы для всесторонней интеграции служб в масштабах предприятия. Версию Standard могут совместно использовать многие бизнес-специалисты для разработки разнообразных сложных процессов. Она имеет базовую системную интеграцию. Версия Express предназначена для небольшого числа пользователей на одиночном сервере, которые начинают работу с бизнес-процессами или не нуждаются в доступе ко многим внешним системам.

На следующей диаграмме показан жизненный цикл типичного процесса разработки. Сюда относятся шаги по компоновке и уточнению службы установки, упрощающие установку приложений процессов в рабочей среде.

Как показано на диаграмме, можно работать исключительно в среде разработки. Однако требуется настроить Process Server и для тестовой, и для рабочей среды.



Обзор моделирования процессов

Процесс - это главный логический модуль в IBM Business Process Manager. Он является контейнером для всех компонентов определения процесса, включая службы, действия и шлюзы; хронометр, сообщение и события исключительных ситуаций; линии последовательности, правила и переменные. При моделировании процесса создается многоразовое Определение бизнес-процесса (BPD).

Компоненты процесса позволяют определить поток операций процесса для конечных пользователей, создавая логику внутри него и интегрируя с другими приложениями и источниками данных. Для понимания того, что происходит внутри процесса во время выполнения, важно понимать компоненты, составляющие процесс во время проектирования.

Компоновка процессов в IBM BPM

В разработку процессов с помощью IBM BPM обычно вовлекаются многие различные лица из разных организаций. Основной задачей является обеспечение создания наилучшего решения, которое выполняет поставленные цели проекта. Для достижения успешных результатов сотрудники коллектива должны работать совместно, чтобы выполнить требования к процессу и последовательно разработать модель вместе с ее реализациями.

Повторное использование элементов в Process Designer

Process Designer позволяет разработчикам процессов повторно использовать существующие элементы текущего приложения или других приложений процессов. Например, если известно, что уже существуют

несколько служб, которые включают Coach и другие совместно используемые элементы, которые требуются вам и другим разработчикам, то можно получить доступ и повторно использовать эти элементы, включив их в набор инструментов. Затем из приложения процесса можно добавить зависимость для этого набора инструментов, в котором находятся эти общие элементы. Службы станут доступны во время выбора реализации операции. Элементы инструментария могут также использоваться разработчиками, работающими над другими приложениями процессов.

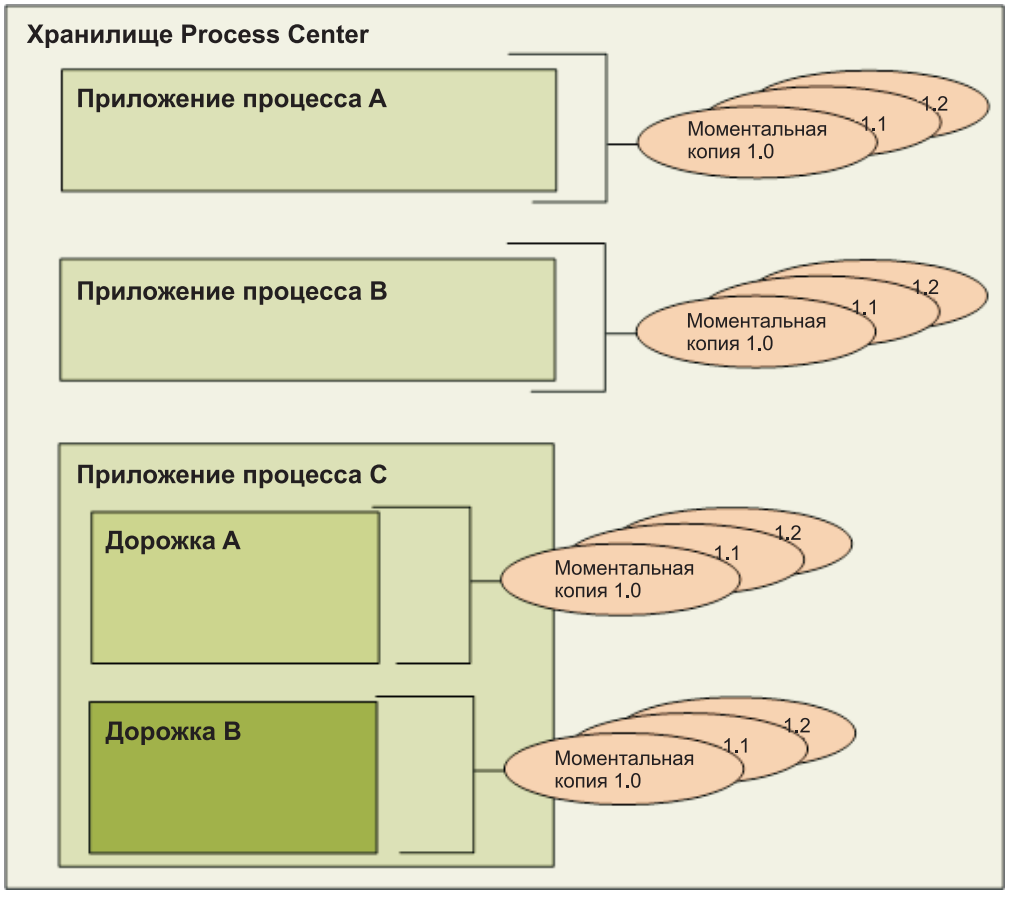
Работа с Проектировщиком с IBM Process Designer

Интерфейс Проектировщик предоставляет инструменты, нужные для моделирования процессов в IBM BPM.

Разработка процессов в Process Center

IBM Process Center служит центральным хранилищем для всех ресурсов проектов, создаваемых в Process Designer. Когда к Process Center подключено несколько клиентов Process Designer пользователи могут совместно использовать элементы (процессы, службы и пр.) и видеть изменения, вносимые другими пользователями. Process Center также можно использовать в качестве хранилища ресурсов, создаваемых в IBM Integration Designer.

При разработке процессов в Process Designer существует доступная в хранилище Process Center иерархия, разработанная для помощи в управлении проектами. На следующем рисунке показана структура иерархии хранилища:



Как показано на предыдущей диаграмме, хранилище Process Center содержит следующие артефакты:

Тип содержимого	Описание
Приложения процессов	Контейнеры для моделей процессов и поддерживающих реализаций, которые аналитики и разработчики BPM создают на панели Designer в IBM Process Designer.
Дорожки	Необязательные единицы деления приложения процесса на задачи коллектива или версии приложения. Если включены, дорожки позволяют вести параллельную разработку. Администраторы решают, требуются ли дополнительные дорожки, и включают их для каждого приложения процесса индивидуально.
Моментальные копии	Моментальные копии записывают состояние элементов приложения процесса или дорожки на определенный момент времени. Обычно моментальные копии представляют вехи или используются для воспроизведения или установки. Моментальные копии можно сравнивать и возвращать прежние моментальные копии. Когда администратор включает дорожки для приложения процесса, он использует моментальную копию в качестве основы для создания новой дорожки.

Приложения процессов: обзор

Приложение процесса представляет собой контейнер для моделей процессов и их реализаций; оно расположено в хранилище. После создания артефактов они собираются в приложение процесса.

Приложения процессов могут содержать следующие артефакты:

- Модели процессов; другое название - определения бизнес-процессов (BPD)
- Ссылки на комплекты инструментов
- Службы для реализации операций или интеграции с другими системами, в том числе Расширенные службы интеграции
- Одна или несколько дорожек
- Модули и библиотеки SCA (созданные в IBM Integration Designer)
- Другие элементы, необходимые для выполнения процесса

Вступительный видеоролик о приложении итерационного процесса и разработке Toolkit с последними вариантами, объектами Snapshot и дорожками (“Разработка приложения процесса и разработка Toolkit”), можно посмотреть на YouTube или в справочной системеIBM Education Assistant Information Center. Доступна стенограмма видеоролика.

Концевая точка приложения процесса, моментальные копии и дорожки

Изменения, внесенные в приложение процесса, динамически сохраняются в хранилище Process Center в последнем варианте (в текущей рабочей версии приложения процесса). Используя сеансы воспроизведения для варианта, можно мгновенно протестировать текущую рабочую версию приложения процесса.

Приложение процесса остается на этом уровне варианта, пока вы не решите создать моментальную копию, сохраняющую состояние элементов библиотеки в приложении процесса или дорожке на определенный момент времени. Обычно, моментальная копия создается перед каждым тестированием интеграции или установкой приложения процесса на сервере Process Center или сервере процессов для тестирования, промежуточной обработки или работы.

Примечание: Концевая точка – это специальная моментальная копия и единственный вид моментальной копии, в которой можно изменять содержимое, но выполнять ее можно только на сервере Process Center. Концевую точку не обязательно устанавливать на сервере процессов.

По умолчанию каждое приложение процесса имеет одну дорожку, называемую `Главная`. Если вы хотите разрешить параллельную разработку приложения процесса, можно создать дополнительные дорожки. Эти необязательные подразделения приложения процесса сохраняют изменения изолированными. Например, представьте, что ваша компания находится в процессе ребрендинга. Во время этого перехода должны поддерживаться текущие приложения процессов, и при этом новые версии разрабатываются на основании

измененного корпоративного названия. В этой ситуации один коллектив может вносить второстепенные исправления в текущую версию приложения процесса (в дорожке Главная), в то время как другой коллектив создает новую версию приложения процесса в отдельной дорожке.

Комплекты инструментов для приложений процессов

Комплекты инструментов — это контейнеры, хранящие элементы библиотеки (например, BPD) для повторного использования приложениями процессов или другими комплектами инструментов. Приложения процессов могут совместно использовать элементы библиотеки из одного или нескольких комплектов инструментов, и комплекты инструментов могут совместно использовать элементы библиотеки из других комплектов инструментов. При наличии прав доступа к объекту Toolkit можно создать зависимость от него и использовать элементы библиотеки этого Toolkit в приложении процесса.

Приложения процессов и приложения бизнес-уровня

У каждого приложения процесса есть приложение бизнес-уровня (BLA), действующее как контейнер для приложения процесса и его ресурсов (в число которых входят модули SCA, комплекты инструментов и библиотеки). Каждая моментальная копия приложения процесса имеет свое собственное BLA. Многие задачи администрирования для моментальной копии (например остановка и запуск на рабочем сервере) выполняются на уровне BLA. Такой подход ускоряет и упрощает администрирование моментальной копии и всех ее ресурсов.

Итерационная разработка приложений процессов и Toolkit

Таблица 3. Введение

Сцена	Звук	Действия на экране
1	Посмотрите ознакомительный видеофильм об итерационной разработке приложений процессов и Toolkit в IBM Business Process Manager.	На начальном экране показано название видеофильма - *Итерационная разработка приложений процессов и Toolkit* - и подзаголовок: *Работа с последними вариантами, Snapshot и дорожками*. Copyright 2013, IBM Corporation.
2	В этом видео описаны приложения процессов и Toolkit и рассказано о том, как с помощью последних вариантов, Snapshot и дорожек управлять их разработкой и развертыванием.	Показан маркированный список тем, охваченных фильмом.
3	С течением времени меняются потребности организации и наблюдается эволюция процессов. При прохождении приложений процессов и Toolkit по этапам цикла разработки, тестирования и работы можно управлять ресурсами и версиями процесса с помощью IBM Business Process Manager.	Показана диаграмма цикла итерационного проектирования. По диаграмме перемещаются стрелки, показывающие циклических характер процесса проектирования.
4	В IBM Business Process Manager приложения процессов играют роль контейнеров для моделей процессов и поддерживающих реализаций, которые разработчики создают в IBM Process Designer.	Отображается вкладка Process App на консоли Process Center. Курсор наводится на имя приложения процесса, затем на ссылку, с помощью которой пользователь может открыть приложение в IBM Process Designer. Отображается экранный текст “Приложение процесса: устанавливаемый контейнер для модели процесса”.

Таблица 3. Введение (продолжение)

Сцена	Звук	Действия на экране
5	Toolkit - это контейнеры, содержащие те же самые артефакты, которые содержатся в приложениях процессов. Но, в отличие от приложений процессов, они не могут быть установлены и запущены на сервере процессов. В Toolkit хранятся артефакты, которые могут быть многократно использованы одним или несколькими приложениями процессов. В дальнейшем Toolkit косвенным образом устанавливаются с каждым приложением процесса, которое на них ссылается.	Отображается вкладка Toolkit на консоли Process Center. Курсор наводится на имя Toolkit, затем на ссылку, с помощью которой пользователь может открыть его в IBM Process Designer. Отображается экранный текст “Toolkit: контейнер для многоразовых артефактов”.

Таблица 4. Последние варианты, Snapshot и дорожки

Сцена	Звук	Действия на экране
6	IBM Business Process Manager поддерживает итерационный подход к разработке приложений процессов и Toolkit с использованием следующих объектов: • Последний вариант, представляющий собой текущую рабочую версию • Snapshot, которые записывают состояние элементов библиотеки в определенный момент • Дорожки, представляющие собой необязательные ветви в процессе приложения или в Toolkit, которые можно использовать для параллельной разработки нескольких версий	Отображается маркированный список комплектов, включенных в IBM BPM, которые поддерживают итерационный подход к разработке приложений процессов и Toolkit.
7	При разработке приложения процесса или Toolkit в IBM Process Designer все внесенные изменения сохраняются в хранилище Process Center. Эта текущая рабочая версия приложения процесса или Toolkit называется последний вариант.	Отображается диаграмма процесса в IBM Process Designer. Открывается компонент диаграммы и слово *данные* изменяется на слово *дата* в представлении Coach. Затем изменения сохраняются. Отображается экранный текст “Последний вариант: текущая рабочая версия”.
8	С помощью сеансов воспроизведения для последнего варианта можно мгновенно протестировать текущую рабочую версию приложения процесса или Toolkit. Последний вариант можно запустить только на сервере Process Center, невозможно установить последний вариант на Process Server.	Открывается начальная страница диаграммы процесса и начинается сеанс воспроизведения. При этом открывается интерфейс Inspector и отображается измененный Coach в окне браузера. Отображается экранный текст “Последний вариант: воспроизведение для мгновенного тестирования”.
9	После внесения всех изменений в приложение процесса или Toolkit можно создать его Snapshot. Snapshot записывает состояние всех элементов приложения процесса или Toolkit на определенный момент. После создания Snapshot процессы и связанные элементы остаются доступными для изменения в последнем варианте. В Process Designer доступные Snapshot показаны в разделе Хронология исправлений. Эти Snapshot доступны для просмотра, но изменить их невозможно.	Закрывается окно браузера и создается Snapshot приложения процесса. Отображается интерфейс Designer и курсор наводится на имя созданной Snapshot в разделе Хронология исправлений. Отображается экранный текст “Snapshot: записывает состояние всех элементов”.

Таблица 4. Последние варианты, Snapshot и дорожки (продолжение)

Сцена	Звук	Действия на экране
10	Можно сравнить Snapshot и просмотреть, когда они были созданы и какие процессы были в них добавлены.	Открывается панель сравнения Snapshot. Курсор наводится на дату создания Snapshot и на имя измененного элемента в процессе. Отображается экранный текст “Сравните, чем отличается одна Snapshot от другой”.
11	Когда создается новая Snapshot объекта Toolkit, приложения процессов, ссылающиеся на этот Toolkit, будут и далее использовать прежнюю Snapshot. Если открыть приложение процессов для изменения в IBM Process Designer, будет выдано предупреждение, сообщающее о доступной новой версии Toolkit.	Открывается консоль Process Center. Отображается страница Toolkit и открывается Toolkit. Создается Snapshot объекта Toolkit. Открывается страница Process App, и в Process Designer открывается приложение процесса. Курсор наводится на выданное предупреждение и связанные пункты меню. Отображается экранный текст “Приложения процессов продолжают использовать ссылки на прежнюю Snapshot объекта Toolkit”.
12	Когда приложение процесса или Toolkit готовы к развертыванию, администратор может установить одну или несколько их Snapshot на Process Server с помощью консоли Process Center. Установленные приложения процессов могут ссылаться только на Snapshot объекта Toolkit, а не на текущую версию Toolkit.	Открывается консоль Process Center, затем приложение процессов. Курсор наводится на ту ссылку на Snapshot, с помощью которой администратор может установить Snapshot. Отображается экранный текст “Установка Snapshot приложений процессов на сервер”.
13	С помощью консоли Process Center администраторы могут также управлять Snapshot, например создавать, архивировать и экспортировать Snapshot.	Открывается выпадающее меню рядом с именем Snapshot и курсор наводится на каждый пункт меню. Отображается экранный текст “Управление Snapshot с помощью консоли Process Center”.
14	По умолчанию каждое приложение процесса или Toolkit имеет одну дорожку - с именем Главная. Администраторы могут включить создание дополнительных дорожек, чтобы разрешить параллельную разработку. Эти необязательные подразделения позволяют изолировать изменения.	Открывается страница Управление приложения процессов. Включается переключатель “Разрешить пользователям создавать дорожки в этом приложении процесса”. Отображается экранный текст “Дорожки: необязательные подразделения, позволяющие изолировать изменения”.
15	Новая дорожка создается с помощью Snapshot. При этом выбранная Snapshot копируется на новую дорожку. Другие Snapshot из исходной дорожки на новую дорожку не копируются.	Открывается страница Snapshots приложения процессов. Создается дорожка с именем Дорожка-2. Созданная дорожка открывается, курсор наводится на имя Snapshot копирования. Отображается экранный текст “Создание дорожки из Snapshot”.
16	У каждой дорожки имеется четко различимый последний вариант, или текущая рабочая версия. В приложении процесса или Toolkit с несколькими дорожками можно скопировать ресурсы из Snapshot на одной дорожке в последний вариант другой дорожки. При этом исправленные ресурсы заменяют ресурсы, уже имеющиеся в целевой дорожке, а также добавляются новые ресурсы.	Открывается созданная Snapshot и отображаются параметры и элементы приложения процесса. Отображается определение бизнес-процесса. Ресурсы копируются в последний вариант дорожки Главная. Затем отображаются страницы, на которых показаны новые, обновленные и конфликтующие ресурсы. Отображается экранный текст “Копирование ресурсов на другие дорожки”.

Таблица 4. Последние варианты, Snapshot и дорожки (продолжение)

Сцена	Звук	Действия на экране
17	Здесь показана иерархическая структура Snapshot приложений процессов и дорожек в хранилище Process Center. На этой диаграмме приложения процессов A и B имеют одну дорожку по умолчанию, несколько Snapshot и одни последний вариант. У приложения процесса C имеется две дорожки (каждая с несколькими Snapshot), и окончательный вариант. Snapshot и окончательный вариант на каждой дорожке не зависят друг от друга.	Отображается диаграмма, показывающая взаимоотношения между окончательными вариантами, Snapshot и дорожками. На диаграмме показаны представления следующих приложений процессов: • Приложение процесса A, имеющее одну дорожку, четыре Snapshot и последний вариант • Приложение процесса B, имеющее одну дорожку, три Snapshot и последний вариант • Приложение процесса C, имеющее две дорожки. Одна дорожка содержит три Snapshot и последний вариант, а другая дорожка содержит две Snapshot и последний вариант.

Таблица 5. Сценарий использования последних вариантов, Snapshot и дорожек

Сцена	Звук	Действие на экране
18	Теперь, после знакомства с приложениями процессов и объектами Toolkit и работой с последними вариантами, объектами Snapshot и дорожками, рассмотрим ситуации, в которых их можно использовать. Проследим за действиями Роберта, бизнес-программиста компании ABC.	Показана фотография улыбающегося человека. Подпись к фотографии: “Роберт, бизнес-программист, компания ABC”.
19	Роберт разрабатывает процесс, моделирующий порядок приема новых сотрудников в компанию. Его процесс содержит дорожку по умолчанию - Главная. Роберт вносит изменения в процесс в последнем варианте. Для того чтобы убедиться, что процесс выполняется плавно, он начинает сеанс воспроизведения, запускающий последний вариант на сервере Process Center.	Отображается диаграмма, представляющая среду разработки приложения процесса. В разделе Разработка показаны три вложенных прямоугольника. Внешний прямоугольник представляет хранилище Process Center, средний - приложение процесса, а внутренний - дорожку Главная. Справа от раздела Разработка находится раздел Тестирование. Этот раздел содержит единственный прямоугольник, представляющий сервер Process Center. Под разделом Тестирование находится раздел Рабочая среда. Этот раздел содержит единственный прямоугольник, представляющий Process Server. Для того чтобы показать, что Роберт вносит изменения, на дорожке Главная отображается треугольник, представляющий последний вариант. Когда Роберт запускает сеанс воспроизведения, появляется стрелка, показывающая, что последний вариант копируется на сервер Process Center.

Таблица 5. Сценарий использования последних вариантов, Snapshot и дорожек (продолжение)

Сцена	Звук	Действие на экране
20	Роберт и его коллектив закончили разработку и тестирование приложения процесса, и теперь оно готово для развертывания в рабочей среде. Роберт создает Snapshot приложения процесса. Затем Элис (администратор) устанавливает Snapshot на Process Server. Теперь компания ABC может использовать процесс, созданный Робертом, как инструкции по приему на работу новых сотрудников.	Для того чтобы показать, что Роберт создает Snapshot, рядом с последним вариантом на дорожке Главная появляется овал, представляющий Snapshot. Для обозначения того, что Элис устанавливает Snapshot, появляется стрелка, показывающая, что Snapshot копируется на Process Server.
21	Через некоторое время компания ABC объявляет об изменении торговой марки и о своем будущем названии - компания DEF. Во переходный период необходимо будет обслуживать процесс найма, созданный Робертом, и одновременно разрабатывать новую версию, учитывающую обновленный бренд организации. Коллектив обслуживания вносит незначительные исправления в исходный процесс на той же дорожке Главная, в то время как Роберт и его коллектив создают новую версию приложения по найму сотрудников на дорожке Ребрендинг DEF.	Появляется дополнительная Snapshot, что демонстрирует дальнейшую разработку приложения процесса. В схему приложения процесса добавляется прямоугольник, представляющий дорожку Ребрендинг DEF. Сначала дорожка Ребрендинг DEF содержит одну Snapshot и последний вариант, затем добавляется еще одна Snapshot, что демонстрирует дальнейшую разработку дорожки.
22	Коллективы обслуживания и ребрендинга могут отдельно запускать свои сеансы воспроизведения на сервере Process Center. Когда приложения процессов будут готовы к развертыванию, каждая дорожка сможет установить одну или несколько Snapshot на Process Server.	Стрелки показывают направление последних вариантов из обеих дорожек, выполняющихся на сервере Process Center одновременно. Затем стрелки показывают Snapshot из каждой дорожки, выполняющейся на Process Server одновременно.

Таблица 6. Заключение

Сцена	Звук	Действие на экране
23	В этом видео были описаны приложения процессов и Toolkit и рассказано о том, как IBM Business Process Manager поддерживает итерационную разработку процессов с использованием последних вариантов, Snapshot и дорожек.	Показан маркированный список тем, охваченных фильмом.
24	Для получения дополнительной информации о IBM Business Process Manager и работе с последними вариантами ознакомьтесь с другими нашими видеороликами на YouTube или посетите следующие официальные ресурсы.	Показан маркированный список следующих дополнительных ресурсов: • Канал YouTube WebSphereEducation • IBM Business Process Manager V8.5 - Документация • IBM developerWorks • IBM Education Assistant

Запуск и отладка процессов

С помощью Инспектора отдельные разработчики могут запускать бизнес-процессы и службы на Process Center Server или на удаленном Process Server среды выполнения.

Инспектор в IBM Process Designer имеет ключевое значение при интерактивном подходе к разработке бизнес-процессов. Вся группа разработчиков может использовать Инспектор для демонстрации текущего проекта бизнес-процесса и реализации в сеансах воспроизведения. Сеансы воспроизведения позволяют собрать важную информацию от разных заинтересованных лиц бизнес-процесса, таких как руководители,

конечные пользователи и бизнес-аналитики. При интерактивном подходе к разработке бизнес-процесса обеспечивается соответствие приложений процессов задачам и потребностям всех участников.

Инспектор IBM Process Designer включает несколько инструментов, позволяющих выполнять задачи, аналогичные следующим, в каждой из настроенных сред:

Задача	**Описание**
Управление экземплярами бизнес-процессов	При запуске бизнес-процесса можно просмотреть все ранее выполненные и выполняющиеся в данный момент экземпляры на серверах IBM Business Process Manager в данной среде. Можно управлять запущенными экземплярами, например, останавливая и возобновляя их выполнение. Можно также управлять ранее выполненными экземплярами, фильтруя или удаляя определенные записи.
Пошаговое выполнение и отладка бизнес-процесса	Для выбранного экземпляра можно просмотреть текущий этап выполнения, а затем продолжить пошаговое выполнение процесса, проверяя каждый шаг выполнения. Иерархическое представление бизнес-процессов в сочетании с индикаторами, которые на диаграмме бизнес-процессов называются маркерами, позволяют легко определить текущее положение в бизнес-процессе. Кроме того, можно просмотреть переменные, используемые в каждом шаге, и их значения (где это применимо).

При работе в IBM Integration Designer Инспектор можно использовать, если проект связан с приложением процесса. Также существует несколько инструментов для отладки и тестирования. Дополнительная информация об инструментах Integration Designer приведена в разделах "Тестирование модулей" и "Использование отладчика интеграции для поиска неполадок" по связанным ссылкам.

Установка приложений процессов и управление ими

Жизненный цикл приложения процесса включает установку, администрирование и отмену развертывания моментальных копий. Поддержка версий также включаются в жизненный цикл.

При разработке процессов можно воспользоваться всеми преимуществами итерационного подхода, поддерживаемого инструментами Process Designer. Процессы развиваются во времени, начиная с этапа разработки, продолжая на этапе тестирования и заканчивая этапом эксплуатации. Даже в работе процессы могут продолжать развиваться соответственно меняющимся требованиям. Важно быть готовым к непрерывному жизненному циклу процессов, это позволит с самого начала проектировать эффективно.

На следующем рисунке показан итерационный подход к разработке процесса.

Обычная конфигурация Business Process Manager включает три среды для поддержки разработки и установки процессов.

Среда	Описание
Разработка	Создание и отработка приложений процессов в IBM Process Designer. На панели Проектировщик создаются модели процессов и реализуются шаги этих моделей. С помощью Инспектора демонстрируется ход разработки в сеансах воспроизведения; это позволяет быстро оценить и уточнить прототип. С помощью консоли Process Center установите свои приложения процессов на тестовом или рабочем сервере процессов.
Тестирование	С помощью консоли Process Center установите свои приложения процессов на Process Server в среде тестирования для выполнения формальных тестов контроля качества. Инспектор может применяться для проверки и устранения неполадок.
Рабочая среда	После устранения всех неполадок, выявленных в результате формального тестирования, с помощью консоли Process Center установите приложения процессов на Process Server в рабочей среде. Инспектор используется для изучения и устранения любых неполадок, возникающих в рабочей среде.

Для тестирования, установки и администрирования моментальной копии приложения процесса, содержащей материалы из IBM Business Process Manager Advanced, вашему ИД пользователя или вашей группе должны быть присвоены административные роли защиты Настройщик, Оператор *и* Ответственный за развертывание. Если вам не назначены указанные роли, настройте их, выбрав пункт **Пользователи и группы** на административной консоли WebSphere. См. раздел Роли защиты IBM Business Process Manager.

Стратегии выпуска и установки

Для обеспечения соответствия приложений процессов, которые реализуются и устанавливаются, стандартам качества вашей организации попробуйте определить стратегии выпуска и установки. После определения целей и требований к выпуску новых и обновляемых приложений процессов можно автоматизировать процессов, необходимые для утверждения и запуска программ.

Например, можно определить маршрут процесса к нескольким разным администраторам через различные структуры отчетов в вашей организации. Только после завершения сеанса работы каждого администратора с новым или обновленным процессом этот процесс можно установить в своей рабочей среде и предоставить конечным пользователям. Этапы такой проверки можно создать и реализовать в IBM Business Process Manager Advanced, чтобы обеспечить соблюдение всех рекомендаций организации и наличие всех необходимых подписей. Последним этапом проверки может быть уведомление, отправляемое коллективу ИТ, о том, что утвержденное приложение процессов готово к установке.

Создание, обеспечение доступа и подключение услуг

Бизнес-процессы часто используют службы, которые предоставляют для них необходимые функции. Эти службы показываются на диаграмме бизнес-процесса в виде операции или шага. Например, служба в Process Designer может вызывать внешнюю веб-службу или вызывать сложную и автоматизированную службу, спроектированную в Integration Designer.

Доступ к внешним службам, внешним по отношению к приложению

В этом сценарии рассматриваются различные способы доступа приложения к внешним службам и описываются высокоуровневые задачи, с этим связанные.

Примечание: Этот сценарий применим при работе с IBM Business Process Manager Advanced.

В интегрированном бизнес-приложении *бизнес-службы* взаимодействуют друг с другом для выполнения требуемой функции. Бизнес-служба выполняет повторяемую функцию или задачу, которая вносит свой вклад в достижение бизнес-цели. Но работа по поиску бизнес-служб и подключению к ним не относится к бизнес-функции. Отделение бизнес-функции от задачи управления подключениями к службам делает решение более гибким.

Взаимодействие со службой начинается, когда *клиент службы* отправляет запрос *поставщику службы* для выполнения бизнес-функции. Этот запрос передается в виде *сообщения*, в котором указано, какую функцию следует выполнить. Поставщик службы выполняет запрошенную функцию и отправляет сообщение с

результатом клиенту службы. Обычно для обеспечения обмена данными между службами и реализации других низкоуровневых функций ИТ, которые не зависят от бизнес-функций и бизнес-данных, требуется обработка сообщений. Например, маршрутизация, преобразование протоколов, преобразование, повтор сбойного вызова и динамический вызов службы. Эта обработка называется *передача*.

В состав IBM Integration Designer входит два типа модулей: модули (модули интеграции бизнес-процессов), предназначенные главным образом для хранения бизнес-логики (бизнес-процессов, бизнес-правил и конечных бизнес-автоматов и пр.), и модули передачи, реализующие потоки передачи. Несмотря на то что функции этих типов модулей пересекаются, рекомендуется изолировать бизнес-логику в бизнес-модулях, а логику передачи помещать в модули передачи.

Однако не всегда есть четкая грань между бизнес-логикой и логикой передачи. В этих случаях следует оценить объем *состояния*, то есть данных в переменных, которые потребуется обрабатывать между вызовами служб. Если обработка состояния не требуется или ее объем небольшой, рекомендуется использовать компонент потока передачи. Если необходимо сохранять состояние между вызовами служб или есть данные, которые требуют сохранения в переменных и обработки; рекомендуется использовать компонент бизнес-процесса. Например, при вызове нескольких служб и записи информации, возвращаемой от каждой из них, для последующей обработки после вызова всех служб следует использовать бизнес-процесс, в котором легко можно присвоить возвращаемые данные переменным. Другими словами, если объем состояния слишком велик, значит граница с бизнес-логикой уже пересечена. Следующие разделы помогают прояснить это утверждение.

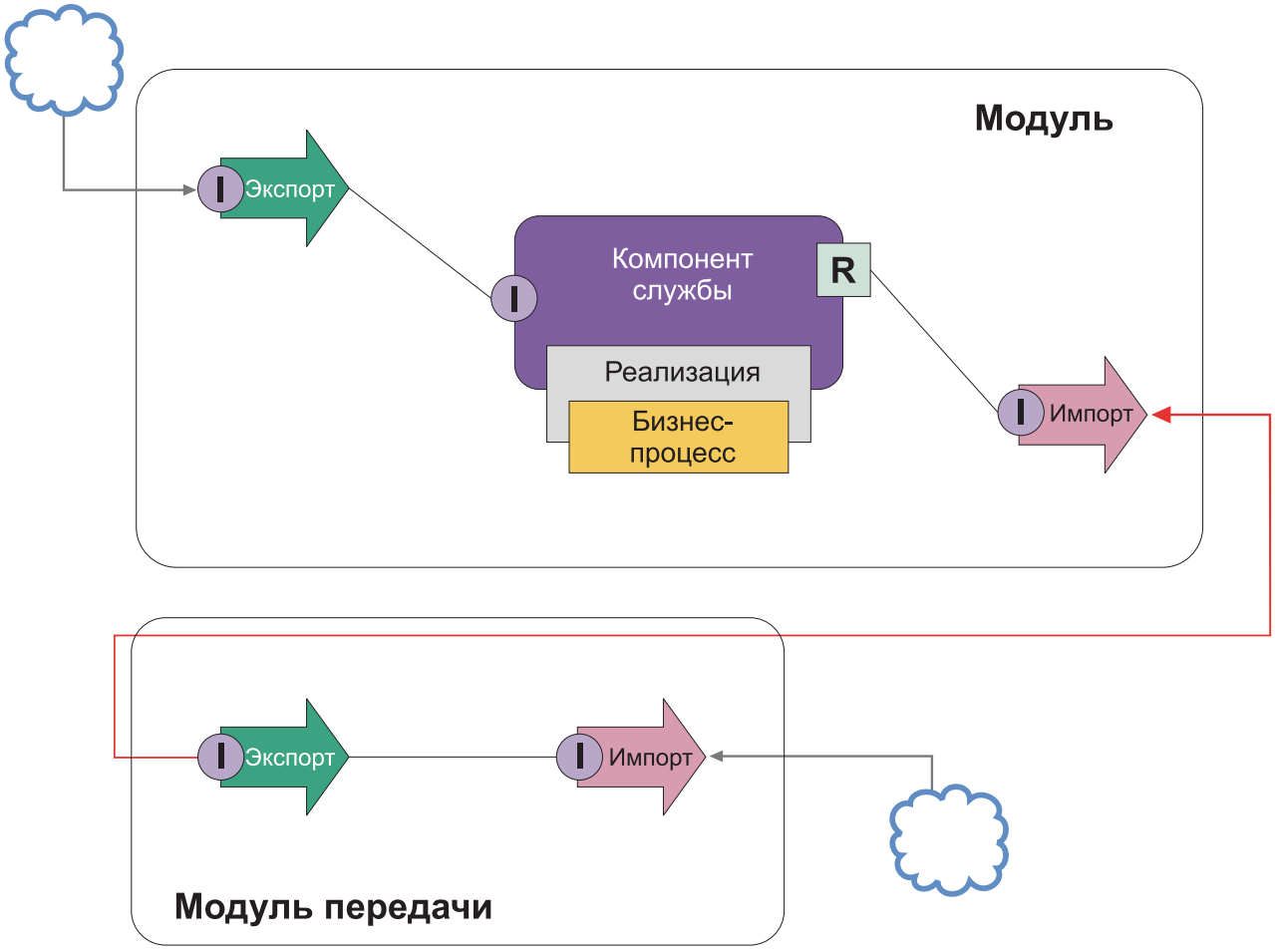
Не существует какого-то одного сценария интеграции, равно как не существует технически неверного ответа. В данных рекомендациях предлагается хорошо зарекомендовавшая себя методика, позволяющая добавить

гибкость и возможность многократного использования. Как всегда, следует тщательно взвесить преимущества и недостатки реализации этих шаблонов в приложении интеграции бизнес-процессов. Рассмотрим конкретные ситуации.

Доступ к компоненту SCA

Элементарный пример доступа к службе: объект импорта вызывает другой компонент SCA, преобразование данных не требуется. Даже в этой ситуации желательно обращаться к внешней службе не прямо из бизнес-модуля, а из модуля передачи. Это даст дополнительную гибкость в будущем, можно будет изменять конечные точки службы, QoS или управление (например, добавить ведение протокола), не затрагивая бизнес-компоненты, пользующиеся службой. Этот шаблон архитектуры называется "разделение ответственности".

Перед тем как выбрать этот шаблон, следует взвесить все его преимущества и последствия дополнительной нагрузки от другого модуля. Если гибкость является главным требованием и предполагается часто вносить изменения в службу, рекомендуется использовать отдельный модуль, как показано в данном документе. Если важнее производительность и предполагается обновлять и повторно развертывать бизнес-логику, рекомендуется использовать единый модуль.



Здесь перечислены высокоуровневые задачи для выполнения этого примера.

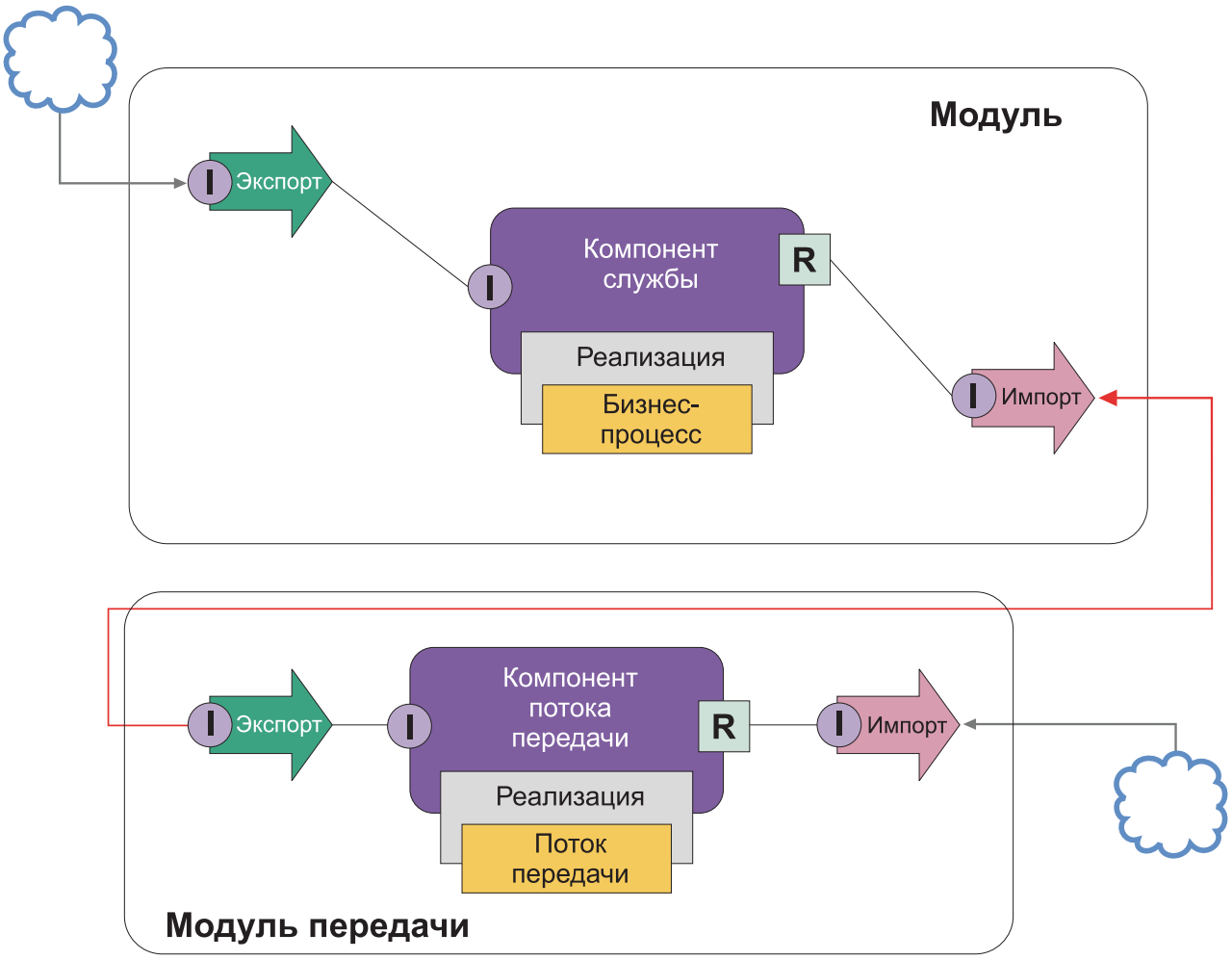
1. Создайте модуль передачи. Пошаговые инструкции приведены в разделе Создание модулей передачи.
2. В модуле передачи создайте объект импорта с соответствующей привязкой для внешней службы, к которой необходимо получить доступ. Пошаговые инструкции приведены в разделе Создание объектов импорта. Дополнительная информация о связываниях приведена в разделе Связывания.

3. Создайте экспорт с таким же интерфейсом, как у импорта. Пошаговые инструкции приведены в разделе Создание объектов экспорта.
4. Создайте привязку SCA для объекта экспорта. Пошаговые инструкции приведены в разделе Создание связываний SCA.
5. В сборке модуля передачи соедините импорта и экспорт проводниками. Сохраните модуль передачи.
6. Создайте модуль. Пошаговые инструкции приведены в разделе Создание модуля для бизнес-служб.
7. Добавьте экспорт и компонент.
8. На панели Бизнес-интеграция перенесите мышью объект экспорта, созданный в модуле передачи (шаг 4), в сборку модуля. Будет создан импорт с такой же привязкой, как у экспорта.
9. Соедините проводниками экспорт с компонентом, а компонент с импортом.
10. Добавьте реализацию компонента. Дополнительная информация о типах реализации приведена в разделе Реализации.

Позднее в модуль передачи можно будет добавить логику передачи (например, ведение протокола или маршрутизацию), не затрагивая бизнес-модуль.

Добавление передачи

Иногда простого вызова внешней службы недостаточно. В некоторых случаях требуется предварительная обработка с помощью модуля передачи, вставляемого между клиентом и поставщиком службы.



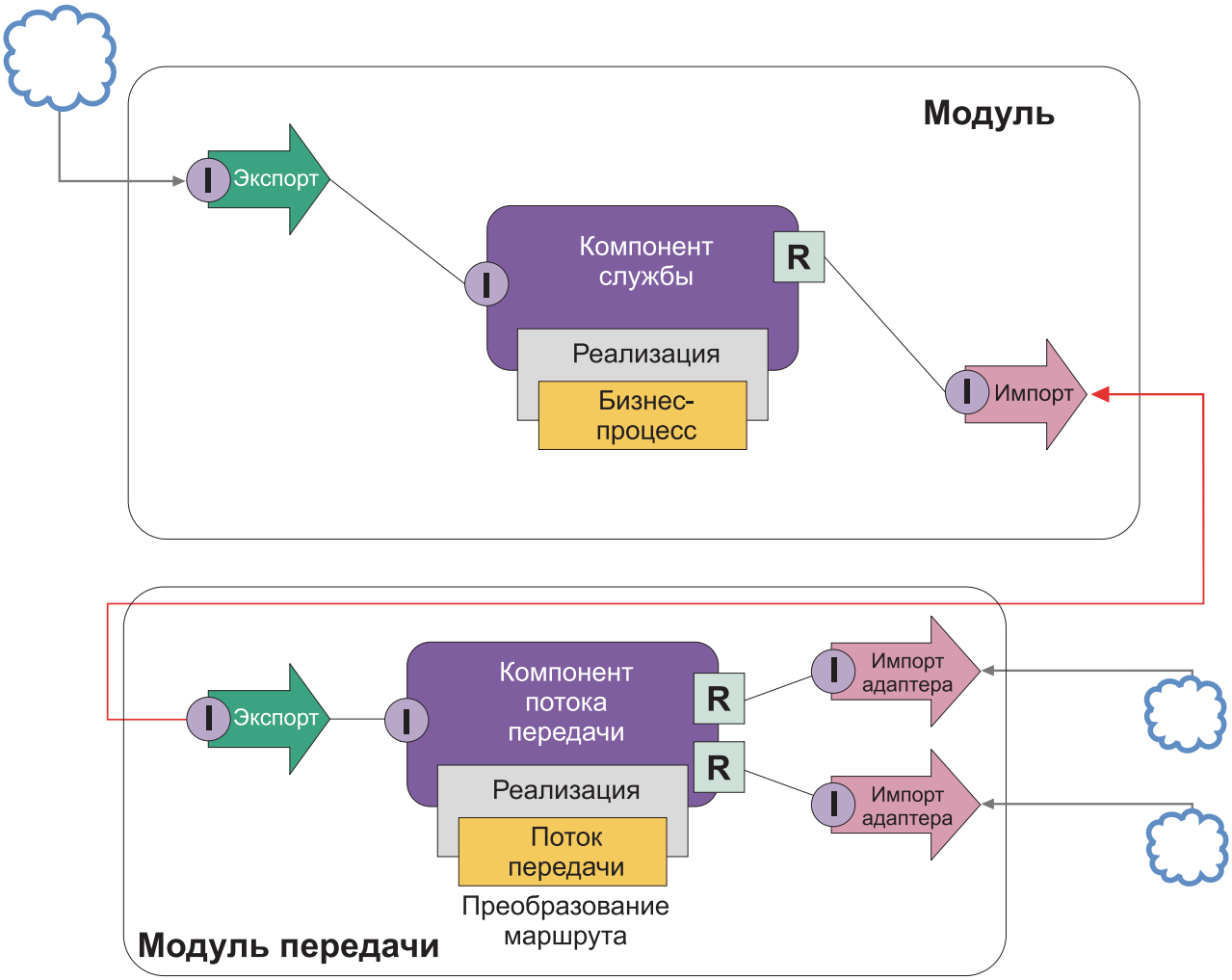
Ниже перечислены некоторые функции, которые может выполнять промежуточный поток передачи:

- Формирование заголовков протокола. За дополнительной информацией обратитесь к разделу Преобразование протокола в справочной системе WebSphere Enterprise Service Bus Information Center.
- Преобразование интерфейса или параметра с помощью примитива Карта бизнес-объектов или Преобразование. Преобразование сообщений
- Выбор службы из статического списка с помощью примитива Фильтр сообщений. Фильтр сообщений
- Вызов нескольких служб для объединения результатов с помощью примитивов Коэффициент разветвления по выходу и Коэффициент разветвления по входу. Объединение и рассылка сообщений
- Обработка сбоев вызова службы (повторный вызов, вызов другой службы) с помощью примитива Вызвать службу. Повторный вызов сбойной службы
- Динамическая маршрутизация (выбор службы во время выполнения, а не во время интеграции), позволяющая ослабить связь между службами и увеличить скорость реакции бизнеса на изменения. В среду выполнения можно добавлять новые службы, не затрагивая уже развернутые модули. Динамическая маршрутизация проявляет свои возможности наиболее полно, когда используется вместе с реестром, для которого требуется примитив передачи Поиск конечной точки. Динамический выбор конечных точек

Доступ к информационным системам предприятия

Службы и артефакты из внешних систем можно импортировать в Integration Designer. Мастер находит приложения и данные в информационных системах предприятия (EIS) и позволяет создать службы на основе найденных приложений и данных. Создаваемые артефакты представляют собой интерфейсы и бизнес-объекты, которые могут использоваться компонентами модуля.

Использование промежуточного модуля передачи между хостом и модулем делает этот модуль более пригодным для повторного использования. В следующем примере поток передачи используется для маршрутизации к правильному хосту и преобразования данных в требуемый хостом формат.



Здесь представлены высокоуровневые задачи для этого примера:

1. Подключитесь к хосту с помощью мастера внешней службы. Шаблон применения мастера внешней службы для доступа к внешним службам одинаковый для всех адаптеров. Дополнительная информация по использованию мастера внешней службы приведена в разделе Шаблон доступа к внешним службам с помощью адаптеров.
2. Создайте модуль. Пошаговые инструкции приведены в разделе Создание модуля для бизнес-служб.
3. Добавьте объект экспорта, компонент и объект импорта со связыванием SCA. Дополнительная информация приведена в разделе Вызов служб.
4. Добавьте к объекту экспорта интерфейс и соедините его проводниками с компонентом.
5. Добавьте реализацию компонента. В одном из свойств реализации укажите, к какой службе должен осуществляться доступ. Дополнительная информация о типах реализации приведена в разделе Реализации.
6. Создайте модуль передачи с объектом экспорта, имеющим связывание SCA и интерфейс, как у объекта импорта модуля, созданного на шаге 2.
7. Соедините проводниками объект экспорта и компонент потока передачи.
8. Для каждого хоста, к которому требуется доступ, создайте объект импорта с помощью соответствующего адаптера исходящих запросов из палитры редактора сборки.
9. Соедините проводниками компонент потока передачи с объектами импорта.

10. Реализуйте компонент потока передачи. Используйте примитив Фильтр сообщений для выбора объекта импорта по значению свойства, заданного в бизнес-логике, и примитив Mapping для каждого объекта импорта адаптера.
11. В модуле выберите объект экспорта модуля передачи в качестве службы, импортируемой в модуль. За пошаговыми инструкциями обратитесь к разделу Вызов службы из другого модуля.

Позднее можно будет внести изменения (например, добавить адаптер или перенастроить адаптер на другой хост) с минимальным воздействием на бизнес-логику.

Доступ к системам обмена сообщениями

Для того чтобы модуль SCA имел возможность обмениваться данными с существующим клиентом обмена сообщениями JMS, MQ или MQ JMS, для объектов импорта и экспорта необходимо создать интерфейсы, бизнес-объекты и связывания. См. раздел Преобразование сообщения в интерфейс SCA.

Для доступа к контексту и данным заголовков помимо бизнес-объектов потоки передачи используют сообщения. Если требуется доступ к данным заголовка JMS или к пользовательскому свойству JMS, следует использовать поток передачи. Если выполняется интеграция с системой MQ, и необходим доступ к информации заголовка MQ, то используйте поток передачи.

Создание или вызов веб-службы

Веб-службы представляют собой самостоятельные приложения, выполняющие бизнес-функции, начиная от простых запросов и заканчивая сложными взаимодействиями между бизнес-процессами. Можно вызвать существующую веб-службу или разработать новую веб-службу, отвечающую вашим требованиям. В этом сценарии приведены необходимые инструкции и даны ссылки на дополнительную информацию.

Некоторые, но возможно не все, службы потребуется создать с нуля с помощью IBM Integration Designer. Во время работы с редактором сборки и редактором бизнес-процесса для сборки служб в бизнес-процесс можно обнаружить, что некоторых служб не хватает. Поэтому может быть полезным создать недостающие службы с помощью инструментов IBM Integration Designer. Обратное также справедливо: после создания нового процесса может быть принято решение сделать все или часть операций процесса доступными извне в виде служб.

Примечание: Данный сценарий может применяться пользователями IBM Integration Designer для IBM Process Server.

Существует несколько причин для разработки веб-служб с помощью IBM Integration Designer:

- При создании службы в IBM Integration Designer можно реализовать службу в соответствии с бизнес-правилами.
- Разработанную в IBM Integration Designer службу Java™ можно экспортировать и как веб-службу, и как службу SCA.
- Преобразование интерфейсов без необходимости писать дополнительный код является еще одним преимуществом. Все преобразования данных можно изъять из кода Java, оставив для разработчика Java простую программу - черный ящик.
- В IBM Integration Designer все службы и взаимосвязи показаны в одном месте.
- Наличие функций рефакторинга также помогает в разработке веб-служб с помощью IBM Integration Designer.

Следует помнить, что веб-службы не могут быть решением всех проблем интеграции. Однако, как и в случае с любым другим технологическим или архитектурным подходом, применение веб-служб в правильном месте и в правильное время имеет свои преимущества.

Экспорты, импорты и привязки

IBM Integration Designer позволяет импортировать стандартные веб-службы и использовать эти службы в составных приложениях.

Для разработки служб в IBM Integration Designer предназначен редактор сборки. Модули, модули передачи, библиотеки и компоненты создаются стандартным способом. Затем к созданным службам можно организовать доступ с помощью экспортов, импортов и привязок. Ниже перечислены шаги по выполнению этих основных задач, ссылки указывают на дополнительную информацию о каждой задаче.

Для веб-служб можно использовать любую из двух привязок: привязка веб-службы или привязка HTTP. Привязка веб-службы содержит спецификацию для передачи сообщений в веб-службу и обратно. Инструменты помогают автоматически создать привязку веб-службы. Привязка HTTP представляет собой стандартный протокол запрос-ответ между клиентом и сервером по протоколу HTTP, опубликованному консорциумом W3C. Для требуется привязки HTTP требуется некоторая начальная настройка параметров конфигурации.

1. Создайте объект экспорта, чтобы опубликовать службу модуля для использования другими модулями.
2. Создайте привязку для экспорта.
   - Создайте привязку веб-службы для объекта экспорта.
   - Создайте привязку HTTP для объекта экспорта.
3. Создайте объект импорта для вызова существующей службы, не являющейся частью собираемого модуля.
   - Создайте привязку веб-службы для объекта импорта.
   - Создайте привязку HTTP для объекта импорта.

Если требуется вызывать веб-службу со страниц JSP, прочитайте соответствующий раздел.

Возможности разработки веб-служб

При открытии редактора, связанного с процессом создания веб-служб, может появиться следующее сообщение:

```
Для этого действия требуется компонент "Развертывание веб-служб".
Включить этот компонент?
```

IBM Integration Designer предоставляет функцию фильтрации, которая называется *возможностями*. В окне Параметры функции и инструменты распределены по категориям возможностей, эти категории, а также подмножества функций категорий, можно включать и выключать. См. раздел Возможности.

Дополнительные сведения об основных концепциях

Этот раздел может служить введением для знакомства с технологиями, применяемыми в IBM Business Process Manager.

Создание сценариев

Сценарии предназначены для работы с компонентами и продуктами из семейства средств, предназначенных для управления бизнес-процессами.

Поддержка версий

Жизненный цикл приложения процесса начинается с создания приложения процесса. Далее оно проходит этапы обновления, развертывания, сопряженного развертывания, отмены развертывания и архивирования. *Поддержка версий* - это способ управления жизненным циклом приложения процесса, в котором уникальным образом идентифицируются отдельные версии приложения процесса.

Механизм работы поддержки версий в IBM Business Process Manager зависит от того, что именно развертывается: приложение процесса, развертываемое из хранилища в IBM Process Center, или приложение J2EE, развертываемое непосредственно из IBM Integration Designer.

По умолчанию поддержка версий включена для приложений процессов и комплектов инструментов, развертываемых в среде выполнения из Process Center. Для приложений J2EE можно включить поддержку версией в IBM Integration Designer.

Кроме того, можно создавать версии неавтоматизированных задач или конечных автоматов, чтобы в среде выполнения могли сосуществовать несколько версий этих сущностей.

Поддержка версий приложений процессов

Поддержка версий обеспечивает идентификацию моментальный копий средой выполнения в жизненном цикле приложения процесса и возможность одновременного выполнения нескольких моментальных копий на сервере процессов.

Для того чтобы понять принципы поддержки версий приложений процессов, важно помнить, что приложение процесса является контейнером, который хранит различные артефакты, используемые в приложении процесса (например, модели процессов или BPD, ссылки на комплекты инструментов, службыили дорожки). Поддержка версий осуществляется на уровне этого контейнера, а не на уровне отдельных артефактов. Для приложений процессов это означает, что изменение версии происходит во время создания моментальной копии.

Можно сравнить моментальные копии и определить различия между версиями. Например, если один разработчик исправил неполадку со службой и сделал моментальную копию ее, содержащую приложение процесса или набор инструментов в этой точке, а затем другой разработчик внес несколько дополнительных изменений в ту же службу и сделал новую моментальную копию, то руководитель проекта может сравнить две моментальные копии и узнать, какие изменения были сделаны и кем. Если руководитель проекта решит, что дополнительные изменения в службе нецелесообразны, то у него будет возможность вернуть моментальную копию с первоначальным исправлением.

Можно выполнять различные версии (моментальные копии) приложения процесса параллельно на сервер. При установки новой моментальной копии или удалите первоначальную, или оставьте ее выполняющейся.

Контекст версии

Каждая моментальная копия содержит уникальные метаданные для идентификации версии (называемые контекстом версии). Можно присвоить этот идентификатор, но IBM рекомендует использовать трехразрядную цифровую систему обозначения версий в формате <основная>.<дополнительная>.<служебная>. Более подробное описание этой схемы обозначения версий содержится в разделах о соглашении об именах.

IBM Business Process Manager назначает глобальное пространство имен для каждого приложения процесса. Конкретно глобальным пространством имен является либо концевая точка приложения процесса, либо конкретная моментальная копия приложения процесса. Имя версии, используемое сервером, не должно быть длиннее семи символов, поэтому назначенным именем является акроним, состоящий из символов имени, присвоенного моментальной копии. Акронимы моментальных копий идентичны именам их моментальных копий, если имя моментальной копии соответствует рекомендованному стилю IBM VRM и не длиннее семи символов. Например, имя моментальной копии 1.0.0 имеет акроним 1.0.0, а имя моментальной копии 10.3.0 – акроним 10.3.0. Акроним моментальной копии должен быть уникальным в пределах контекста приложения процесса в области сервера Process Center. Поэтому нельзя изменять акроним моментальной копии.

Замечания о поддержке версий приложений процессов в нескольких кластерах

Одну и ту же версию приложения процесса можно развернуть в нескольких кластерах в пределах одной ячейки. Для различения этих нескольких развертываний одной версии приложения процессов создайте

моментальную копию для каждого развертывания и включите уникальный для ячейки ИД в имя моментальной копии, (например, v1.0_cell1_1 и v1.0_cell1_2). Каждая моментальная копия является новой версией приложения процесса (только с точки зрения управления жизненным циклом), но содержимое и функции одни и те же.

При развертывании приложения процесса в кластере осуществляется автоматическая синхронизация узлов.

Замечания о поддержке версий для комплектов инструментов Process Designer

Помните, что моментальные копии приложения процесса обычно создаются перед тестированием или установкой. Однако моментальные копии комплекта инструментов делаются, когда он готов к использованию приложениями процессов. Впоследствии если потребуется обновить набор инструментов, необходимо создать другую моментальную копию "концевой точки", после чего владельцы приложений процессов могут решить, переходить ли им на новую моментальную копию.

Поддержка версий для модулей и библиотек

Если модуль или библиотека входит в приложение процесса или комплект инструментов, то он следует соответствующему жизненному циклу (версии, моментальные копии, дорожки и т. д.). Имена модулей и библиотек должны быть уникальными в области действия приложения процесса или комплекта инструментов.

В этом разделе описана поддержка версий модулей и библиотек, используемых с приложениями процессов. Однако если модули развертываются на Process Server прямо из IBM Integration Designer, то можно присваивать номера версий модулям в ходе развертывания, как описано в разделе “Создание модулей и библиотек с поддержкой версий”.

Связанные с IBM Process Center модуль или библиотека должны иметь доступ к своим зависимым библиотекам в том же приложении процесса или в зависимом комплекте инструментов.

В следующей таблице перечислены варианты выбора, доступные в редакторе зависимостей в IBM Integration Designer, когда библиотека связана с приложением процесса или комплектом инструментов:

Таблица 7. Зависимости для модулей, приложений процессов или комплектов инструментов и глобальные библиотеки

Область библиотеки	**Описание**	**Может зависеть от . . .**
Модуль	Для каждого модуля, использующего библиотеку, на сервере существует копия библиотеки.	Библиотека с областью модуля может зависеть от всех типов библиотек.
Приложение процесса или комплект инструментов	Библиотека, совместно используемая всеми модулями в области приложения процесса или комплекта инструментов. Этот параметр имеет силу, если развертывание осуществляется посредством IBM Process Center. Если развертывание выполняется без IBM Process Center, то библиотека копируется в каждый модуль. **Примечание:** Для библиотек, созданных в IBM Integration Designer версии 8, по умолчанию применяется общий доступ на уровне **Приложение процесса или комплект инструментов**.	Библиотека этого типа может зависеть только от глобальных библиотек.

Таблица 7. Зависимости для модулей, приложений процессов или комплектов инструментов и глобальные библиотеки (продолжение)

Область библиотеки	Описание	Может зависеть от . . .
Глобальная	Библиотека, совместно используемая всеми запущенными модулями.	Глобальная библиотека может зависеть только от других глобальных библиотек. **Примечание:** Для развертывания глобальной библиотеки необходимо настроить общую библиотеку WebSphere. Дополнительная информация приведена в разделе “Модули и зависимости библиотек”.

Модули и библиотеки, связанные с инструментариями и приложениями процессов

Вам не обязательно поддерживать версии модулей и библиотек, связанных с инструментариями и приложениями процессов.

Модули и библиотеки, связанные с инструментарием или приложением процесса, не требуют поддержки версий. Фактически невозможно создать версию модуля или библиотеки, связанной с инструментарием или приложением процесса в редакторе зависимостей. Модули и библиотеки, связанные с инструментарием или приложением процесса, используют моментальные копии (функцию Process Center) для достижения того же результата, который обеспечивается поддержкой версий.

Библиотеки, связанные с инструментарием или приложением процесса, не имеют требуемого номера версии в разделе Библиотеки редактора Зависимостей, потому что версии не нужны.

Соглашения о присвоении имен

Соглашение о присвоении имен позволяет различить версии приложения процесса по мере того как оно проходит этапы обновления, развертывания, сопряженного развертывания, отмены развертывания и архивирования.

В этом разделе описан соглашения, применяемые для уникальной идентификации версий приложения процесса.

Контекст версии - это аббревиатура, уникальным образом описывающая приложение процесса или комплект инструментов. Для каждого типа аббревиатуры применяется соглашение об именах. Длина аббревиатуры не может превышать семь символов из набора [A-Z0-9_], за исключением аббревиатуры моментальной копии, которая также может содержать точку.

- Аббревиатура приложения процесса создается вместе с приложением процесса. Ее длина не должна превышать семь символов.
- Аббревиатура моментальной копии создается автоматически вместе моментальной копией. Ее длина не должна превышать семь символов.

  Если имя моментальной копии также является допустимой аббревиатурой моментальной копии, то аббревиатура и имя моментальной копии будут совпадать.

  Примечание: При использовании функций маршрутизации, поддерживающих компоненты потока передачи с учетом версии, присваивайте имя моментальной копии согласно схеме <*версия*>.<*выпуск*>.<*модификация*>, например, **1.0.0**. Поскольку аббревиатура моментальной копии ограничена семью символами, то можно использовать не более пяти цифр (плюс две точки). Поэтому увеличивать значение в полях следует с осторожностью, так как все символы после первых семи усекаются.

  Например, имя моментальной копии **11.22.33** усекается в аббревиатуре моментальной копии до **11.22.3**.
- Аббревиатура дорожки автоматически составляется из первых букв каждого слова в имени дорожки. Например, для новой дорожки с именем **My New Track** будет использоваться аббревиатура **MNT**.

Имя и аббревиатура по умолчанию для дорожки - **Main**. При развертывании на сервере IBM Process Center в контексте версии аббревиатура дорожки используется в том случае, если она отличается от **Main**.

Определение бизнес-процесса в приложении процесса обычно задается аббревиатурой имени приложения процесса, аббревиатурой моментальной копии и именем определения бизнес-процесса. При возможности используйте уникальные имена в определениях бизнес-процесса. Повторяющиеся имена могут привести в следующим ошибкам:

- Для экспорта определений бизнес-процесса в виде веб-служб может потребоваться посредник.
- Определение бизнес-процесса, созданное в IBM Process Designer, может быть недоступным в процессе BPEL, созданном в IBM Integration Designer.

Контекст версии может меняться в зависимости от способа развертывания приложения процесса.

Соглашения об именах для развертывания на сервере Process Center:

На IBM Process Center можно развернуть моментальную копию приложения процесса, а также моментальную копию комплекта инструментов. Кроме того, можно развернуть головную версию приложения процесса или комплекта инструментов. Термин *головная версия* означает текущую рабочую версию приложения процесса или комплекта инструментов. Контекст версии может меняться в зависимости от типа развертывания.

Для приложений процессов головная версия приложения процесса или какая-либо его моментальная копия позволяет уникальным образом идентифицировать версию.

Комплекты инструментов могут быть развернуты с несколькими приложениями процессов, но жизненный цикл комплекта инструментов связан с жизненным циклом приложения процесса. Каждое приложение процесса имеет собственную копию зависимых комплектов инструментов, развернутых на сервере. Совместно использование развернутого комплекта инструментов несколькими приложениями процессов невозможно.

Если с приложением процесса связана дорожка, имя которой отличается от применяемого по умолчанию (**Main**), то аббревиатура дорожки также составляет часть контекста версии.

Дополнительная информация приведена в разделе “Примеры” на стр. 34 ниже в этой статье.

Моментальные копии приложения процесса

Для развертывания моментальной копии приложения процесса контекст версии включает в себя следующие элементы:

- Аббревиатура приложения процесса
- Аббревиатура дорожки приложения процесса (для дорожек с именем, отличным от **Main**)
- Аббревиатура моментальной копии приложения процесса

Автономные комплекты инструментов

Для развертывания моментальной копии комплекта инструментов контекст версии включает в себя следующие элементы:

- Аббревиатура комплекта инструментов
- Аббревиатура дорожки комплекта инструментов (для дорожек с именем, отличным от **Main**)
- Аббревиатура моментальной копии инструментов

Головные версии

Головные версии приложений процессов используются в итерационном тестировании в Process Designer. Они могут быть развернуты только на серверах Process Center.

Для развертывания головной версии приложения процесса контекст версии включает в себя следующие элементы:

- Аббревиатура приложения процесса
- Аббревиатура дорожки приложения процесса (для дорожек с именем, отличным от **Main**)
- "Головная версия"

Головные версии комплектов инструментов также используются в итерационном тестировании в Process Designer. Они не могут быть развернуты на рабочем сервере.

Для развертывания моментальной копии головной версии контекст версии включает в себя следующие элементы:

- Аббревиатура комплекта инструментов
- Аббревиатура дорожки комплекта инструментов (для дорожек с именем, отличным от **Main**)
- "Головная версия"

Примеры

Ресурсы должны иметь уникальные имена и определяться во внешней области с помощью контекста версии.

- В следующей таблице показаны примеры уникальных имен. В этом примере головная версия приложения процесса использует имя дорожки по умолчанию (**Main**):

Таблица 8. Головная версия приложения процесса с именем дорожки по умолчанию

Тип имени	**Пример**
Имя приложения процесса	**Process Application 1**
Аббревиатура приложения процесса	**PA1**
Дорожка приложения процесса	**Main**
Аббревиатура дорожки приложения процесса	"" (если дорожка - **Main**)
Моментальная копия приложения процесса	
Аббревиатура моментальной копии приложения процесса	**Tip**

  Все модули SCA, связанные с этой головной версией приложения процесса, включают контекст версии, как показано в следующей таблице:

Таблица 9. Модули SCA и файлы EAR с учетом версий

Название модуля SCA	**Имя с учетом версии**	**Имя EAR/приложения с учетом версии**
M1	PA1-Tip-M1	PA1-Tip-M1.ear
M2	PA1-Tip-M2	PA1-Tip-M2.ear

- В следующей таблице показан пример головной версии приложения процесса, где имя дорожки отличается от значения по умолчанию:

Таблица 10. Головная версия приложения процесса с именем дорожки, отличающимся от принятого по умолчанию

Тип имени	**Пример**
Имя приложения процесса	**Process Application 1**

Таблица 10. Головная версия приложения процесса с именем дорожки, отличающимся от принятого по умолчанию (продолжение)

Тип имени	Пример
Аббревиатура приложения процесса	**PA1**
Дорожка приложения процесса	**Track1**
Аббревиатура дорожки приложения процесса	**T1**
Моментальная копия приложения процесса	
Аббревиатура моментальной копии приложения процесса	**Tip**

Все модули SCA, связанные с этой головной версией приложения процесса, включают контекст версии, как показано в следующей таблице:

Таблица 11. Модули SCA и файлы EAR с учетом версий

Название модуля SCA	Имя с учетом версии	Имя EAR/приложения с учетом версии
M1	PA1-T1-Tip-M1	PA1-T1-Tip-M1.ear
M2	PA1-T1-Tip-M2	PA1-T1-Tip-M2.ear

Аналогичные соглашения об именах применяются к расширенным головным версиям комплектов и развертываниям моментальных копий. Они применимы также к дополнительным моментальным копиям, установленным в Process Server.

- В следующей таблице показаны примеры уникальных имен. В этом примере моментальная копия приложения процесса использует имя дорожки по умолчанию (**Main**):

Таблица 12. Моментальная копия приложения процесса с именем дорожки по умолчанию

Тип имени	Пример
Имя приложения процесса	**Process Application 1**
Аббревиатура приложения процесса	**PA1**
Дорожка приложения процесса	**Main**
Аббревиатура дорожки приложения процесса	"" (если дорожка - **Main**)
Моментальная копия приложения процесса	**Process Shapshot V1**
Аббревиатура моментальной копии приложения процесса	**PSV1**

Все модули SCA, связанные с этой моментальной копией приложения процесса, включают контекст версии, как показано в следующей таблице:

Таблица 13. Модули SCA и файлы EAR с учетом версий

Название модуля SCA	Имя с учетом версии	Имя EAR/приложения с учетом версии
M1	PA1-PSV1-M1	PA1-PSV1-M1.ear
M2	PA1-PSV1-M2	PA1-PSV1-M2.ear

- В следующей таблице показан пример моментальной копии приложения процесса, где имя дорожки отличается от значения по умолчанию:

Таблица 14. Моментальная копия приложения процесса с именем дорожки, отличающимся от принятого по умолчанию

Тип имени	Пример
Имя приложения процесса	**Process Application 1**

Таблица 14. Моментальная копия приложения процесса с именем дорожки, отличающимся от принятого по умолчанию (продолжение)

Тип имени	**Пример**
Аббревиатура приложения процесса	**PA1**
Дорожка приложения процесса	**Track1**
Аббревиатура дорожки приложения процесса	**T1**
Моментальная копия приложения процесса	**Process Snapshot V1**
Аббревиатура моментальной копии приложения процесса	**PSV1**

Все модули SCA, связанные с этой моментальной копией приложения процесса, включают контекст версии, как показано в следующей таблице:

Таблица 15. Модули SCA и файлы EAR с учетом версий

Название модуля SCA	**Имя с учетом версии**	**Имя EAR/приложения с учетом версии**
M1	PA1-T1-PSV1-M1	PA1-T1-PSV1-M1.ear
M2	PA1-T1-PSV1-M2	PA1-T1-PSV1-M2.ear

Соглашения об именах для развертывания на Process Server:

На Process Server можно развернуть моментальную копию приложения процесса. Аббревиатура моментальной копии приложения процесса уникальным образом идентифицирует версию.

Для развертывания моментальной копии приложения процесса контекст версии включает в себя следующие элементы:

- Аббревиатура приложения процесса
- Аббревиатура моментальной копии приложения процесса

Ресурсы должны иметь уникальные имена и определяться во внешней области с помощью контекста версии. В следующей таблице показаны примеры уникальных имен:

Таблица 16. Пример имен и аббревиатур

Тип имени	**Пример**
Имя приложения процесса	**Process Application 1**
Аббревиатура приложения процесса	**PA1**
Моментальная копия приложения процесса	**1.0.0**
Аббревиатура моментальной копии приложения процесса	**1.0.0**

Контекст версии является часть идентификации ресурса, такого как модуль или библиотека.

В следующей таблице показаны примеры контекста версии для двух модулей и связанных файлов EAR:

Таблица 17. Модули SCA и файлы EAR с учетом версий

Название модуля SCA	**Имя с учетом версии**	**Имя EAR/приложения с учетом версии**
M1	PA1-1.0.0-M1	PA1-1.0.0-M1.ear
M2	PA1-1.0.0-M2	PA1-1.0.0-M2.ear

В следующей таблице показаны примеры контекста версии для двух библиотек в области приложения процесса и связанных файлов JAR:

Таблица 18. Библиотеки в области приложения процесса и файлы JAR с учетом версии

Имя библиотеки SCA с областью действия приложения процесса	Имя с учетом версии	Имя JAR с учетом версии
Lib1	PA1-1.0.0-Lib1	PA1-1.0.0-Lib1.jar
Lib2	PA1-1.0.0-Lib2	PA1-1.0.0-Lib2.jar

Связывание с учетом версий

Приложения процессов могут содержать модули SCA, включающие связывание импорта и экспорта. При сопряженном развертывании приложений связывание для каждой версии приложения должно быть уникальным. В некоторых случаях это достигается при развертывании приложений, поскольку часть описаний связывания обновляется автоматически. В других случаях уникальность связывания требуется обеспечить вручную, обновив его после развертывания.

Связывание *с учетом версии* применяется для заданной версии приложения процесса. Это гарантирует уникальность связывания для приложений процессов. В следующих разделах перечислены описания связывания, которые обновляются автоматически для обеспечения учета версий, а также указаны действия, которые необходимы в среде выполнения для описаний связывания, не учитывающих версии. Дополнительная информация об операциях при создании модулей приведена в разделе “Замечания по использованию связывания”.

SCA

Если связывание импорта и экспорта определено в одной и той же области приложения процесса, то целевой объект связывания SCA переименовывается автоматически с учетом версий во время развертывания.

Если эти связывания не определены в одной и той же области приложения процесса, то в протокол заносится сообщение. Для изменения целевого адреса конечной точки необходимо изменить связывание импорта после развертывания. Это можно сделать из административной консоли.

Веб-служба (JAX-WS или JAX-RPC)

Целевой адрес конечной точки связывания веб-службы автоматически переименовывается с учетом версий, если выполнены все следующие условия:

- В адресе используется соглашение об именах по умолчанию:
  http://*IP-адрес*:*порт*/*имя-модуля*Web/sca/*имя-экспорта*
- Адрес конечной точки имеет формат SOAP/HTTP.
- Связывание импорта и экспорта определено в одной и той же области приложения процесса.

Если эти условия не выполнены, в протокол заносится сообщение. Дальнейшие действия зависят от способа развертывания приложения процесса:

- В случае сопряженного развертывания приложения процесса необходимо вручную переименовать URL конечной точки SOAP/HTTP или URL целевой очереди SOAP/JMS, чтобы он был уникальным для разных версий приложения процесса. Это можно сделать из административной консоли.
- Если развертывается только одна версия приложения процесса, то это сообщение можно проигнорировать

Для сопряженного развертывания моментальной копии связывания веб-службы SOAP/JMS дальнейшие действия зависят от способа развертывания приложения процесса:

- Если импорт и целевой объект экспорта относятся к одному и тому же приложению процесса, то перед публикацией приложения процесса в Process Center и созданием моментальной копии выполните следующее:

1. Измените URL конечной точки экспорта. Целевой объект и фабрика соединений должны быть уникальными.
2. Измените URL конечной точки импорта, чтобы он совпадал с указанным для экспорта на предыдущем шаге.

- Если импорт и целевой объект экспорта относятся к разным приложениям процессов, то выполните следующее:
  1. Измените URL конечной точки экспорта. Целевой объект и фабрика соединений должны быть уникальными.
  2. Опубликуйте приложение процесса в Process Center.
  3. Создайте моментальную копию.
  4. Разверните приложение процесса на Process Server.
  5. В административной консоли WebSphere измените URL соответствующей конечной точки импорта, чтобы он совпадал с указанным для экспорта.

HTTP

URL конечной точки связывания HTTP автоматически переименовывается с учетом версий, если выполнены все следующие условия:

- В адресе используется соглашение об именах по умолчанию:

  http(s)://*IP-адрес*:*порт*/*имя-модуля*Web/*путь-контекста-в-экспорте*
- Связывание импорта и экспорта определено в одной и той же области приложения процесса.

Если эти условия не выполнены, в протокол заносится сообщение. Дальнейшие действия зависят от способа развертывания приложения процесса:

- В случае сопряженного развертывания приложения процесса необходимо вручную переименовать URL конечной точки, чтобы он был уникальным для разных версий приложения процесса. Это можно сделать из административной консоли.
- Если развертывается только одна версия приложения процесса, то это сообщение можно проигнорировать

JMS и базовый JMS

Системные и базовые связывания JMS автоматически учитывают версии.

Примечание: Для пользовательских и базовых связываний JMS во время развертывания автоматическое переименование не применяется, чтобы связывание могло учитывать версии. Если связывание определено пользователем, то необходимо переименовать следующие атрибуты для обеспечения уникальности различных версий приложения процесса:

- Конфигурация конечной точки
- Целевая очередь приема
- Имя порта получателя запросов (если он определен)

При изменении конечной точки модуля настройте соответствующий целевой адрес отправки.

MQ/JMS и MQ

Для связываний MQ/JMS или MQ во время развертывания автоматическое переименование не применяется, чтобы связывание могло учитывать версии.

Необходимо переименовать следующие атрибуты для обеспечения уникальности различных версий приложения процесса:

- Конфигурация конечной точки
- Целевая очередь приема

При изменении конечной точки модуля настройте соответствующий целевой адрес отправки.

EJB

Для связывания EJB во время развертывания автоматическое переименование не применяется, чтобы связывание могло учитывать версии.

Необходимо переименовать атрибуты имен JNDI для обеспечения уникальности различных версий приложения процесса.

Обратите внимание, что эти новые имена JNDI должны также обновляться в клиентских приложениях.

EIS

Если имя по умолчанию адаптера ресурса (***имя-модуля*приложение:описание-адаптера**) не было изменено во время развертывания, то он переименовывается автоматически с учетом версий.

Если имя по умолчанию адаптера ресурса было изменено, то необходимо использовать уникальные имена адаптеров ресурсов для различных версий приложения.

Если имена адаптеров ресурса не являются уникальными, то во время развертывания в протокол заносится предупреждение. После развертывания можно вручную изменить имена адаптеров ресурсов из административной консоли.

Динамический вызов с учетом версии

В компонентах потока передачи можно настроить динамическое определение маршрутов сообщений, отправляемых в конечные точки. При создании модуля посредника можно настроить поиск маршрута к конечной точке с учетом версии.

Если для моментальной копии используется стиль IBM_VRM (<*версия*>.<*выпуск*>.<*модификация*), то можно экспортировать файл EAR приложения процесса в WebSphere Service Registry and Repository (WSRR). Тогда при создании модуля посредника можно настроить поиск маршрута к конечной точке с учетом версии. Например, в поле **Стратегия соответствия** можно указать **Найти конечную точку с последней совместимой версией служб SCA на основе модулей** и в поле **Тип связывания** выбрать **SCA**.

В этом случае будущие версии приложения процесса, развернутые на сервере и опубликованные в WSRR, будут использовать последние совместимые версии службы конечной точки, которые динамически будет находить модуль посредника.

Обратите внимание, что вместо этого можно настроить целевой объект в SMOHeader, и это значение будет передаваться в сообщении-запросе.

Развертывание приложений процессов с модулями и проектами Java

Приложения процессов могут содержать пользовательские модули Java EE и проекты Java. При сопряженном развертывании приложений пользовательский модуль Java EE для каждой версии приложения должен быть уникальным.

Обратите внимание, что пользовательские модули Java EE и проекты Java развертываются на сервере, если они развернуты с модулем SCA, в котором объявлена зависимость от них. Если параметр **Развернуть с модулем** (значение по умолчанию) не выбран при объявлении зависимости, то необходимо развернуть модуль или проект вручную.

Развертывание приложений процессов с бизнес-правилами и селекторами

При развертывании нескольких версий приложения процесса с бизнес-правилами и селекторами необходимо учитывать способ использования соответствующих мета-данных версиями.

Динамические мета-данные для бизнес-правила или компонента селектора определяются во время выполнения на основе имени компонента, целевого пространства имен компонента и типа компонента. Если в одной среде выполнения развернуто две или более версии приложения процесса, содержащего бизнес-правило или селектор, то эти версии будут совместно использовать одну и ту же логику правила (бизнес-правило) или мета-данные маршрутизации (селектор).

Для того чтобы все версии приложения процесса, содержащего бизнес-правило или селектор, применяли собственные динамические мета-данные (логика правила или маршрутизация), сделайте целевое пространство имен уникальным для каждой версии приложения процесса.

Объекты конфигурации

С помощью команд инструмента администрирования из командной строки WebSphere (wsadmin) AdminConfig можно обращаться к свойствам базы данных и к свойствам защиты в IBM Business Process Manager и изменять их.

Термин *объект конфигурации* обозначает объект, к которому можно обращаться с помощью команд wsadmin AdminConfig. Дополнительная информация приведена в разделе Свойства конфигурации защиты.

Архитектура развертывания

Архитектура развертывания IBM Business Process Manager состоит из программных процессов, именуемых серверами, топологических элементов, именуемых узлами и ячейками, а также хранилища конфигурации, применяемого для хранения информации о конфигурации.

Ячейки

В IBM Business Process Manager *ячейками* называются логические группы, включающие один или несколько узлов в распределенной сети.

Ячейка представляет собой концепцию конфигурации, т.е. способ, который используют администраторы для логической связи узлов друг с другом. Администраторы определяют узлы ячейки в соответствии с особыми критериями, имеющими смысл в среде организации.

Административные данные конфигурации хранятся в файлах XML. Ячейка хранит основные файлы конфигурации каждого сервера из каждого узла ячейки. У каждого узла и сервера есть собственные логические файлы конфигурации. Изменения, вносимые в логические файлы конфигурации сервера или логического узла, носят временный характер, если сервер входит в ячейку. Пока логические изменения действуют, они переопределяют конфигурацию из ячейки. Изменения основных файлов конфигурации узла и сервера, внесенные на уровне ячейки, заменяют любые временные изменения на узле после синхронизации документов конфигурации из ячейки с узлами. Синхронизация выполняется при возникновении определенных событий, например при запуске сервера.

Серверы

Серверы обеспечивают основные функции IBM Business Process Manager. Серверы процессов расширяют или дополняют функции сервера приложений, связанные с обработкой модулей архитектуры компонентов служб (SCA). Другие серверы (администраторы развертывания и агенты узлов) используются для управления серверами процессов.

Сервер процессов может быть *автономным* или *управляемым*. Управляемый сервер может быть элементом *кластера*. Набор управляемых серверов, кластеров серверов и другого промежуточного программного обеспечения называется *средой развертывания*. В среде развертывания каждый управляемый сервер или кластер настроен для выполнения определенной функции (например, целевой хост, хост модуля приложения или сервер инфраструктуры событий общего формата). Автономный сервер настроен для выполнения всех необходимых функций.

Серверы обеспечивают среду выполнения для модулей SCA, для ресурсов, используемых этими модулями (источниками данных, спецификациями активации и целевыми объектами JMS), а также для ресурсов IBM (адресаты сообщений , контейнеры Business Process Choreographer, и серверы инфраструктуры событий общего формата).

Агент узла - административный агент, представляющий узел системы и управляющий серверами данного узла. Агенты узлов выполняют мониторинг серверов на хостах и направляют административные запросы на серверы. Агент узла создается при объединении узла с администратором развертывания.

Администратор развертывания - административный агент, предоставляющий централизованную панель управления для нескольких серверов и кластеров.

Автономный сервер определяется автономным профайлом, администратор развертывания - профайлом администратора развертывания, управляемые серверы создаются на *управляемом узле*, определяемом профайлом управляемого узла.

Автономные серверы:

Автономный сервер предоставляет среду для развертывания модулей SCA в одному процессе сервера. К этому процессу сервера относятся административная консоль, целевая среда развертывания, поддержка обмена сообщениями, администратор правил бизнес-процессов и сервер инфраструктуры событий общего формата.

Автономный сервер отличается простотой установки и имеет консоль Быстрое начало работы, из которой можно запускать и останавливать сервер, открывать галерею примеров и административную консоль. При открытии галереи примеров после установки примеров IBM Business Process Manager в автономном сервере развертывается пример системы. Ресурсы, используемые в данном примере, можно найти в административной консоли.

На автономном сервере можно развернуть собственное решение, однако он не имеет производительности, масштабируемости или устойчивости, необходимой для рабочей среды. В качестве рабочей среды рекомендуется использовать среду сетевого развертывания.

Можно начать с автономного сервера, а потом включить его в среду сетевого развертывания, объединив его с ячейкой администратора развертывания, *если другие узлы не были объединены с этой ячейкой*. Невозможно объединить несколько автономных серверов в одной ячейке. Для того чтобы добавить автономный сервер, используйте административную консоль администратора развертывания или команду **addNode**. Если для объединения используется команда **addNode**, автономный сервер не должен работать.

Автономный сервер определяется профайлом автономного сервера.

Кластеры:

Кластеры - группы совместно управляемых серверов, участвующих в управлении рабочей нагрузкой.

Кластер может включать узлы или отдельные серверы приложений. Как правило, узел - это физическая компьютерная система со своим IP-адресом, на которой работает один или несколько серверов приложений. Кластеры можно объединять в конфигурацию ячейки, которая логически связывает многие серверы и кластеры, имеющие разные конфигурации и приложения, учитывая стоящие перед администратором задачи и соображения целесообразности для организации.

Кластеры применяются для распределения нагрузки между серверами. Серверы, входящие в состав кластера, называются элементами кластера. При установке приложения в кластере оно устанавливается на всех элементах кластера.

Поскольку каждый элемент кластера содержит одни и те же приложения, задачи клиентов можно распределять по системам в зависимости от имеющихся у них ресурсов, присвоив им весовые коэффициенты.

Присваивание весовых коэффициентов серверам в кластере позволяет повысить производительность и упростить аварийное восстановление. Задачи назначаются тем серверам, у которых есть ресурсы для их выполнения. Если один из серверов недоступен для выполнения задачи, то она назначается другому элементу кластера. Среда с возможностью переназначения задач имеет очевидные преимущества перед использованием одного сервера приложений, который может быть перегружен при наличии слишком большого числа запросов.

Профайлы

Профайл определяет уникальную рабочую среду с отдельными файлами команд, файлами конфигурации и файлами протокола. Профайлы определяют три разных типа сред в системах IBM Business Process Manager: автономный сервер, администратор развертывания и управляемый узел.

С помощью профайлов можно включить в систему несколько рабочих сред без необходимости установки нескольких копий исполняемых файлов IBM Business Process Manager.

С помощью Утилита командной строки BPMConfig можно создать профайлы IBM BPM. В качестве альтернативных способов создания профайлов администратора развертывания или управляемого узла можно использовать утилиту командной строки **manageprofiles** или ее графический интерфейс Инструмент управления профайлами (PMT). PMT больше не поддерживается при создании автономного профайла.

Примечание: В распределенных платформах каждый профайл имеет уникальное имя. В z/OS все профайлы имеют имя “default”; переименовывать, изменять, копировать или удалять профайлы в z/OS нельзя.

Типы профайлов

В IBM Business Process Manager V8.5 доступны следующие типы профайлов IBM BPM:

Автономный профайл IBM BPM
Автономный профайл определяет автономные серверы с функциями и возможностями для конкретных конфигураций IBM BPM Express. Для создания автономного профайла требуется шаблон профайла `BPM/BpmServer`, который поддерживается только в IBM BPM Express.

Профайл администратора развертывания IBM BPM
Профайл администратора развертывания определяет администратор развертывания, предоставляющий один административный интерфейс для логической группы серверов на одной или нескольких рабочих станциях. Для создания профайла администратора развертывания требуется шаблон профайла `BPM/BpmDmgr`, который поддерживается только в IBM BPM Standard и IBM BPM Advanced.

Профайл управляемого узла IBM BPM
Профайл управляемого узла определяет управляемый узел, если узел объединен с администратором развертывания. Для создания профайла управляемого узла требуется шаблон профайла `BPM/BpmNode`, который поддерживается только в IBM BPM Standard и IBM BPM Advanced.

Указанные типы профайлов доступны для всех конфигураций IBM BPM.

Таблица 19. Доступные типы профайлов IBM BPM

	Типы профайлов		
Конфигурация IBM BPM	**Автономные**	**Администратор развертывания**	**Управляемый узел**
IBM BPMExpress	Да	Нет	Нет
IBM BPM Standard	Нет	Да	Да
IBM BPMРасширенный	Нет	Да	Да

Таблица 19. Доступные типы профайлов IBM BPM (продолжение)

	Типы профайлов		
Конфигурация IBM BPM	**Автономные**	**Администратор развертывания**	**Управляемый узел**
Среда полнофункционального тестирования (UTE) для IBM Integration Designer	Да	Необязательный	Необязательный

Каталог профайла

Каждый профайл в системе находится в собственном каталоге, который содержит все файлы. Задайте каталог расположения профайла при его создании. По умолчанию это каталог `profiles` в каталоге установки IBM Business Process Manager. Например, профайл Dmgr находится в каталоге `C:\Program Files\IBM\WebSphere\ProcServer\profiles\Dmgr`.

Стандартный профайл

Первый профайл, созданный в одном экземпляре продукта IBM Business Process Manager, считается *стандартным*. Стандартный профайл является стандартным целевым объектом для команд, запущенных из каталога `bin` в каталоге установки IBM Business Process Manager. Если в системе существует всего один профайл, к нему обращается каждая команда. Если создается другой профайл, его можно сделать стандартным.

Примечание: Стандартный профайл необязательно имеет имя “default”.

Дополнение профайлов

Если профайл администратора развертывания, профайл управляемого узла или профайл автономного сервера уже создан для WebSphere Application Server Network Deployment, его можно *дополнить* для поддержки IBM Business Process Manager в дополнение к существующей функции. Для того чтобы расширить профайл, сначала необходимо установить IBM Business Process Manager. Затем используйте утилиту командной строки **`manageprofiles`** для дополнения автономного профайла либо воспользуйтесь инструментом управления профайлами или утилитой командной строки **`manageprofiles`** для дополнения профайла администратора развертывания или управляемого узла.

Важное замечание: В среде сетевого развертывания необходимо сначала дополнить профайл администратора развертывания, а затем дополнить профайлы управляемого узла.

Ограничение: Невозможно дополнить автономный профайл или профайл администратора развертывания, в котором был изменен реестр пользователей WebSphere VMM по умолчанию, например для использования LDAP.

Администраторы развертывания

Администратор развертывания представляет собой сервер, управляющий операциями для логических групп, ячеек или других серверов. Администратор развертывания является центральным элементом администрирования серверов и кластеров.

При создании среды развертывания первым создается профайл администратора развертывания. Каждая создаваемая среда развертывания имеет консоль Быстрое начало работы, с помощью которой можно запускать или останавливать администратор развертывания и запустить его административную консоль. Административная консоль администратора развертывания применяется для управления серверами и кластерами в ячейке. В частности, она применяется для настройки серверов и кластеров, добавления серверов в кластеры запуска и остановки серверов и кластеров, а также развертывания модулей SCA.

Несмотря на то, что администратор развертывания представляет собой один из типов серверов, модули в самом администраторе развертывания развертывать нельзя.

Узлы

Узел - это логическая группа управляемых серверов.

Узел, как правило, соответствует логической или физической компьютерной системе с отдельным IP-адресом хоста. Узлы не могут охватывать несколько компьютеров. Имена узлов идентичны имени хоста компьютера.

Узлы в топологии сетевого развертывания могут быть как управляемыми, так и неуправляемыми. Агент управляемого узла управляет его конфигурацией и серверами. Неуправляемые узлы агента не имеют.

Управляемые узлы:

Управляемый узел - это узел, объединенный с администратором развертывания, который содержит агент узла и может включать управляемые серверы. В управляемом узле можно настраивать и запускать управляемые серверы.

Серверы, настроенные на управляемом узле, составляют ресурсы среды развертывания. Эти серверы создаются, настраиваются, запускаются, останавливаются, управляются и удаляются с помощью административной консоли администратора развертывания.

Агент управляемого узла управляет всеми серверами этого узла.

Если узел объединяется, процесс агента узла создается автоматически. Этот агент узла должен работать для управления конфигурацией профайла. Например, при выполнении следующих задач:

- Запуск и остановка процессов сервера.
- Синхронизация данных конфигурации в администраторе развертывания с копией на узле.

Однако агент узла не обязательно должен работать для того, чтобы приложения могли запускать или настраивать ресурсы узла.

Управляемый узел может содержать один или несколько серверов, которые управляются администратором развертывания. Решения можно развертывать на серверах управляемого узла, но он не содержит галереи примеров приложений. Управляемый узел определяется профайлом управляемого узла и имеет консоль Быстрое начало работы.

Неуправляемые узлы:

У неуправляемого узла нет агента узла, управляющего его серверами.

У неуправляемых узлов в топологии сетевого развертывания могут быть определения серверов, например веб-серверов, но не определения серверов приложений. Неуправляемые узлы нельзя встроить в ячейку. Это означает, что на неуправляемый узел нельзя добавить агент узла. Особым типом неуправляемого узла является автономный сервер. Администратор развертывания не поддерживает управление автономными серверами, так как информация о них отсутствует в ячейке. Автономный сервер можно встроить в ячейку. В результате этого автоматически создается агент узла. Узел становится управляемым узлом ячейки.

Агенты узлов

Агенты узла представляют собой административных агентов, которые передают административные запросы на серверы.

Агент узла - это сервер, который работает в компьютерной системе каждого хоста, включенного в конфигурацию сетевого развертывания. Этот агент применяется только для администрирования и не

участвует в работе функций обслуживания приложений. На агенте узла выполняется ряд важных административных функций, в том числе службы передачи файлов, синхронизации конфигурации и мониторинга производительности.

Особенности присвоения имен профайлам, узлам, серверам, хостам и ячейкам

В этом подразделе описываются зарезервированные термины и неполадки, связанные с именами профайлов, узлов, серверов, хостов и ячеек (в соответствующих случаях). Информация из этого раздела относится к распределенным платформам.

Особенности имен профайлов

В качестве имени профайла можно использовать любое уникальное имя со следующими ограничениями. Не используйте следующие символы в именах профайлов:

- Пробелы
- Специальные символы, недопустимые в имени каталога для данной ОС, такие как *, & или ?.
- Символы косой черты (/) и обратной косой черты (\)

Двухбайтовые символы допустимы.

Windows **Особенности указания пути к каталогу:** длина пути к каталогу установки не должна превышать 60 символов. Число символов в имени каталога *profiles_directory_path\profile_name* не должно превышать 80 символов.

Примечание: При создании профайла в среде Windows рекомендуется использовать соглашение о кратких именах, поскольку в системах Windows длина пути не должна превышать 255 символов.

Особенности присвоения имен для узлов, серверов, хостов и ячеек

Зарезервированные имена: Избегайте использования зарезервированных имен в качестве значений полей. Использование зарезервированных имен может привести к непредсказуемым результатам. Зарезервированы следующие слова:

- cells
- nodes
- servers
- clusters
- applications
- deployments

Описание полей имен узлов и хостов, а также страниц имен узлов, хостов и ячеек: При создании профайлов используйте соответствующие рекомендации по именованию.

- Профайлы автономных серверов
- Профайлы администратора развертывания
- Профайлы управляемых узлов

Таблица 20. Рекомендации по именованию профайлов автономного сервера

Имя поля	Стандартное значение	Ограничения	Описание
Имя узла	Linux UNIX Windows *shortHostName* Node *NodeNumber*, где: • *shortHost Name* - краткое имя хоста. • *NodeNumber* - последовательный номер, начинающийся с 01.	Избегайте использования зарезервированных имен.	Выберите любое имя. В целях улучшения организации установки экземпляра используйте уникальное имя, если планируется создание нескольких серверов в системе.
Имя сервера	Linux UNIX Windows server1	Используйте уникальное имя сервера.	Логическое имя сервера.
Имя хоста	Linux UNIX Windows Длинная форма имени сервера имен доменов (DNS).	Используйте полное имя хоста, допускающее обращение к нему через вашу сеть.	Используйте фактическое имя DNS или IP-адрес рабочей станции, чтобы включить связь с ним. Дополнительная информация об имени хоста приведена после следующей таблицы.

Таблица 21. Рекомендации по именованию профайлов администратора развертывания

Имя поля	Стандартное значение	Ограничения	Описание
Имя узла	Linux UNIX Windows *shortHostName* Cell Manager*Node Number*, где: • *shortHost Name* - краткое имя хоста. • *NodeNumber* - последовательный номер, начинающийся с 01.	Используйте уникальное имя администратора развертывания. Избегайте использования зарезервированных имен.	Имя используется для администрирования в ячейке администратора развертывания.
Имя хоста	Linux UNIX Windows Длинная форма имени сервера имен доменов (DNS).	Используйте полное имя хоста, допускающее обращение к нему через вашу сеть. Избегайте использования зарезервированных имен.	Используйте фактическое имя DNS или IP-адрес рабочей станции, чтобы включить связь с ним. Дополнительная информация об имени хоста приведена после следующей таблицы.

Таблица 21. Рекомендации по именованию профайлов администратора развертывания (продолжение)

Имя поля	Стандартное значение	Ограничения	Описание
Имя ячейки	Linux UNIX Windows *shortHostName* Cell *CellNumber*, где: • *shortHost Name* - краткое имя хоста. • *CellNumber* - последовательный номер, начинающийся с 01.	Используйте уникальное имя ячейки администратора развертывания. Имя ячейки должно быть уникальным в любых ситуациях, при которых продукт работает на одной физической рабочей станции или в кластере из нескольких рабочих станций, таком как Sysplex. Кроме того, имя ячейки должно быть уникальным в любых ситуациях, при которых необходимо сетевое подключение между объектами: между ячейками или клиентом и каждой ячейкой. Имена ячеек также должны быть уникальными, если их пространства имен будут объединяться. В противном случае произойдет исключительная ситуация javax.naming.Name NotFoundException, при возникновении которой необходимо создать ячейки с уникальными именами.	Все объединенные узлы становятся элементами ячейки администратора развертывания, имя которой используется на странице Имена узлов, хостов и ячеек в инструменте управления профайлами.

Таблица 22. Рекомендации по именованию профайлов для управляемых узлов

Имя поля	Стандартное значение	Ограничения	Описание
Имя узла	Linux UNIX Windows *shortHostName* Node *NodeNumber*, где: • *shortHost Name* - краткое имя хоста. • *NodeNumber* - последовательный номер, начинающийся с 01.	Избегайте использования зарезервированных имен. Используйте уникальное имя в ячейке администратора развертывания.	Имя используется для администрирования в ячейке администратора развертывания, в которую добавляется профайл управляемого узла. Используйте уникальное имя в ячейке администратора развертывания.
Имя хоста	Linux UNIX Windows Длинная форма имени сервера имен доменов (DNS).	Используйте полное имя хоста, допускающее обращение к нему через вашу сеть.	Используйте фактическое имя DNS или IP-адрес рабочей станции, чтобы включить связь с ним. Дополнительная информация об имени хоста приведена после следующей таблицы.

Особенности указания имени хоста:

Имя хоста - сетевое имя физической рабочей станции, на которой установлен узел. Имя хоста должно определять физический узел сети на сервере. Если на сервере есть несколько сетевых карт, имя хоста или IP-адрес должны указывать на одну из них. Удаленные узлы используют имя хоста для подключения к нему и связи с ним.

IBM Business Process Manager соответствует протоколу IP версии 4 (IPv4) и версии 6 (IPv6). Пр вводе IP-адресов в административной консоли или в других программах можно использовать любой формат. Обратите внимание на то, что если в системе реализован протокол IPv6, то IP-адреса необходимо вводить в формате IPv6, и наоборот, если IPv6 недоступен, вводите IP-адреса в формате IPv4. Дополнительная информация о IPv6 приведена в следующем описании: IPv6.

Инструкции по определению правильного имени хоста рабочей станции:

- Выберите имя хоста для доступа к данной рабочей станции в сети.
- Не используйте общий идентификатор localhost в качестве этого значения.
- Не устанавливайте продукты IBM Business Process Manager на сервер, имя хоста которого использует символы двухбайтового набора (DBCS). Символы DBCS не поддерживаются в именах хостов.
- Не используйте нижнее подчеркивание (_) в именах серверов. Интернет-стандарты предусматривают, что имена доменов должны соответствовать требованиям к именам хостов, описанным в официальных стандартах протокола IP RFC 952 и RFC 1123. Имена доменов должны состоять только из букв (в нижнем или верхнем регистре) и цифр. Имена доменов могут также включать дефис (-), если они не находятся в конце имени. Символы нижнего подчеркивания (_) в именах хостов не поддерживаются. Если продукт IBM Business Process Manager установлен на сервере, имя которого содержит нижнее подчеркивание, используйте для доступа к этому серверу его IP-адрес до тех пор, пока он не будет переименован.

Если в одной системе выявлены параллельные узлы с уникальными IP-адресами, определите каждый IP-адрес в таблице сервера имен доменов (DNS). Файлы конфигурации серверов не поддерживают преобразование имени домена в адрес для нескольких IP-адресов на рабочей станции с одним сетевым адресом.

Значение, заданное для имени хоста, используется в качестве значения свойства hostName в документации конфигурации. Задайте значение имени хоста в одном из следующих форматов:

- Строка полного имени хоста серверов имен доменов (DNS), например xmachine.manhattan.ibm.com
- Строка стандартного краткого имени хоста DNS, например xmachine
- Числовой IP-адрес, например 127.1.255.3

Полное имя хоста DNS связано с такими преимуществами, как полная однозначность и гибкость. Можно изменять фактический IP-адрес хоста системы, не изменяя при этом конфигурацию сервера. Это значение имени хоста особенно полезно, если планируется частое изменение IP-адреса при использовании протокола настройки динамического хоста (DHCP) для назначения IP-адресов. Недостаток этого формата заключается в том, что он зависит от DNS. Если DNS недоступен, соединение отсутствует.

Краткое имя хоста может также распознаваться динамически. Краткое имя хоста позволяет повторное определение в файле локального хоста, поэтому система может запустить сервер даже при отключении от сети. Задайте краткое имя 127.0.0.1 (локальный циклический адрес) в файле хоста, чтобы работать в режиме отключения. Недостатком краткого имени является то, что он зависит от DNS при удаленном доступе. Если DNS недоступен, соединение отсутствует.

Числовой IP-адрес дает такое преимущество, как отсутствие необходимости преобразования имен через DNS. Удаленный узел может подключиться к узлу, которому присвоен числовой IP-адрес, даже если DNS недоступен. Недостаток этого формата заключается в том, что числовой IP-адрес является фиксированным. Для того чтобы изменить IP-адрес рабочей станции, необходимо изменить параметр свойства hostName в документации конфигурации. Таким образом, не используйте числовой IP-адрес, если используется протокол DHCP или если IP-адрес изменяется регулярно. Другим недостатком этого формата является то, что если хост отключен от сети, узел становится недоступным.

BPMN 2.0

Определения бизнес-процессов IBM Business Process Manager поддерживают производный класс Common Executable класса моделирования процессов BPMN 2.0, предназначенный для работы с исполняемыми моделями.

BPMN (Business Process Model and Notation) - это основополагающий стандарт для процессов IBM Process Designer и IBM Process Center. Диаграммы определений бизнес-процессов (BPD) соответствуют требованиям спецификации BPMN. В этом разделе рассмотрены отдельные способы применения BPMN 2.0 в IBM Business Process Manager. Подробная информация о BPMN приведена на странице спецификации BPMN: http://www.bpmn.org/.

IBM Business Process Manager поддерживает следующие типы задач BPMN 2.0:

- Нет (абстрактная задача в спецификации BPMN 2.0)
- Системная задача (служебная задача в спецификации BPMN 2.0)
- Пользовательская задача
- Сценарий
- Задача решения (задача бизнес-правила в спецификации BPMN 2.0)

Вместо задач отправки и приема BPMN можно использовать промежуточные события сообщений IBM BPM.

Нотация BPMN 2.0

Начиная с версии 7.5.1, значки задач BPMN 2.0 Process Designer на диаграммах BPD собраны на упрощенной палитре и отображаются на диаграммах процессов. Значок позволяет определить тип операции: системная задача, пользовательская задача, задача решения, сценарий или связанный процесс. В моделях, созданных с помощью предыдущих версий, при просмотре в версии 7.5.1 и выше для операций отображаются типы и значки задач BPMN 2.0.

Операции и задачи

По сравнению с предыдущими версиями Process Designer был внесен ряд изменений в терминологию. В частности, были переименованы отдельные типы операций.

- Операции службы (автоматические) теперь называются системными задачами.
- Операции службы (задачи), расположенные на несистемной полосе, теперь называются пользовательскими задачами.
- Операции службы (задачи), расположенные на системной полосе, теперь называются задачами решения, если они связаны со службой решения.
- Операции службы (задачи), расположенные на системной полосе, теперь называются системными задачами, если они связаны со службами, отличными от службы решения.
- Операции Javascript теперь называются задачами сценариев.
- Операции вложенных процессов теперь называются связанными процессами.
- Внешние операции из предыдущих версий Process Designer доступны в качестве внешних реализаций пользовательских или системных задач.

Шлюзы

Изменения нотаций по сравнению со шлюзами предыдущих версий отсутствуют. Однако внесены изменения в терминологию. Шлюз решения теперь называется *шлюзом исключающего ИЛИ*, шлюз простого разбиения или соединения теперь называется *параллельным шлюзом*, а шлюз условного разбиения или соединения теперь называется *шлюзом И*.

Кроме того, предусмотрен новый тип шлюза - *шлюз событий*. Шлюз событий представляет точку разветвления, альтернативные пути в которой выбираются с учетом возникающих событий, таких как прием

сообщения, а не вычисления выражений на основе данных процесса (как в случае включающих и исключающих шлюзов).

Непрерывающие события

В BPMN 2.0 добавлена нотация непрерывающих событий. По умолчанию граничное событие прерывает связанную операцию. В ответ на возникновение события операция останавливается с передачей маркера вниз по потоку последовательности события. Непрерывающее событие не останавливает связанную операцию, которая продолжает работу в параллельном режиме. При этом создается новый маркер, который передается вниз по потоку последовательности события. Граница непрерывающего события отображается в виде пунктирной линии.

Промежуточные события, связанные с операциями, называются прерывающими промежуточными событиями, если они закрывают связанные операции. В противном случае они называются непрерывающими промежуточными событиями.

Событие запуска

Спецификация BPMN допускает отсутствие событий запуска и завершения в модели процесса. В Process Designer наличие событий запуска и завершения в модели процесса обязательно.

В Process Designer доступны различные типы событий запуска:

процессы

- нет
- сообщение
- специальные

подпроцессы

- нет

подпроцессы событий

- ошибка
- сообщение
- таймер

Тип события запуска можно изменить в свойствах события. Процесс может содержать несколько событий запуска типа Сообщение, но только одно событие запуска типа Нет.

События завершения

Доступно четыре типа событий завершения: *сообщение*, *остановка*, *ошибка* и *нет*. При необходимости тип конечного события можно изменить.

Если дочерний процесс, вызванный родительским процессом, выполняет действие события остановки, то дочерний процесс останавливается, а родительский продолжает работу.

Подпроцессы

В спецификации BPMN определено два типа подпроцессов: встроенные и многоразовые. Оба типа доступны в Process Designer. Встроенные подпроцессы в Process Designer называются просто *подпроцессами*; это новое понятие в версии 7.5.1. Многоразовые подпроцессы BPMN в Process Designer называются *связанным процессом*.

Подпроцесс существует в родительском процессе и представляет собой способ группировки связанных элементов процесса. Подпроцессы группируют несколько шагов в одну операцию. Подпроцесс доступен

только родительскому процессу. Подпроцесс существует в области вызывающего процесса и обладает доступом ко всем переменным в этой среде. Для встроенного подпроцесса не предусмотрены входные и выходные параметры.

Помимо подпроцесса и связанного процесса в Process Designer применяется подпроцесс событий, который представляет собой специальный подпроцесс для обработки событий. Он не связан с другими операциями в потоке последовательности и выполняется только при возникновении конкретного события запуска.

Связанные процессы

Многоразовый подпроцесс BPMN называется *связанным процессом* в Process Designer. Он существует за пределами текущего процесса и может быть вызван текущим процессом. Он является многоразовым, поскольку его могут вызывать определения других процессов. Связанный процесс задает входные и выходные параметры и не обладает доступом к области или среде вызывающего процесса. Связанный процесс аналогичен вложенным процессам, которые поддерживались в предыдущих версиях; поведение операции не изменилось. Операции вложенных процессов теперь называются связанными процессами. Связанный процесс выглядит аналогично подпроцессу с толстой границей. Кроме того, он выделен в окне Inspector.

Циклы

В спецификации BPMN описана концепция повторяемой операции. Может повторяться отдельная операция или подпроцесс, содержащий несколько операций. При развертывании повторяемой операции отображаются вложенные операции, которые будут выполнены несколько раз. Условие всегда проверяется в начале каждого повторения цикла. Возможность проверки условия в конце повторения не предусмотрена.

В IBM Business Process Manager доступен *цикл с несколькими экземплярами*, который выполняется ограниченное число раз - вложенные операции выполняются последовательно или параллельно.

Импорт процессов, отличных от BPMN

Можно импортировать модели, созданные в IBM WebSphere Business Modeler, для применения в Process Designer. За информацией об импорте BPMN 2.0 обратитесь к разделу Связывание элементов IBM WebSphere Business Modeler с конструктивными элементами IBM Business Process Manager. При необходимости можно импортировать модели BPMN 2.0, созданные с помощью IBM WebSphere Business Compass, Rational Software Architect или других сред моделирования.

Определения бизнес-процессов (BPD)

Для моделирования процесса в IBM Process Designer необходимо создать определение бизнес-процесса (BPD). Определение бизнес-процесса может быть основано на импортированной модели BPMN.

Определение бизнес-процесса представляет собой многократно используемую модель процесса, описывающую общую часть всех экземпляров этой модели процесса во время выполнения. Определение BPD должно содержать событие запуска, событие завершения, по крайней мере одну полосу и одну или несколько операций. Дополнительная информация об ограничениях на использование символов в BPD приведена в разделе "Соглашения о присвоении имен в IBM Process Designer", указанном в ссылках на связанную информацию.

Определение бизнес-процесса (BPD) должно содержать полосу для каждой системы или группы пользователей, участвующих в процессе. Полоса может являться полосой участника или полосой системы. Однако при необходимости можно создать такое определение BPD, в котором операции группы и системы объединены в одной полосе. За информацией о создании BPD обратитесь к разделу "Создание определения бизнес-процесса (BPD)", указанном в ссылках на связанную информацию.

Любого пользователя или группу можно назначить ответственным за операции в полосе участника. По умолчанию любая созданная полоса назначается группе Все пользователи. BPD можно выполнить и протестировать в компоненте Inspector, используя эту группу по умолчанию. Группа Все пользователи содержит всех пользователей, которые входят в состав группы защиты **tw_allusers** - специальной группы защиты, в которую автоматически добавляются все пользователи системы.

Полоса системы содержит операции, выполняемые определенной системой IBM Process Center. У любой операции должна быть реализация, определяющая операцию и свойства задачи. Во время реализации разработчик создает службы или пишет сценарии JavaScript, необходимые для выполнения операций в полосе системы. Обратитесь к разделу "Описание типов служб", указанному в ссылках на связанную информацию, за информацией о службах.

При создании любого BPD необходимо объявить переменные, которые будут содержать бизнес-данные, передаваемые из одной операции в другую внутри процесса. Для получения информации о реализации переменных обратитесь к разделу "Управление переменными и создание связей", указанному в ссылках на связанную информацию.

При необходимости в BPD можно добавить события. В программе IBM BPM события могут возникать при наступлении определенной даты, при возникновении исключительной ситуации или при получении сообщения. Триггер определяет тип события, выбранного для реализации. За дополнительной информацией о доступных типах событий и их триггерах обратитесь к разделу "Моделирование событий".

Привязки

В основе архитектуры на основе служб лежит понятие *службы* - функциональной единицы, в работе которой используется взаимодействие между компьютерными устройствами. *Экспорт* определяет внешний интерфейс (или точку доступа) модуля, благодаря чему компоненты SCA модуля могут предоставлять свои службы внешним клиентам. *Импорт* определяет интерфейс доступа к службам, расположенным вне модуля, что позволяет вызывать службы внутри модуля. Вместе с объектами импорта и экспорта используются *привязки* определенных протоколов, определяющие способ передачи данных в модуль и из него.

Экспорты

Внешние клиенты могут вызывать компоненты SCA модуля интеграции с помощью различных протоколов (таких как HTTP, JMS, MQ и RMI/IIOP) с данными в различных форматах (в том числе XML, CSV, COBOL и JavaBeans). Объекты экспорта - это компоненты, которые получают запросы из внешних источников и вызывают компоненты IBM Business Process Manager, использующие модель программирования SCA.

Например, на следующем рисунке объект экспорта получает запрос по протоколу HTTP от приложения-клиента. Данные преобразуются в бизнес-объект - формат, применяемый компонентами SCA. После этого вызывается компонент, которому передается этот объект данных.

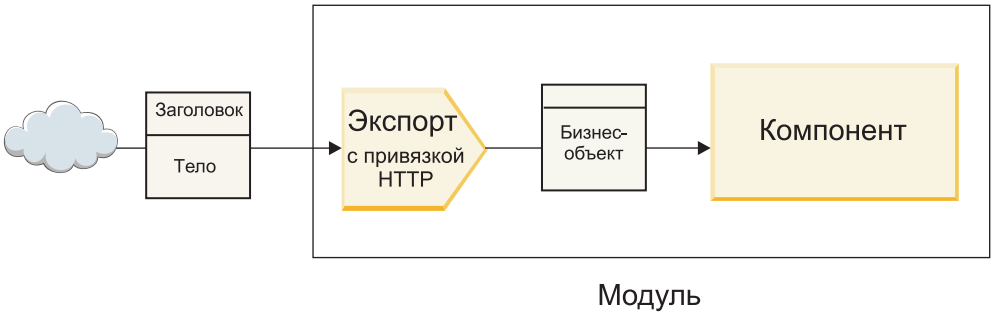


Рисунок 1. Экспорт с привязкой HTTP

Импорты

Компоненту SCA может потребоваться вызвать внешнюю службу, не поддерживающую SCA и принимающую данные в другом формате. Для вызова внешней службы в компоненте SCA используется объект импорта. Объект импорта вызывает целевую службу тем способом, который поддерживается этой службой.

Например, на следующем рисунке запрос компонента SCA отправляется во внешнюю службу через объект импорта. Бизнес-объект, то есть данные в формате, поддерживаемом в компоненте SCA, преобразуются в формат, ожидаемый службой, после чего вызывается служба.

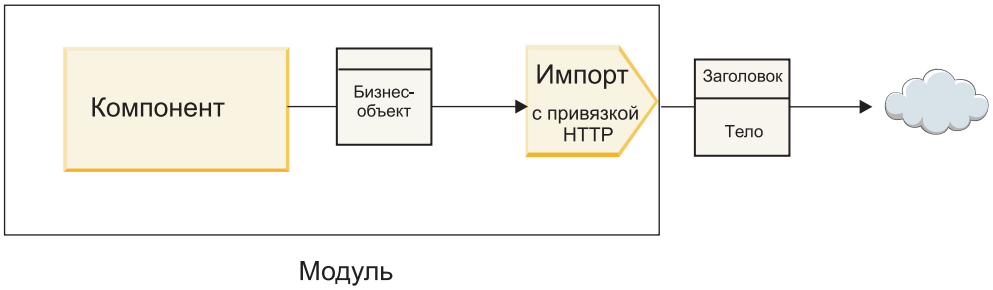


Рисунок 2. Импорт с привязкой HTTP

Список привязок

С помощью Integration Designer можно создать привязку для импорта или экспорта и настроить привязку. В следующем списке перечислены доступные типы привязок.

- SCA

  Привязка SCA, которая используется по умолчанию, позволяет службе взаимодействовать со службами из других модулей SCA. Используя импорт с привязкой SCA, можно получить доступ к службе из другого модуля SCA. Используя экспорт с привязкой SCA, можно предоставить службу для других модулей SCA.

- Веб-служба

  Привязка веб-службы позволяет получить доступ к внешней службе, используя универсальные сообщения SOAP и параметры качества обслуживания. Используя привязку веб-службы, можно добавлять прикрепления в сообщение SOAP.

  Привязка веб-службы может использовать транспортный протокол SOAP/HTTP (SOAP через HTTP) или SOAP/JMS (SOAP через JMS). Вне зависимости от типа транспортного протокола (HTTP или JMS), используемого для передачи сообщений SOAP, привязка веб-службы обеспечивает синхронное взаимодействие по типу запрос-ответ.

- HTTP

  Привязка HTTP позволяет получить доступ к внешней службе по протоколу HTTP, если используются сообщения, отличные от SOAP или необходим прямой доступ по HTTP.. Эта привязка используется при работе с веб-службами, основанными на модели HTTP (то есть службами, поддерживающими стандартные операции интерфейса HTTP, такие как GET, PUT, DELETE и т. п.).

- Enterprise JavaBean (EJB)

  Привязка EJB позволяет компонентам SCA взаимодействовать со службами, предоставляемыми бизнес-логикой Java EE, которая выполняется на сервере Java EE.

- EIS

  Привязка EIS (enterprise information system - информационная система предприятия), используемая совместно с адаптером ресурсов JCA, позволяет получать доступ к службам информационной системы предприятия или предоставлять свои службы для использования в EIS.

- Привязки JMS

Привязки Java Message Service (JMS), базового JMS и WebSphere MQ JMS (MQ JMS) используются для взаимодействия с системами доставки сообщений, если для обеспечения надежности необходимо асинхронное взаимодействие через очереди сообщений.

Объект экспорта с одной из привязок JMS отслеживает поступление сообщений в очередь и асинхронно отправляет ответ (если он есть) в очередь ответов. Объект импорта с одной из привязок JMS составляет и отправляет сообщение в очередь JMS и отслеживает поступление ответа в очередь.

– JMS

  Привязка JMS позволяет получить доступ к встроенному в WebSphere провайдеру JMS.

– Базовый JMS

  Привязка базового JMS позволяет получить доступ к системе доставки сообщений другого поставщика.

– MQ JMS

  Привязка MQ JMS позволяет получить доступ к подмножеству функций JMS из системы доставки сообщений WebSphere MQ. Эту привязку можно использовать в том случае, если для работы приложения достаточно этого подмножества функций JMS.

• MQ

  Привязка WebSphere MQ позволяет взаимодействовать с внутренними приложениями MQ, встраивая их в среду архитектуры на основе служб и предоставляя доступ к информации заголовка, специфичной для MQ. Эту привязку следует использовать в том случае, если планируется применять внутренние функции MQ.

Обзор привязок экспорта и импорта

Экспорт делает службы из модуля интеграции доступными для внешних клиентов, а импорт позволяет компонентам SCA в модуле интеграции вызывать внешние службы. Привязка, связанная с экспортом или импортом, определяет взаимосвязь между сообщениями протокола и бизнес-объектами. Кроме того, она определяет способ выбора операций и обработки ошибок.

Передача информации при экспорте

Объект экспорта получает запрос, адресованный компоненту, с которым экспорт связан проводником, через транспортный протокол, который определяется соответствующей привязкой (например, HTTP).

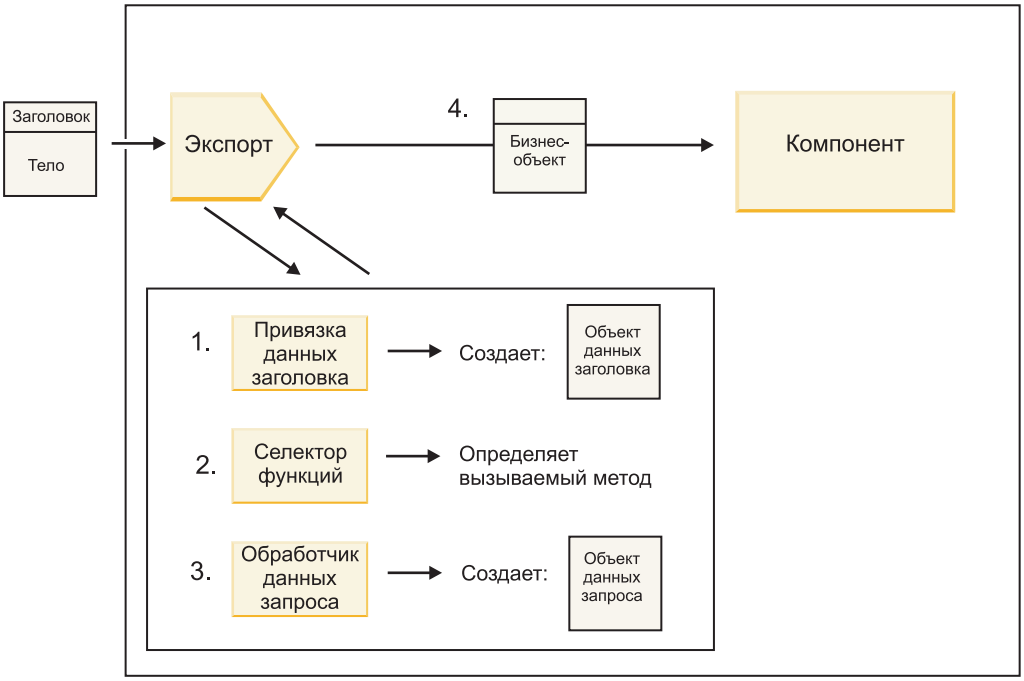


Рисунок 3. Передача запроса в компонент через объект экспорта

Когда объект экспорта получает запрос, выполняется следующая последовательность действий:

1. Для привязок WebSphere MQ: привязка данных заголовка преобразует заголовок протокола в объект данных заголовка.
2. Селектор функции определяет имя внутреннего метода, используя сообщение протокола. Имя внутреннего метода преобразуется конфигурацией экспорта в имя операции интерфейса экспорта.
3. Обработчик данных запроса или привязка данных в методе преобразует запрос в бизнес-объект запроса.
4. Объект экспорта вызывает метод компонента для бизнес-объекта запроса.
   - Привязка экспорта HTTP, привязка экспорта веб-службы и привязка экспорта EJB выполняют синхронный вызов компонента SCA.
   - Привязки экспорта JMS, базового JMS, MQ JMS и WebSphere MQ выполняют асинхронный вызов компонента SCA.

Обратите внимание, что при экспорте может передаваться информация о заголовках и пользовательских свойствах, полученных по протоколу, если включено распространение информации о контексте. В этом случае заголовки и пользовательские свойства доступны компонентам, связанным с экспортом. Более подробные сведения приведены в разделе “Распространение информации” справочной системы WebSphere Integration Developer Information Center.

В случае двунаправленной операции компонент возвращает ответ.

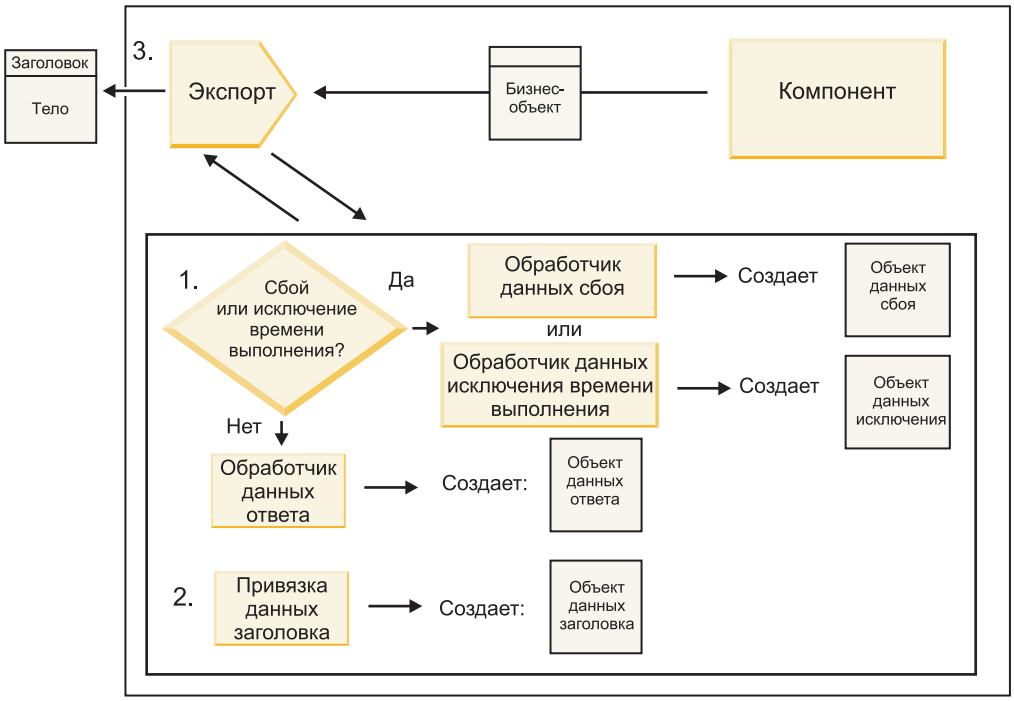


Рисунок 4. Возврат ответа через объект экспорта

Выполняется следующая последовательность действий:

1. Если привязка экспорта получает обычный ответ, то обработчик данных ответа или привязка данных в методе преобразуют бизнес-объект в ответ.

   Если в ответ получен сбой, то обработчик сбоев или привязка данных в методе преобразуют сбой в ответ о сбое.

   Только для привязок экспорта HTTP: если в ответ получена исключительная ситуация времени выполнения, то вызывается обработчик исключительных ситуаций, если он настроен.
2. Для привязок WebSphere MQ: привязка данных заголовка преобразует объекты данных заголовка в заголовки протокола.
3. Объект экспорта отправляет ответ службы по транспортному протоколу.

Передача информации при импорте

Компоненты отправляют запросы к службе, расположенной вне модуля, используя импорт. Запрос отправляется по транспортному протоколу, который определяется соответствующей привязкой.

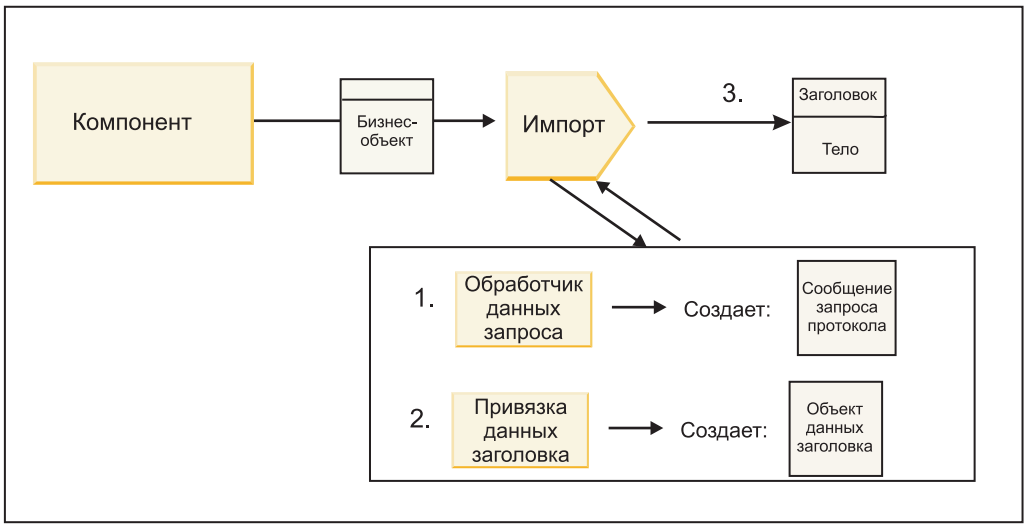


Рисунок 5. Передача данных из компонента в службу через объект импорта

Компонент вызывает функцию импорта, передавая в нее бизнес-объект запроса.

Примечание:

- Привязка импорта HTTP, привязка импорта веб-службы или привязка импорта EJB должна вызываться синхронно в компоненте.
- Привязка импорта JMS, базового JMS, MQ JMS или WebSphere MQ должна вызываться асинхронно.

После вызова функции импорта в компоненте выполняется следующая последовательность действий:

1. Обработчик данных запроса или привязка данных в методе преобразует бизнес-объект запроса в сообщение запроса протокола.
2. Для привязок WebSphere MQ: привязка данных заголовка из метода задает бизнес-объект заголовка в заголовке протокола.
3. Объект импорта вызывает службу с помощью запроса к службе, переданного через транспортный протокол.

Если выполняется двунаправленная операция, то служба возвращает ответ, и выполняется следующая последовательность действий:

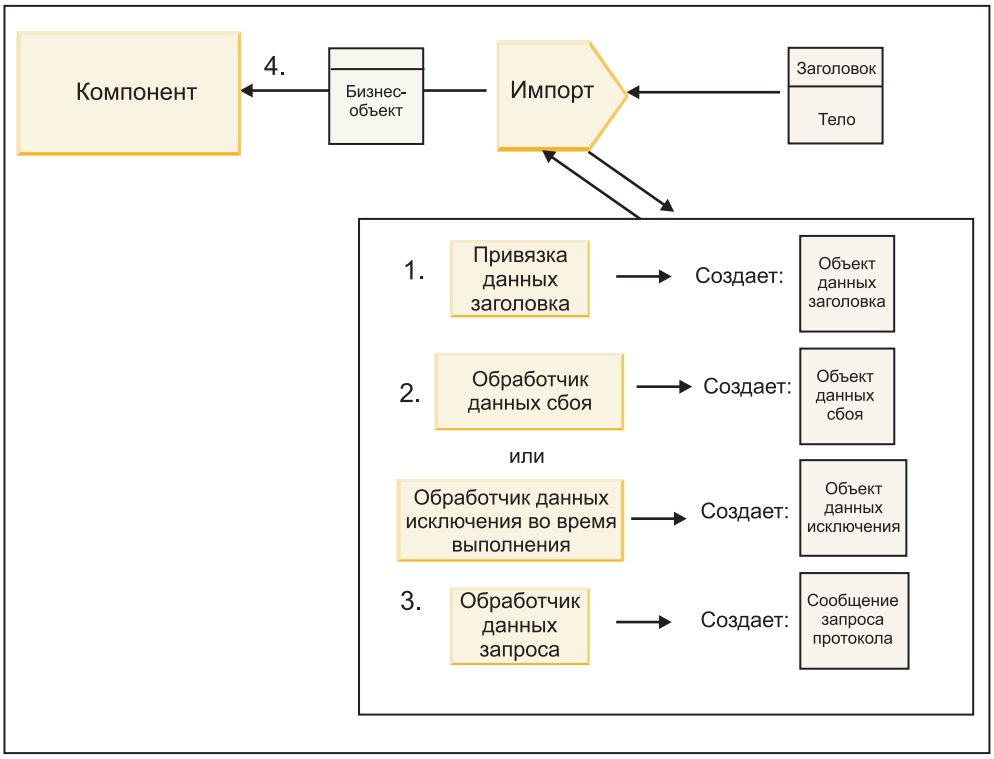


Рисунок 6. Возврат ответа через объект импорта

1. Для привязок WebSphere MQ: привязка данных заголовка преобразует заголовок протокола в объект данных заголовка.
2. Делается проверка того, является ли ответ сбоем.
   - Если ответ является сбоем, то селектор сбоев определяет соответствующий сбой WSDL. Обработчик сбоев из метода преобразует сбой в ответ о сбое.
   - Если ответ является исключительной ситуацией времени выполнения, то вызывается обработчик исключительных ситуаций, если он настроен.
3. Обработчик данных ответа или привязка из метода преобразуют ответ в бизнес-объект ответа.
4. Объект импорта возвращает бизнес-объект ответа в компонент.

Конфигурация привязки экспорта и импорта

Одной из основных особенностей привязок экспорта и импорта является преобразование (десериализация) данных из внутреннего формата проводника в бизнес-объект или преобразование (сериализация) бизнес-объекта во внутренний формат проводника. Для привязок экспорта можно дополнительно выбрать функцию, определяющую операцию над данными. И для привязок экспорта, и для привязок импорта можно указать способ обработки ошибок, возникающих во время обработки.

Кроме того, в привязках можно указать параметры, специфичные для типа транспорта. Например, для привязки HTTP можно указать URL конечной точки. Специфичная для транспорта информация, настраиваемая для привязки HTTP, описана в разделах “Создание привязки импорта HTTP” и “Создание привязки экспорта HTTP”. Информация о других привязках также приведена в Information Center.

Преобразование формата данных в объектах импорта и экспорта:

При настройке привязки экспорта или импорта в IBM Integration Designer в числе прочих параметров указывается формат данных привязки.

- Для привязок экспорта, через которые приложение-клиент отправляет запросы и получает ответы от компонента SCA, требуется указывать формат внутренних данных. Система выбирает обработчик данных

или привязку данных с учетом формата для преобразования внутренних данных в бизнес-объект (который применяется в компоненте SCA) и для обратного преобразования бизнес-объекта во внутренние данные (то есть в ответ для приложения-клиента).

- Для привязок импорта, через которые компонент SCA отправляет запросы и получает ответы от внешней службы, необходимо указывать формат внутренних данных. Система выбирает обработчик данных или привязку данных с учетом формата для преобразования бизнес-объекта во внутренние данные и обратного преобразования.

IBM Business Process Manager предоставляет набор предопределенных форматов данных, а также обработчиков данных или привязок данных, поддерживающих эти форматы. При необходимости можно создать собственные обработчики данных и зарегистрировать для них формат данных. За дополнительной информацией обратитесь к разделу “Разработка обработчиков данных” в справочной системе IBM Integration Designer Information Center.

- *Обработчики данных* не зависят от протокола и выполняют преобразование данных в другой формат. В IBM Business Process Manager обработчики данных как правило преобразуют внутренние данные (такие как XML, CSV или COBOL) в бизнес-объект, а также бизнес-объект во внутренние данные. Поскольку обработчики не зависят от протокола, один обработчик можно использовать в различных привязках экспорта и импорта. Например, обработчик данных XML можно использовать как с привязками экспорта и импорта HTTP, так и с привязками экспорта и импорта JMS.
- *Привязки данных* также выполняют преобразование внутренних данных в бизнес-объект (и наоборот), однако они зависят от протокола. Например, привязку данных HTTP можно использовать только с привязками экспорта и импорта HTTP. В отличии от обработчиков данных, привязку данных HTTP нельзя использовать в привязке импорта или экспорта MQ.

  Примечание: Три привязки данных HTTP (HTTPStreamDataBindingSOAP, HTTPStreamDataBindingXML и HTTPServiceGatewayDataBinding) устарели и больше не используются в IBM Business Process Manager версии 7.0. При любой возможности вместо них следует использовать обработчики данных.

Как уже было отмечено ранее, при необходимости можно создать собственные обработчики данных. Кроме того, можно создать свои привязки данных, однако рекомендуется создавать именно обработчики данных, поскольку их можно использовать в любых привязках.

Обработчики данных:

Обработчики данных настраиваются для привязок импорта и экспорта и предназначены для преобразования данных в другой формат, не связанный с определенным протоколом. Несколько обработчиков данных входят в состав продукта и при необходимости можно создать собственный обработчик данных. Обработчик данных можно связать с привязкой импорта или экспорта на одном из двух уровней: его можно связать со всеми операциями интерфейса экспорта или импорта либо с определенной операцией над запросом или ответом.

Предопределенные обработчики данных

Необходимый обработчик данных можно выбрать в IBM Integration Designer.

В следующей таблице перечислены предопределенные обработчики данных и описано, каким образом они выполняют преобразование входящих и исходящих данных.

Примечание: За исключением специально отмеченных случаев, обработчики данных можно использовать вместе с привязками JMS, базового JMS, MQ JMS, WebSphere MQ и HTTP.
За более подробной информацией обратитесь к разделу “Обработчики данных” справочной системы Integration Designer Information Center.

Таблица 23. Предопределенные обработчики данных

Обработчик данных	Внутренние данные в бизнес-объект	Бизнес-объект во внутренние данные
ATOM	Преобразует ленты новостей ATOM в бизнес-объект ленты ATOM.	Сериализует бизнес-объект ленты ATOM и преобразует его в ленты новостей ATOM.
С разделителем	Преобразует данные с разделителем в бизнес-объект.	Сериализует бизнес-объект и преобразует его в данные с разделителем, в том числе CSV.
Фиксированной ширины	Преобразует данные фиксированной ширины в бизнес-объект.	Сериализует бизнес-объект и преобразует его в данные фиксированной ширины.
Обрабатывается WTX	Передает данные для преобразования в WebSphere Transformation Extender (WTX). Имя карты связей WTX определяется обработчиком данных.	Передает данные для преобразования в WebSphere Transformation Extender (WTX). Имя карты связей WTX определяется обработчиком данных.
Обрабатывается WTX Invoker	Передает данные для преобразования в WebSphere Transformation Extender (WTX). Имя карты связей WTX предоставляется пользователем.	Передает данные для преобразования в WebSphere Transformation Extender (WTX). Имя карты связей WTX предоставляется пользователем.
JAXB	Преобразует объекты Java Bean в бизнес-объект с помощью правил преобразования, определенных в спецификации Java Architecture for XML Binding (JAXB).	Десериализует бизнес-объект и преобразует его в объекты Java Bean с помощью правил преобразования, определенных в спецификации JAXB.
JAXWS **Примечание:** Обработчик данных JAXWS можно использовать только в привязке EJB.	Применяется в привязке EJB для преобразования объекта Java с ответом или исключительной ситуацией в бизнес-объект ответа согласно правилам преобразования, определенным в спецификации Java API for XML Web Services (JAX-WS).	Применяется в привязке EJB для преобразования бизнес-объекта в параметры метода Java в соответствии с правилами преобразования, определенными в спецификации JAX-WS.
JSON	Преобразует данные JSON в бизнес-объект.	Сериализует бизнес-объект и преобразует его в данные JSON.
Внутреннее тело	Преобразует внутренние байты, текст, карту, поток или объект в бизнес-объект одного из пяти типов (текст, байты, карта, поток или объект).	Преобразует любой из пяти типов базовых бизнес-объектов в байты, текст, карту, поток или объект.
SOAP	Преобразует сообщение SOAP (включая заголовок) в бизнес-объект.	Сериализует бизнес-объект и преобразует его в сообщение SOAP.
XML	Преобразует данные XML в бизнес-объект.	Сериализует бизнес-объект и преобразует его в данные XML.
UTF8XMLDataHandler	Преобразует данные XML в кодировке UTF-8 в бизнес-объект.	Сериализует бизнес-объект и преобразует его в данные XML в кодировке UTF-8 при отправке сообщения.

Создание обработчика данных

Подробная информация о создании обработчика данных приведена в разделе “Разработка обработчиков данных” справочной системы Integration Designer Information Center.

Привязки данных:

Привязки данных настраиваются для привязок импорта и экспорта и предназначены для преобразования данных в другой формат. Для каждого протокола предусмотрена своя привязка данных. Несколько привязок данных входят в состав продукта и при необходимости можно создать собственную привязку данных. Привязку данных можно связать с привязкой импорта или экспорта на одном из двух уровней: ее можно связать со всеми операциями интерфейса экспорта или импорта либо с определенной операцией над запросом или ответом.

Для выбора необходимой привязки данных и создания собственной привязки данных используется IBM Integration Designer. Инструкции по созданию привязок данных приведены в разделе “Обзор привязок JMS, MQ JMS и базового JMS” в справочной системе IBM Integration Designer Information Center.

Привязки JMS

В следующей таблице перечислены привязки данных, которые можно использовать вместе со следующими объектами:

- Привязки JMS
- Базовые привязки JMS
- Привязки WebSphere MQ JMS

В таблице приведено описание задач, выполняемых привязками данных.

Таблица 24. Предопределенные привязки данных для привязок JMS

Привязка данных	Внутренние данные в бизнес-объект	Бизнес-объект во внутренние данные
Сериализованный объект Java	Преобразует сериализованный объект Java в бизнес-объект (который преобразуется в тип входа или выхода в WSDL).	Преобразует бизнес-объект в сериализованный объект Java из сообщения объекта JMS.
Байты с оболочкой	Извлекает байты из входящего байтового сообщения JMS и преобразует их в бизнес-объект JMSBytesBody.	Извлекает байты из бизнес-объекта JMSBytesBody и добавляет их внутрь исходящего байтового сообщения JMS
Запись карты связей с оболочкой	Извлекает имя, значение и тип из каждой записи входящего сообщения карты связей JMS и создает список бизнес-объектов MapEntry. Полученный список добавляется в бизнес-объект JMSMapBody	Извлекает имя, значение и тип из списка MapEntry, расположенного внутри бизнес-объекта JMSMapBody, и создает соответствующие записи в исходящем сообщении карты связей JMS.
Объект с оболочкой	Извлекает объект из входящего сообщения объекта JMS и добавляет его в бизнес-объект JMSObjectBody.	Извлекает объект из бизнес-объекта JMSObjectBody и добавляет его в исходящее сообщение объекта JMS.
Текст с оболочкой	Извлекает текст из входящего текстового сообщения JMS и добавляет его в бизнес-объект JMSTextBody.	Извлекает текст из бизнес-объекта JMSTextBody и добавляет его в исходящее текстовое сообщение JMS.

Привязки WebSphere MQ

В следующей таблице перечислены привязки данных, которые можно использовать вместе с WebSphere MQ, и описаны задачи, выполняемые этими привязками.

Таблица 25. Предопределенные привязки данных для привязок WebSphere MQ

Привязка данных	Внутренние данные в бизнес-объект	Бизнес-объект во внутренние данные
Сериализованный объект Java	Преобразует сериализованный объект Java из входящего сообщения в бизнес-объект (который преобразуется в тип входа или выхода в WSDL).	Преобразует бизнес-объект в сериализованный объект Java в исходящем сообщении
Байты с оболочкой	Извлекает байты из неструктурированного байтового сообщения MQ и преобразует их в бизнес-объект JMSBytesBody.	Извлекает байты из бизнес-объекта JMSBytesBody и преобразует байты в исходящее неструктурированное байтовое сообщение MQ.
Текст с оболочкой	Извлекает текст из неструктурированного текстового сообщения MQ и преобразует его в бизнес-объект JMSTextBody.	Извлекает текст из бизнес-объекта JMSTextBody и преобразует его в неструктурированное текстовое сообщение MQ.
Запись потока с оболочкой	Извлекает имя и тип из каждой записи входящего потокового сообщения JMS и создает список бизнес-объектов StreamEntry. Полученный список добавляется в бизнес-объект JMSStreamBody.	Извлекает имя и тип из списка StreamEntry, содержащегося в бизнес-объекте JMSStreamBody, и создает соответствующие записи в исходящем сообщении JMSStreamMessage.

В дополнение к привязкам данных, описанным в разделе Табл. 25, WebSphere MQ поддерживает привязки данных заголовка. Дополнительная информация приведена в IBM Integration Designer Information Center.

Привязки HTTP

В следующей таблице перечислены привязки данных, которые могут применяться при работе с HTTP, и описаны задачи, выполняемые этими привязками данных.

Таблица 26. Предопределенные привязки данных для привязок HTTP

Привязка данных	Внутренние данные в бизнес-объект	Бизнес-объект во внутренние данные
Байты с оболочкой	Извлекает байты из текла входящего сообщения HTTP и добавляет их в бизнес-объект HTTPBytes.	Извлекает байты из бизнес-объекта HTTPBytes и добавляет их в тело исходящего сообщения HTTP.
Текст с оболочкой	Извлекает текст из тела входящего сообщения HTTP и добавляет его в бизнес-объект HTTPText.	Извлекает текст из бизнес-объекта HTTPText и добавляет его в тело исходящего сообщения HTTP.

Селекторы функций в привязках экспорта:

Селектор функции указывает, какую операцию над данными необходимо выполнить при получении сообщения с запросом. Селекторы функций настраиваются в привязке экспорта.

Для примера рассмотрим экспорт SCA, предназначенный для экспорта интерфейса. Интерфейс содержит две операции: Create и Update. Экспорт использует привязку JMS, которая считывает сообщения из очереди.

Когда сообщение поступает в очередь, данные о нем передаются в объект экспорта, но дополнительно требуется определить, какую из двух операций интерфейса следует вызвать для связанного компонента. Операция выбирается с помощью селектора функций и конфигурации привязки экспорта.

Селектор функций возвращает внутреннее имя функции (имя функции в клиентской системе, отправившей сообщение). Внутреннее имя функции преобразуется в имя операции или функции интерфейса, связанного с экспортом. Например, на следующем рисунке селектор функций возвращает внутреннее имя функции (CRT)

из полученного сообщения, это имя преобразуется в операцию Create, после чего компоненту SCA отправляется бизнес-объект с операцией Create.

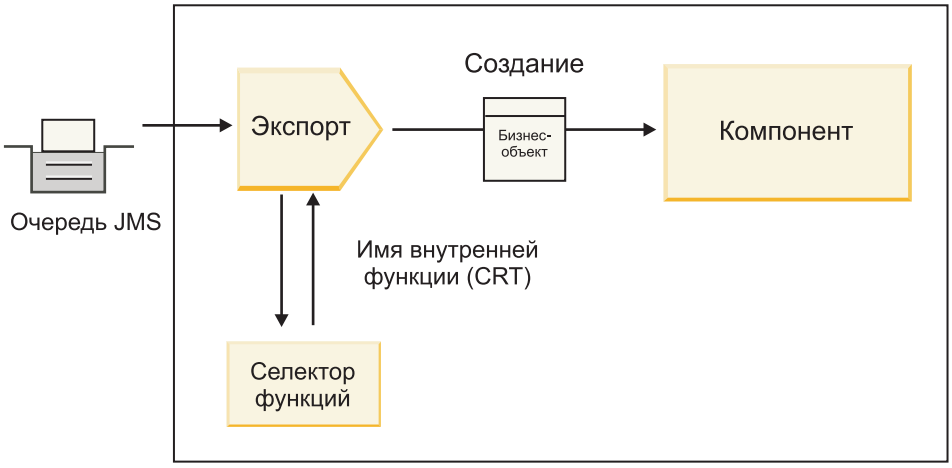


Рисунок 7. Селектор функций

Если интерфейс содержит только одну операцию, то селектор функций указывать не нужно.

В составе продукта поставляется несколько готовых селекторов функций, которые описаны в приведенных ниже разделах.

Привязки JMS

В следующей таблице указаны селекторы функций, которые можно использовать вместе со следующими объектами:

- Привязки JMS
- Базовые привязки JMS
- Привязки JMS WebSphere MQ

Таблица 27. Предопределенные селекторы функций для привязок JMS

Селектор функций	Описание
Селектор функций JMS для простых привязок данных JMS	Для выбора имени операции применяется свойство сообщения JMSType.
Селектор функций для свойства заголовка JMS	Возвращает значение строкового свойства JMS с именем TargetFunctionName из заголовка.
Селектор функций для шлюза служб JMS	Определяет тип запрашиваемой операции (однонаправленная или двунаправленная), используя заданное клиентом свойство JMSReplyTo.

Привязки WebSphere MQ

В следующей таблице указаны селекторы функций, которые можно применять в привязках WebSphere MQ.

Таблица 28. Предопределенные селекторы функций для привязок WebSphere MQ

Селектор функций	Описание
Селектор функций handleMessage MQ	Возвращает handleMessage в качестве значения, которое преобразуется с помощью привязок метода объекта экспорта в имя операции интерфейса.

Таблица 28. Предопределенные селекторы функций для привязок WebSphere MQ (продолжение)

Селектор функций	Описание
В MQ используется селектор функций по умолчанию JMS	Считывает внутреннюю операцию из свойства TargetFunctionName папки в заголовке MQRFH2.
В MQ используется формат тела сообщения в качестве внутренней функции	Находит поле формата в последнем заголовке и возвращает его значение в виде строки.
Селектор функций для типа MQ	Создает метод в привязке экспорта путем получения данных по URL, содержащему свойства Msd, Set, Type и Format и указанному в заголовке MQRFH2.
Селектор функций для шлюза служб MQ	Определяет имя операции с помощью свойства MsgType из заголовка MQMD.

Привязки HTTP

В следующей таблице указаны селекторы функций, которые можно использовать в привязках HTTP.

Таблица 29. Предопределенные селекторы функций для привязок HTTP

Селектор функций	Описание
Селектор функций HTTP, основанный на заголовке TargetFunctionName	Использует настроенное клиентом свойство заголовка HTTP TargetFunctionName для определения операции, которую следует вызывать во время выполнения из объекта экспорта.
Селектор функций HTTP, основанный на URL и методе HTTP	Использует относительный путь из URL и дополняет его методом HTTP, указанным клиентом, для выбора внутренней операции, определенной в экспорте.
Селектор функций для шлюза служб HTTP, основанный на URL и имени операции	Выбирает вызываемый метод по URL, если в URL запроса указано "operationMode = oneway".

Примечание: Используя IBM Integration Designer, можно создать собственный селектор функций. Информация о создании селектора функций приведена в справочной системе IBM Integration Designer Information Center. Например, инструкции по созданию селектора функций для привязок WebSphere MQ приведены в разделе “Обзор селекторов функций MQ”.

Обработка сбоев:

Привязки импорта и экспорта можно настроить для обработки сбоев (например, исключительных ситуаций бизнес-уровня), возникающих во время работы, указав обработчики данных сбоев. Обработчик данных сбоев можно настроить на трех уровнях: обработчик данных сбоев можно связать со сбоем, с операцией или со всеми операциями привязки.

Обработчик данных сбоев обрабатывает данные сбоя и преобразует их в тот формат, в котором они могут быть отправлены привязкой импорта или экспорта.

- Для привязки экспорта обработчик данных сбоев преобразует бизнес-объект исключительной ситуации, полученный от компонента, в ответное сообщение, которое может быть обработано приложением-клиентом.
- Для привязки импорта обработчик данных сбоев преобразует данные о сбое или ответное сообщение, полученное от службы, в бизнес-объект исключительной ситуации, который может быть обработан компонентом SCA.

Привязка импорта вызывает селектор сбоев, который определяет, является ли ответное сообщение обычным ответом, сбоем бизнес-уровня или исключительной ситуацией времени выполнения.

С помощью привязки можно настроить обработчик данных сбоев для отдельного сбоя, операции или всех операций.

- Если обработчик данных сбоев настроен на всех трех уровнях, то вызывается тот из них, который связан со сбоем.
- Если обработчик данных сбоев настроен на уровне операции и привязки, то вызывается обработчик, связанный с операцией.

Для настройки обработки сбоев в IBM Integration Designer используются два редактора. В редакторе интерфейса можно определить сбои операции. После создания привязки с этим интерфейсом редактор на панели свойств позволяет настроить способ обработки сбоев. За дополнительной информацией обратитесь к разделу “Селектор сбоя” в справочной системе IBM Integration Designer Information Center.

Обработка сбоев в привязках экспорта:

Если во время обработки запроса приложения-клиента возникает сбой, то привязка экспорта может вернуть информацию о нем клиенту. Параметры обработки и возврата сбоев настраиваются в привязке экспорта.

Привязка экспорта настраивается с помощью IBM Integration Designer.

Во время обработки запроса клиент вызывает объект экспорта, передавая ему запрос, а объект экспорта вызывает компонент SCA. Во время обработки запроса компонент SCA может вернуть ответ с данными или сгенерировать исключительную ситуацию бизнес-уровня либо исключительную ситуацию времени выполнения. В последнем случае привязка экспорта преобразует исключительную ситуацию в сообщение о сбое и отправляет его клиенту, как показано на следующем рисунке и описано в приведенных ниже разделах.

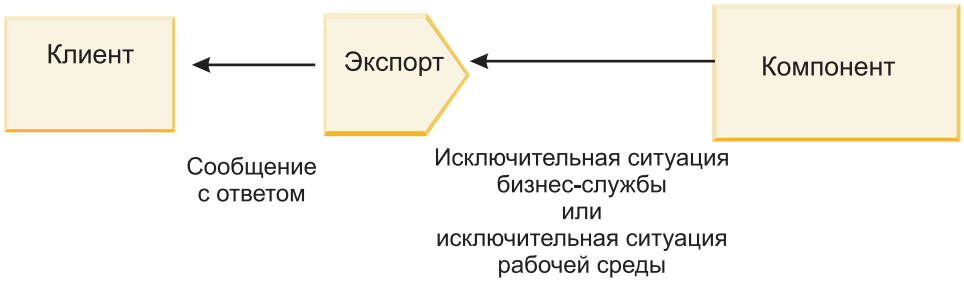


Рисунок 8. Как информация о сбое передается из компонента в клиент через привязку экспорта

Для обработки сбоев можно создать пользовательский обработчик данных или привязку данных.

Сбои бизнес-уровня

Сбои бизнес-уровня - это ошибки или исключительные ситуации в работе бизнеса, которые возникают во время обработки.

Изучите следующий интерфейс, содержащий операцию createCustomer. В этой операции определено два сбоя бизнес-уровня: CustomerAlreadyExists и MissingCustomerId.

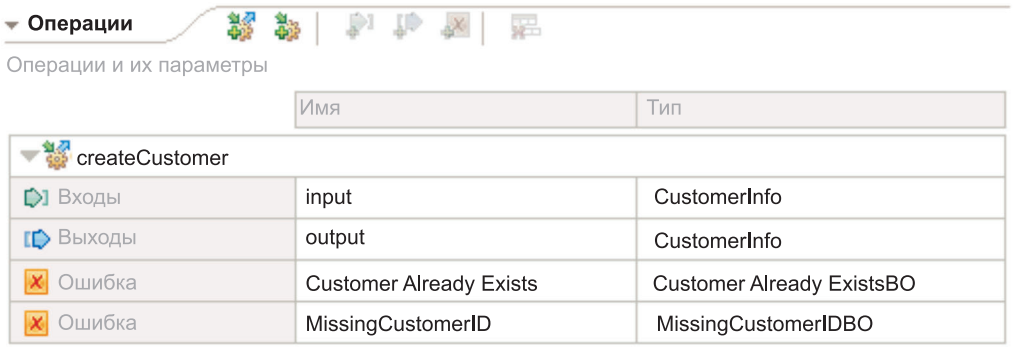


Рисунок 9. Интерфейс с двумя сбоями

В этом примере при попытке клиента создать уже существующего получателя (для компонента SCA) компонент генерирует сбой CustomerAlreadyExists и передает его в объект экспорта. Объект экспорта должен вернуть этот сбой бизнес-уровня тому клиенту, который отправил запрос. Для этого используется обработчик данных сбоя, который настроен в привязке экспорта.

Если привязка экспорта получает сбой бизнес-уровня, выполняются следующие действия:

1. Привязка определяет, какой обработчик данных сбоя следует вызвать для обработки данного сбоя. Если исключительная ситуация бизнес-уровня содержит имя сбоя, то вызывается настроенный для этого сбоя обработчик данных. В противном случае имя сбоя определяется по типу сбоя.
2. Привязка вызывает обработчик данных сбоя и передает в него объект данных из исключительной ситуации бизнес-уровня.
3. Обработчик данных сбоя преобразует объект данных сбоя в ответное сообщение и возвращает его в привязку экспорта.
4. Объект экспорта возвращает ответное сообщение клиенту.

Если исключительная ситуация бизнес-уровня содержит имя сбоя, то вызывается настроенный для этого сбоя обработчик данных. В противном случае имя сбоя определяется по типу сбоя.

Исключительные ситуации времени выполнения

Исключительная ситуация времени выполнения - это исключительная ситуация, которая возникает в приложении SCA во время обработки запроса и не связана со сбоем бизнес-уровня. В отличие от сбоев бизнес-уровня исключительные ситуации времени выполнения не требуется определять в интерфейсе.

В ряде случаев имеет смысл передавать исключительные ситуации времени выполнения в приложение-клиент, для того чтобы это приложение могло выполнить соответствующие действия.

Например, если клиент отправляет запрос на создание получателя в компонент SCA, и во время обработки запроса возникает ошибка прав доступа, то компонент генерирует исключительную ситуацию времени выполнения. Эту исключительную ситуацию необходимо вернуть в клиент, для того чтобы он принял необходимые меры по получению прав доступа. Для этого применяется обработчик данных исключительной ситуации времени выполнения, настроенный в привязке экспорта.

Примечание: Обработчик данных исключительной ситуации времени выполнения можно настроить только в привязках HTTP.

Исключительная ситуация времени выполнения обрабатывается так же, как и сбой бизнес-уровня. Если обработчик данных исключительной ситуации времени выполнения настроен, то выполняются следующие действия:

1. Привязка экспорта вызывает необходимый обработчик данных и передает в него исключительную ситуацию времени выполнения.

2. Обработчик данных преобразует объект данных сбоя в ответное сообщение и возвращает его в привязку экспорта.
3. Объект экспорта возвращает ответное сообщение клиенту.

Обработка сбоев и исключительных ситуаций времени выполнения является необязательной. Если сбои или исключительные ситуации не требуется передавать клиенту, то не настраивайте обработчик данных сбоев или исключительной ситуации времени выполнения.

Обработка ошибок привязками импортов:

Некоторый компонент использует импорт для отправки запроса некоторой службе вне модуля. Если во время обработки запроса происходит ошибка, то служба возвращает ошибку привязке импорта. Привязку импорта можно настроить с указанием, как должна быть обработана ошибка и возвращена компоненту.

Для настройки привязки импорта используется IBM Integration Designer. Также можно указать обработчик данных ошибок (или привязку данных), а также указать селектор ошибок.

Обработчики данных ошибок

Служба, которая обрабатывает отправки запросов привязке импорта, информацию об ошибке в виде исключительной ситуации или ответное сообщение, которое содержит данные ошибки.

Привязка импорта преобразует исключительную ситуацию службы или ответное сообщение в исключительную ситуации бизнес-процесса службы или исключительную ситуацию во время выполнения службы, как показано на следующем рисунке и описано в следующих разделах.

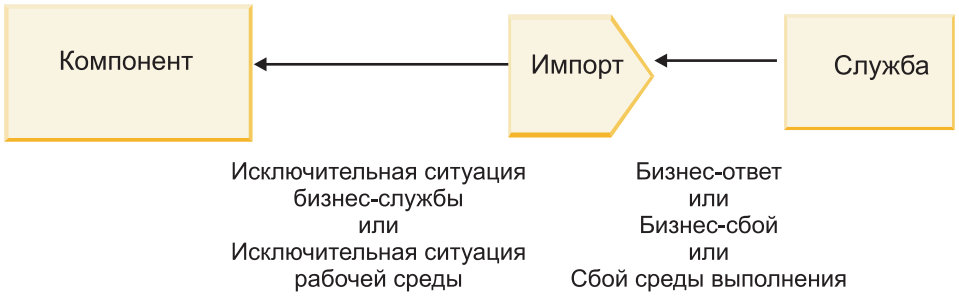


Рисунок 10. Отправка информации об ошибке из службы через импорт компоненту

Для обработки ошибок можно создать пользовательский обработчик данных или привязку данных.

Селекторы сбоев

При настройке привязки импорта можно задать селектор ошибок. Селектор ошибок определяет, является ли ответ импорта действительно ответом, исключительной ситуацией бизнес-процесса или ошибкой во время выполнения. Из тела или заголовка ответа он также определяет собственное имя ошибки, которое преобразуется конфигурацией привязки в имя ошибки в связанном интерфейсе.

Для работы с импортами JMS, MQ JMS, Generic JMS, WebSphere MQ и HTTP доступны два типа готовых селекторов ошибок:

Таблица 30. Готовые селекторы ошибок

Тип селектора ошибок	Описание
На основе заголовков	На основе заголовков входящего ответного сообщения определяет, является ли ответное сообщение ошибкой бизнес-процесса, исключительной ситуацией во время выполнения или нормальным сообщением.
SOAP	Определяет, является ли ответное сообщение SOAP нормальным ответом, ошибкой бизнес-процесса или исключительной ситуацией во время выполнения.

Далее приведены примеры селекторов ошибок на основе заголовков и селектора ошибок SOAP.

- Селектор ошибок на основе заголовков

  Для того чтобы приложение могло показать, что входящее сообщение является ошибкой бизнес-процесса, для ошибок бизнес-процесса во входящем сообщении должно быть два заголовка, которые выглядят следующим образом:

  ```
  Имя заголовка = FaultType, Значение заголовка = Бизнес-процесс
  Имя заголовка = FaultName, Значение заголовка = <пользовательское собственное имя ошибки>
  ```

  Для того чтобы приложение могло показать, что входящее сообщение является исключительной ситуацией во время выполнения, во входящем сообщении должен быть один заголовок, который выглядят следующим образом:

  ```
  Имя заголовка = FaultType, Значение заголовка = Во время выполнения
  ```

- Селектор ошибок SOAP

  Ошибка бизнес-процесса может быть передана в составе сообщения SOAP со следующим пользовательским заголовком SOAP. "CustomerAlreadyExists" – это имя ошибки в данном случае.

  ```
  <ibmSoap:BusinessFaultName
  xmlns:ibmSoap="http://www.ibm.com/soap">CustomerAlreadyExists
  <ibmSoap:BusinessFaultName>
  ```

Селектор ошибок произволен. Если селектор ошибок не указан, то привязка импорта не сможет определить тип ответа. При этом привязка рассматривает его как ответ бизнес-процесса и вызывает обработчик данных ответа или привязку данных.

Можно создать пользовательский селектор ошибок. Этапы создания пользовательского селектора ошибок приведены в разделе “Разработка пользовательского селектора ошибок” справочной системы IBM Integration Designer Information Center.

Ошибки бизнес-процессов

Ошибка бизнес-процесса может наблюдаться, когда в обработке запроса присутствует ошибка. Например, в случае отправки запроса на создание клиента, который уже существует, служба отправляет исключительную ситуацию бизнес-процесса привязке импорта.

После получения исключительной ситуации бизнес-процесса привязкой этапы обработки зависят от того, настроен ли селектор ошибок для привязки.

- Если селектор ошибок не настроен, то привязка вызывает обработчик данных ответа или привязку данных.
- Если селектор ошибок настроен, то выполняется следующая обработка:
  1. Привязка импорта вызывает селектор ошибок для определения, является ли ответ ошибкой бизнес-процесса, ответом бизнес-процесса или ошибкой во время выполнения.
  2. Если ответ является ошибкой бизнес-процесса, то привязка импорта вызывает селектор ошибок для определения собственного имени ошибки.

3. Привязка импорта определяет ошибку WSDL соответственно собственному имени ошибки, возвращенному селектором ошибок.
4. Привязка импорта определяет обработчик данных ошибки, который настроен на ошибку WSDL.
5. Привязка импорта вызывает обработчик данных ошибки с данными ошибки.
6. Обработчик данных ошибки преобразует данные ошибки в объект данных и возвращает его привязке импорта.
7. Привязка импорта создает объект исключительной ситуации бизнес-процесса службы с объектом данных и именем ошибки.
8. Импорт возвращает объект исключительной ситуации бизнес-процесса службы компоненту.

Исключительные ситуации выполнения

Исключительная ситуация во время выполнения может наблюдаться, когда имеет место неполадка со связью со службой. Обработка исключительной ситуации во время выполнения аналогична обработке исключительной ситуации бизнес-процесса. Если селектор ошибок настроен, то выполняется следующая обработка:

1. Привязка импорта вызывает соответствующий обработчик данных исключительной ситуации во время выполнения с данными исключительной ситуации.
2. Обработчик данных исключительной ситуации во время выполнения преобразует данные исключительной ситуации в объект исключительной ситуации во время выполнения службы и возвращает его привязке импорта.
3. Импорт возвращает объект исключительной ситуации во время выполнения службы компоненту.

Взаимодействие между модулями SCA и службами Open SCA

IBM WebSphere Application Server V7.0 Feature Pack for Service Component Architecture (SCA) предоставляет простую, но мощную программную модель для создания приложений на основе спецификаций Open SCA. Модули SCA IBM Business Process Manager используют привязки импортов и экспортов для взаимодействия со службами Open SCA, разработанными в среде Rational Application Developer и находящимися в WebSphere Application Server Feature Pack for Service Component Architecture.

Приложение SCA вызывает приложение Open SCA при помощи привязки импорта. Приложение SCA получает вызов от приложения Open SCA при помощи привязки экспорта. Список поддерживаемых привязок приведен в “Вызов служб через стыковочные привязки” на стр. 71.

Вызов служб Open SCA из модулей SCA

Приложения SCA, разработанные с помощью IBM Integration Designer, могут вызывать приложения Open SCA, разработанные в среде Rational Application Developer. В этом разделе приведен пример вызова службы Open SCA из модуля SCA с помощью привязки импорта SCA.

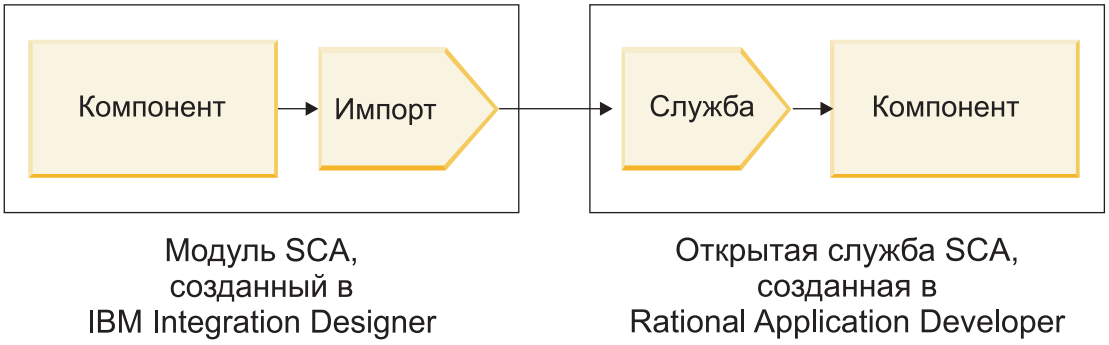


Рисунок 11. Компонент в модуле SCA, вызывающий службу Open SCA

Для вызова службы Open SCA не требуется никакой специальной настройки.

Для соединения со службой Open SCA при помощи привязки импорта SCA в привязке импорта указываются имя компонента и имя службы Open SCA.

1. Для получения имени целевого компонента и службы из комплекса Open SCA выполните следующее:
   a. Откройте вкладку **Свойства**, выбрав **Окно > Показать панель > Свойства**.
   b. Откройте редактор комплекса, дважды щелкнув на диаграмме комплекса, содержащей компонент и службу. Например, для компонента с именем **customer** схемой комплекса является **customer.диаграмма-комплекса**.
   c. Щелкните на целевом компоненте.
   d. В поле **Имя** на вкладке **Свойства** укажите имя целевого компонента.
   e. Щелкните на значке службы, связанной с компонентом.
   f. В поле **Имя** на вкладке **Свойства** укажите имя службы.
2. Для настройки импорта IBM Business Process Manager на соединение со службой Open SCA выполните следующее:
   a. В IBM Integration Designer перейдите на вкладку **Свойства** импорта SCA, используемого для соединения со службой Open SCA.
   b. В поле **Имя модуля** введите имя компонента из этапа 1d.
   c. В поле **Имя экспорта** введите имя службы из этапа 1f.
   d. Сохраните свою работу нажатием клавиш Ctrl+S.

Вызов модулей SCA из служб Open SCA

Приложения Open SCA, разработанные в среде Rational Application Developer, могут вызывать приложения SCA, разработанные с помощью IBM Integration Designer. В этом примере приведен пример вызова модуля SCA (при помощи привязки экспорта SCA) из службы Open SCA.

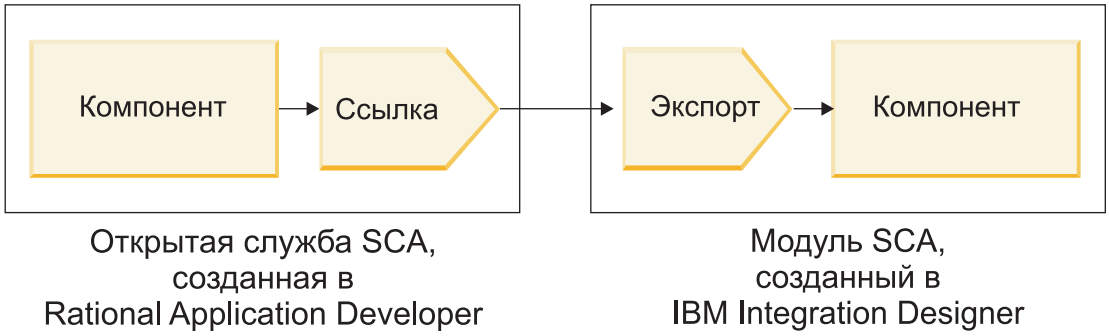


Рисунок 12. Откройте вызывающий службу SCA компонент в модуле SCA

Для соединения с компонентом SCA при помощи привязки ссылки Open SCA укажите имя модуля и имя экспорта.

1. Для получения имени целевого модуля и экспорта выполните следующее:
   a. В IBM Integration Designer откройте модуль в редакторе сборки, дважды щелкнув на модуле.
   b. Щелкните на экспорте.
   c. В поле **Имя** на вкладке **Свойства** укажите имя экспорта.
2. Настройте ссылку SCA, которая будет использоваться для соединения с модулем IBM Business Process Manager и экспортом:
   a. В Rational Application Developer Откройте редактор комплекса, дважды щелкнув на диаграмме комплекса, содержащей компонент и службу.
   b. Щелкните на значке ссылки, чтобы открыть свойства ссылки на вкладке **Свойства**.
   c. Откройте вкладку **Привязка** на левой стороне страницы.
   d. Щелкните на **Привязки** и затем на **Добавить**.
   e. Выберите привязку **SCA**.
   f. В поле **Uri** введите имя модуля IBM Business Process Manager, косую черту (“/”) и затем имя экспорта (которое было определено на этапе 1c).
   g. Нажмите кнопку **OK**.
   h. Сохраните свою работу нажатием клавиш Ctrl+S.

Вызов служб через стыковочные привязки

Следующие привязки поддерживаются для возможности взаимодействия со службой Open SCA.

- Привязка SCA

  В IBM Business Process Manager, когда модуль SCA вызывает службу Open SCA при помощи привязки импорта SCA, поддерживаются следующие стили вызовов:

  – Асинхронный (односторонний)
  – Синхронный (запрос/ответ)

  Интерфейс импорта SCA и интерфейс службы Open SCA должны использовать стыкуемость веб-служб (WS-I), совместимую с интерфейсом WSDL.

  Отметим, что привязка SCA поддерживает транзакцию и контекстное распространение защиты.
- Привязка веб-службы (JAX-WS) с протоколом SOAP1.1/HTTP или SOAP1.2/HTTP

  Интерфейс импорта SCA и интерфейс службы Open SCA должны использовать стыкуемость веб-служб (WS-I), совместимую с интерфейсом WSDL.

  Кроме того, поддерживаются следующие качества службы:

  – Атомарная транзакция веб-служб

  - Защита веб-служб
- Привязка EJB

  Когда используется привязка EJB, интерфейс Java используется для определения взаимодействия между модулем SCA и службой Open SCA.

  Отметим, что привязка EJB поддерживает транзакцию и контекстное распространение защиты.
- Привязки JMS

  Интерфейс импорта SCA и интерфейс службы Open SCA должны использовать стыкуемость веб-служб (WS-I), совместимую с интерфейсом WSDL.

  Поддерживаются следующие провайдеры JMS:
  - WebSphere Platform Messaging (привязка JMS)
  - WebSphere MQ (привязка MQ JMS)

Примечание: Business Graphs не задействуется ни через какие привязки SCA и поэтому не поддерживаются в интерфейсах, используемых для взаимодействия с WebSphere Application Server Feature Pack for Service Component Architecture.

Типы привязок

Вместе с объектами импорта и экспорта используются *привязки* определенных протоколов, определяющие способ передачи данных в модуль и из него.

Выбор типа связывания:

При создании приложения необходимо знать, как выбрать привязку, отвечающую требованиям приложения.

В IBM Integration Designer доступны привязки, предоставляющие различные возможности. Используя этот список, можно определить тип привязки, который в наибольшей степени отвечает требованиям приложения.

Привязку *Service Component Architecture (SCA)* рекомендуется выбирать при соблюдении следующих условий:

- Все службы содержатся в модулях, то есть внешние службы не используются.
- Требуется разделить функции по различным модулям SCA, которые взаимодействуют непосредственно друг с другом.
- Используются сильносвязанные модули

Привязку *Веб-служба* рекомендуется использовать, если выполняются следующие условия:

- Требуется обеспечить доступ к внешней службе или предоставить службу по Интернету.
- Используются слабосвязанные службы.
- Предпочтителен синхронный обмен данными, т.е. запрос от одной службы может ожидать ответа другой.
- Внешняя служба или предоставляемая служба использует протокол SOAP/HTTP или SOAP/JMS.

Связывание *HTTP* рекомендуется использовать, если выполняются следующие условия:

- Требуется общаться к внешней службе через Интернет или предоставлять службу через Интернет, и при этом используются другие веб-службы, такие как GET, PUT и DELETE.
- Используются слабосвязанные службы.
- Предпочтителен синхронный обмен данными, т.е. запрос от одной службы может ожидать ответа другой.

Привязку *Enterprise JavaBeans (EJB)* рекомендуется выбирать при соблюдении следующих условий:

- Связывание предназначено для импортированной службы EJB или службы, к которой осуществляют доступ клиенты EJB.
- Импортированная служба слабосвязанная.
- Взаимодействия EJB с сохранением состояния не требуются.

- Предпочтителен синхронный обмен данными, т.е. запрос от одной службы может ожидать ответа другой.

Привязку *Информационная система предприятия (EIS)* рекомендуется выбирать при соблюдении следующих условий:

- Требуется обеспечить доступ к службе в системе EIS с использованием адаптера ресурса.
- Синхронный обмен данными предпочтителен по сравнению с асинхронным.

Привязку *Служба обмена сообщениями Java (JMS)* рекомендуется выбирать при соблюдении следующих условий:

Важное замечание: Существует несколько типов связываний JMS. Если обмен сообщениями SOAP будет осуществляться с помощью JMS, воспользуйтесь привязкой веб-служб с протоколом SOAP/JMS. См. раздел “Привязки веб-служб” на стр. 74.

- Требуется обеспечить доступ к системе обмена сообщениями.
- Используются слабосвязанные службы.
- Асинхронный обмен данными предпочтителен по сравнению с синхронным.

Привязку *Базовая служба обмена сообщениями Java (JMS)* рекомендуется выбирать при соблюдении следующих условий:

- Требуется обеспечить доступ к системе обмена сообщениями, выпущенной не компанией IBM.
- Используются слабосвязанные службы.
- Надежность важнее, чем быстродействие: асинхронный обмен данными подходит лучше, чем синхронный.

Привязку *Очередь сообщений (MQ)* рекомендуется выбирать при соблюдении следующих условий:

- Требуется обеспечить доступ к системе обмена сообщениями WebSphere MQ, а также использование внутренних функций MQ.
- Используются слабосвязанные службы.
- Надежность важнее, чем быстродействие: асинхронный обмен данными подходит лучше, чем синхронный.

Связывание *MQ JMS* рекомендуется использовать, если выполняются следующие условия:

- Требуется обеспечить доступ к системе обмена сообщениями WebSphere MQ в контексте JMS; то есть набор функций JMS является достаточным для приложения.
- Используются слабосвязанные службы.
- Надежность важнее, чем быстродействие: асинхронный обмен данными подходит лучше, чем синхронный.

Привязки SCA:

Привязка архитектуры компонентов служб (SCA) позволяет службе взаимодействовать с другими службами в других модулях. Импорт с привязкой SCA позволяет обращаться к службе в другом модуле SCA. Экспорт с привязкой SCA обеспечивает доступ других модулей к данной службе.

IBM Integration Designer применяется для создания и настройки привязок SCA для импортов и экспортов в модулях SCA.

Если модули выполняются на одном и том же сервере или развернуты в одном и том же кластере, то привязка SCA является простейшей и скорейшей для использования.

После развертывания модуля с привязкой SCA на сервере административную консоль можно использовать для просмотра информации о привязке или, в случае привязки импорта, для изменения выбранных свойств привязки.

Привязки веб-служб:

Привязки веб-служб - это способ передачи сообщений из компонента архитектуры компонентов служб (SCA) в веб-службу и обратно.

Обзор привязок веб-служб: Advanced

Привязка импорта веб-службы позволяет вызывать внешнюю веб-службу из компонентов архитектуры компонентов служб (SCA). Привязка экспорта веб-службы позволяет делать компоненты SCA доступными клиентам в виде веб-служб.

С помощью привязки веб-службы можно обращаться к внешним службам посредством согласованных сообщений SOAP и QoS.

Integration Designer применяется для создания и настройки привязок веб-служб для импортов и экспортов в модулях SCA. Доступны следующие типы привязок веб-служб:

- SOAP1.2/HTTP и SOAP1.1/HTTP

  Эти типы привязок основаны на Java API for XML Web Services (JAX-WS), программном API на Java для создания веб-служб.

  – Используйте SOAP1.2/HTTP, если веб-служба соответствует спецификации SOAP 1.2.
  – Используйте SOAP1.1/HTTP, если веб-служба соответствует спецификации SOAP 1.1.

  Важное замечание: При развертывании приложения со связыванием веб-службы (JAX-WS) на целевом сервере необходимо выключить опцию **Запускать компоненты по необходимости**. Дополнительная информация приведена в разделе “Проверка конфигурации сервера” на стр. 82.

  Выбрав один из типов связывания, можно отправлять сообщения SOAP с вложениями.

  Связывание веб-службы работает со стандартными сообщениями SOAP. Тем не менее, различные типы связывания веб-службы (JAX-WS) позволяют настроить способ обработки или составления сообщений SOAP. Например, можно обрабатывать нестандартные элементы в сообщениях SOAP или применять дополнительную обработку для сообщения SOAP. В ходе настройки связывания указывается пользовательский обработчик данных, который выполняет такую обработку сообщений SOAP.

  В связывании веб-службы (JAX-WS) можно использовать наборы стратегий. Набор стратегий является набором типов стратегий, каждый из которых обеспечивает качество обслуживания (QoS). Например, набор стратегий WSAddressing обеспечивает независимую от транспортного протокола адресацию веб-служб и сообщений. Для выбора набора стратегий для связывания можно использовать Integration Designer.

  Примечание: Для использования набора стратегий на языке Security Assertion Markup Language (SAML) необходимо выполнить дополнительные действия по настройке, описанные в разделе “Импорт наборов стратегий SAML” на стр. 80.

- SOAP1.1/HTTP

  Эта привязка позволяет создавать веб-службы с сообщением в формате SOAP с использованием Java API for XML-based RPC (JAX-RPC).

- SOAP1.1/JMS

  Эта привязка позволяет отправлять и принимать сообщения SOAP с помощью адресатов службы сообщений Java (JMS).

Независимо от того, какой протокол (HTTP или JMS) используется для передачи сообщения SOAP, связывание веб-службы всегда обрабатывает взаимодействия запрос/ответ синхронно. Вызывающая поставщика службы нить блокируется до получения ответа от поставщика. Дополнительная информация приведена в разделе “Синхронный вызов”.

Важное замечание: Следующие сочетания связывания веб-служб не могут использоваться в экспортах в одном и том же модуле. Если требуется экспортировать компоненты с помощью более чем одной из этих привязок экспорта, их необходимо поместить в разные модули и затем связать эти модули с компонентами посредством связывания SCA:

- SOAP 1.1/JMS и SOAP 1.1/HTTP с использованием JAX-RPC
- SOAP 1.1/HTTP с использованием JAX-RPC и SOAP 1.1/HTTP с использованием JAX-WS
- SOAP 1.1/HTTP с использованием JAX-RPC и SOAP 1.2/HTTP с использованием JAX-WS

После развертывания на сервере модуля SCA, содержащего связывание веб-службы, в административной консоли можно просмотреть информацию о связывания или изменить свойства связывания.

Примечание: Веб-службы позволяют приложениям взаимодействовать посредством стандартных описаний служб и стандартных форматов обмена сообщениями. Например, привязки импорта и экспорта веб-службы могут взаимодействовать со службами, реализованными на основе web Services Enhancements (WSE) версии 3.5 и Windows Communication Foundation (WCF) версии 3.5 для Microsoft .NET. При взаимодействии с такими службами необходимо обеспечить следующее:

- Файл WSDL, используемый для доступа к экспорту веб-службы, должен включать непустое действие SOAP для каждой операции в интерфейсе.
- При отправке сообщений в экспорт веб-службы клиент веб-службы должен содержать или заголовок SOAPAction, или заголовок wsa:Action.

Распространение заголовка SOAP: Advanced

При обработке сообщений SOAP может потребоваться получить доступ к информации каких-либо заголовков SOAP полученных сообщений, передать сообщения с заголовками SOAP, которым присвоены заданные значения, или разрешить прохождение заголовков SOAP через модуль.

При настройке связывания веб-службы в Integration Designer можно указать, что заголовки SOAP должны передаваться.

- Когда информация приходят в экспорт или ответы - в импорт, становится доступным заголовок SOAP. При этом модуль может работать со значениями заголовка и изменять заголовок.
- Когда запросы отправлены из экспорта или ответы - из импорта, в соответствующие сообщения можно включать заголовки SOAP.

Формат и наличие передаваемых заголовков SOAP могут модифицироваться наборами стратегий, настроенными в импорте или экспорте. См. раздел Табл. 31 на стр. 76.

Для того чтобы настроить передачу заголовков SOAP для импорта или экспорта, на панели Свойства Integration Designer откройте вкладку **Передача заголовка протокола** и настройте требуемые параметры.

Заголовок WS-Addressing

Заголовок WS-Addressing может передаваться связыванием веб-службы (JAX-WS).

При передаче заголовка WS-Addressing необходимо учитывать следующее:

- Если передача заголовка WS-Addressing включена, то заголовок будет передаваться в модуль в следующих ситуациях:
  - Когда запросы получены в экспорте
  - Когда ответы получены в импорте
- Заголовок WS-Addressing header не передается в исходящие сообщения из IBM Business Process Manager, то есть он не передается когда запросы отправлены из импорта или ответы - из экспорта.

Заголовок WS-Security

Заголовок WS-Security может передаваться как связыванием веб-службы (JAX-WS), так и связыванием веб-службы (JAX-RPC).

Спецификация WS-Security веб-служб описывает расширения для обмена сообщениями SOAP, отвечающие за сохранение целостности сообщений, конфиденциальность и идентификацию для отдельных сообщений. Эти механизмы можно использовать для подключения ряда моделей защиты и технологий шифрования.

При передаче заголовка WS-Security необходимо учитывать следующее:

- Если передача заголовка WS-Security включена, то заголовок будет передаваться через модуль в следующих ситуациях:
  - Когда запросы получены в экспорте
  - Когда запросы отправлены из импорта
  - Когда ответы получены в импорте
- По умолчанию заголовок *не* будет передаваться, когда ответы отправлены из экспорта. Однако если свойство JVM **WSSECURITY.ECHO.ENABLED** задано равным **true**, то заголовок будет передаваться, когда ответы отправлены из экспорта. В этом случае, если заголовок WS-Security не изменяется в пути запроса, то он может автоматически перейти из запросов в ответы.
- Точный формат сообщения SOAP, отправленного из импорта для запроса или из экспорта для ответа, может не совпадать с полученным в исходном сообщении SOAP. Потому цифровая подпись должна считаться утратившей силу. Если цифровая подпись необходима для отправляемых сообщений, то ее необходимо создать с помощью соответствующих процедур стратегий защиты, а заголовки WS-Security, связанные с цифровой подписью в полученных сообщениях, должны быть удалены в модуле.

Для передачи заголовка WS-Security необходимо включить схему WS-Security в модуль приложения. Дополнительная информация о включении схемы приведена в разделе “Включение схемы WS-Security в модуль приложения” на стр. 77.

Как передаются заголовки

Способ передачи заголовков зависит от параметров стратегии защиты, настроенных для связывания импорта или экспорта, как показано в Табл. 31:

Таблица 31. Как передаются заголовки защиты

	Связывание экспорта в отсутствие стратегии защиты	**Связывание экспорта при наличии стратегии защиты**
Связывание импорта в отсутствие стратегии защиты	Заголовки защиты передаются без изменений через модуль. Они не расшифровываются. В исходящем сообщении эти заголовки точно соответствуют полученным. При этом цифровая подпись может быть нарушена.	Заголовки защиты расшифровываются и передаются через модуль с проверкой подписи и идентификацией. Расшифрованные заголовки отправляются как исходящие. При этом цифровая подпись может быть нарушена.
Связывание импорта при наличии стратегии защиты	Заголовки защиты передаются без изменений через модуль. Они не расшифровываются. Заголовки не должны передаваться с импорт. В противном случае возникает ошибка из-за дублирования.	Заголовки защиты расшифровываются и передаются через модуль с проверкой подписи и идентификацией. Заголовки не должны передаваться с импорт. В противном случае возникает ошибка из-за дублирования.

Настройте соответствующие наборы стратегий для связывания импорта и экспорта. Это изолирует клиента служб от изменений конфигурации или требований QoS поставщика служб. Если стандартные заголовки SOAP видимы в модуле, то это влияет на операции в модуле, например, на протокол и трассировку. Передача заголовков SOAP через модуль из полученного сообщения в отправленное означает, что преимущества изоляции теряются.

Стандартные заголовки, такие как заголовки WS-Security, не следует передавать в запросе в импорт или в ответе в экспорт, когда с импортом или экспортом связан набор стратегий, отвечающий за создание таких заголовков. В противном случае возникнет ошибка из-за дублирования заголовков. Вместо этого необходимо явно удалить эти заголовки или настроить связывание импорта или экспорта так, чтобы эти заголовки протокола не передавались.

Доступ к заголовкам SOAP

Когда сообщение, содержащее заголовки SOAP, приходит от импорта или экспорта веб-службы, заголовки помещаются в раздел заголовков объекта сообщения службы (SMO). Доступ к информации заголовка описан в разделе “Доступ к информации заголовка SOAP в SMO”.

Включение схемы WS-Security в модуль приложения

Ниже описана процедура включения схемы в модуль приложения:

- Если компьютер, на котором работает Integration Designer, подключен к Интернету, то выполните следующие действия:
  1. В проекции Бизнес-интеграция выберите **Зависимости** для своего проекта.
  2. Откройте раздел **Стандартные ресурсы** и выберите или **Файлы схемы WS-Security 1.0**, или **Файлы схемы WS-Security 1.1**, чтобы импортировать схему в модуль.
  3. Очистите и заново скомпонуйте проект.
- Если компьютер, на котором работает Integration Designer, не подключен к Интернету, то схему можно загрузить на другой компьютер, подключенный к Интернету. Затем можно будет скопировать схему на компьютер, на котором работает Integration Designer.
  1. Загрузите схему на компьютер, у которого есть подключение к Интернету:
     a. Выберите **Файл > Импорт > Бизнес-интеграция > WSDL и XSD**.
     b. Выберите **Удаленный WSDL** или **Файл XSD**.
     c. Импортируйте следующие схемы:

        http://www.w3.org/2003/05/soap-envelope/

        http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd

        http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd
  2. Скопируйте схемы на компьютер, у которого отсутствует подключение к Интернету.
  3. Импортируйте схемы на компьютер, у которого отсутствует подключение к Интернету.
     a. Выберите **Файл > Импорт > Бизнес-интеграция > WSDL и XSD**.
     b. Выберите **Локальный WSDL** или **Файл XSD**.
  4. Измените расположение схемы для oasis-wss-wssecurity_secext-1.1.xsd:
     a. Откройте схему из файла *расположение-рабочей-среды*/*имя-модуля*/StandardImportFilesGen/oasis-wss-wssecurity-secext-1.1.xsd.
     b. Измените:

        ```
        <xs:import namespace='http://www.w3.org/2003/05/soap-envelope'
        schemaLocation='http://www.w3.org/2003/05/soap-envelope/'/>
        ```

        на:

        ```
        <xs:import namespace='http://www.w3.org/2003/05/soap-envelope'
        schemaLocation='../w3/_2003/_05/soap_envelope.xsd'/>
        ```

     c. Измените:

```
<xs:import namespace='http://www.w3.org/2001/04/xmlenc#'
schemaLocation='http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd'/>
```

на:

```
<xs:import namespace='http://www.w3.org/2001/04/xmlenc#'
schemaLocation='../w3/tr/_2002/rec_xmlenc_core_20021210/xenc-schema.xsd'/>
```

5. Измените расположение схемы для oasis-200401-wss-wssecurity-secext-1.0.xsd:
   a. Откройте схему из файла *расположение-рабочей-среды*/*имя-модуля*/StandardImportFilesGen/oasis-200401-wss-wssecurity-secext-1.0.xsd.
   b. Измените:

```
<xsd:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
```

на:

```
<xsd:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="../w3/tr/_2002/rec_xmldsig_core_20020212/xmldsig-core-schema.xsd"/>
```

6. Очистите и заново скомпонуйте проект.

Передача заголовков транспортного протокола: Advanced

При обработке сообщений SOAP может потребоваться получить доступ к информации каких-либо транспортных заголовков полученных сообщений, передать сообщения с транспортными заголовками, которым присвоены заданные значения, или разрешить прохождение транспортных заголовков через модуль.

При настройке привязки веб-службы в Integration Designer можно указать, что транспортные заголовки должны передаваться.

- Когда запросы приходят в экспорт или ответы - в импорт, становится доступным заголовок транспортного протокола. При этом модуль может работать со значениями заголовка и изменять заголовок.
- Когда ответы отправляются из экспорта или запросы - из импорта, в соответствующие сообщения можно включать заголовки транспортного протокола.

Настройка передачи заголовков

Для того чтобы настроить передачу транспортных заголовков для импорта или экспорта, выполните следующие действия:

1. На панели Свойства Integration Designer выберите **Связывание > Передача**.
2. Настройте параметры передачи транспортных заголовков.

Примечание: По умолчанию передача транспортных заголовков выключена и может применяться только в среде выполнения версии не ниже 7.0.0.3. Обратите внимание, что в версии 7.0.0.3 передача транспортных заголовков возможна только для заголовков HTTP.

Если включена передача транспортных заголовков, то заголовки будут передаваться через модуль из полученных сообщений, и если их явно не удалить, то заголовки будут использоваться в последующих вызовах в той же нити.

Примечание: Передача транспортных заголовков невозможна при использовании связывания веб-службы (JAX-RPC).

Доступ к информации заголовка

Когда для принимаемых сообщений включена передача транспортных заголовков, все транспортные заголовки, включая пользовательские, доступны в объекте сообщения службы (SMO). Значения заголовков

можно изменить, также можно создать новые заголовки. Однако проверка значений не выполняется, и недопустимые или ошибочные заголовки могут привести к сбою веб-службы.

При настройке заголовков HTTP необходимо учитывать следующее:

- Все изменения заголовков, зарезервированных для веб-службы, не будут сохраняться в исходящем сообщении. Например, модуль веб-службы сам устанавливает значения заголовков HTTP version, method, Content-Type, Content-Length и SOAPAction.
- Если значение заголовка - это число, то необходимо непосредственно присвоить число (а не строку). Например, следует использовать **Max-Forwards = 5** (а не **Max-Forwards = Max-Forwards: 5**) и **Age = 300** (а не **Age = Age: 300**).
- Если размер сообщения-запроса не превышает 32 КБ, то веб-служба удаляет заголовок Transfer-Encoding и вместо него в заголовке Content-Length указывает фиксированный размер сообщения.
- WAS.channel.http сбрасывает значение Content-language в пути ответа.
- Ошибочное значение Upgrade приводит к ошибке с кодом 500.
- Следующие заголовки дописывают значение, зарезервированное модулем веб-службы, к пользовательским параметрам заголовка:
  - User-Agent
  - Cache-Control
  - Pragma
  - Accept
  - Connection

Получить данные заголовка можно одним из следующих способов:

- Используя примитив передачи для доступа к структурам SMO

  Дополнительная информация по работе с примитивами передачи приведена по ссылкам “Связанная информация”.
- Используя SPI службы контекста

  В следующем примере кода транспортные заголовки HTTP читаются из службы контекста:

```
HeadersType headerType = ContextService.INSTANCE.getHeaders();
HTTPHeaderType httpHeaderType = headerType.getHTTPHeader();
List HTTPHeader  httpHeaders = httpHeaderType.getHeader();
if(httpHeaders!=null){
 for(HTTPHeader httpHeader: httpHeaders){
   String httpHeadername = httpHeader.getName();
   String httpHeaderValue = httpHeader.getValue();
     }
}
List PropertyType  properties = headerType.getProperties();
if(properties!=null){
  for(PropertyType property: properties){
   String propertyName = property.getName();
   String propertyValue = property.getValue().toString();
     }
}
```

Устранение неполадок

При неполадках с отправкой новых заголовков можно перехватить сообщение TCP/IP такими инструментами, как TCP/IP Monitor в Integration Designer. Открыть TCP/IP Monitor можно, выбрав **Запуск/Отладка > TCP/IP Monitor** на странице Параметры.

Просмотреть значения заголовков можно также с помощью трассировки JAX-WS: **org.apache.axis2.*=all: com.ibm.ws.websvcs.*=all:**

Работа с привязками веб-служб (JAX-WS): Advanced

В привязки веб-служб (JAX-WS), применяемые в приложениях, можно добавить описание QoS на языке Security Assertion Markup Language (SAML). Для импорта набора стратегий сначала необходимо использовать административную консоль. Также административная консоль позволяет проверить, правильно ли настроен сервер для работы с привязкой веб-службы (JAX-WS).

Импорт наборов стратегий SAML: Advanced

Security Assertion Markup Language (SAML) - это основанный на XML стандарт OASIS для обмена данными из атрибутов пользовательских профайлов и защиты. Во время настройки привязки веб-службы (JAX-WS) в Integration Designer можно указать набор стратегий SAML. Сначала с помощью административной консоли IBM Business Process Manager открывается доступ к наборам стратегий SAML для того чтобы их можно было импортировать в Integration Designer.

Наборы стратегий SAML обычно располагаются в каталоге конфигурации профайла:

каталог-профайла/config/templates/PolicySets

Перед началом этой процедуры убедитесь в наличии следующих каталогов (которые содержат наборы стратегий) в каталоге конфигурации профайла:

- SAML11 Bearer WSHTTPS default
- SAML20 Bearer WSHTTPS default
- SAML11 Bearer WSSecurity default
- SAML20 Bearer WSSecurity default
- SAML11 HoK Public WSSecurity default
- SAML20 HoK Public WSSecurity default
- SAML11 HoK Symmetric WSSecurity default
- SAML20 HoK Symmetric WSSecurity default
- Имя-пользователя WSHTTPS default

Если каталоги не находятся в каталоге конфигурации профайла, скопируйте их в этот каталог из следующего расположения:

корневой-каталог-сервера-приложений/profileTemplates/default/documents/config/templates/PolicySets

Импортируйте наборы стратегий в административную консоль, выберите из них те, к которым требуется открыть доступ для Integration Designer, а затем сохраните файл .zip для каждого из этих наборов стратегий в расположении, доступном для Integration Designer.

1. Импортируйте наборы стратегий, выполнив следующие действия:
   a. В административной консоли выберите **Службы > Наборы стратегий > Наборы стратегий приложений**.
   b. Выберите действия **Импорт > Из хранилища по умолчанию**.
   c. Выберите наборы стратегий SAML по умолчанию и нажмите кнопку **OK**.
2. Экспортируйте наборы стратегий для того чтобы приложение Integration Designer смогло их использовать:
   a. На странице Наборы стратегий приложения выберите набор стратегий, который требуется импортировать, и нажмите кнопку **Экспортировать**.

      Примечание: Если страница Наборы стратегий приложений на данный момент не открыта, выберите **Службы > Наборы стратегий > Наборы стратегий приложений** в административной консоли.

b. На следующей странице щелкните на ссылке на файл .zip набора стратегий.
c. В окне Загрузка файла нажмите кнопку **Сохранить** и укажите расположение, доступное для Integration Designer.
d. Нажмите кнопку **Назад**.
e. Выполните действия с 2a на стр. 80 по 2d для каждого набора стратегий, который требуется экспортировать.

Наборы стратегий SAML сохраняются в файлах .zip и готовы к импорту в Integration Designer.

Импортируйте наборы стратегий в Integration Designer, как описано в разделе “Наборы стратегий”.

Вызов веб-служб, которые требуют простой идентификации HTTP: 

Простая идентификация HTTP использует имя пользователя и пароль для идентификации клиента службы на защищенной точке. Можно настроить простую идентификацию HTTP для отправки и получения запросов веб-служб.

Настройка простой идентификации HTTP для получения запросов веб-служб производится путем настройки привязки экспорта API Java для веб-служб XML (JAX-WS), как описано в разделе Создание и назначение ролей защиты спискам экспорта веб-служб.

Можно включить простую идентификацию HTTP для запросов веб-служб, отправляемых привязкой импорта JAX-WS, одним из двух способов:

- при настройке привязки импорта в модуле SCA можно выбрать существующий набор стратегий идентификации HTTP с именем BPMHTTPBasicAuthentication (который поставляется с привязкой импорта веб-службы (JAX-WS)) или любой другой набор стратегий, содержащий стратегию HTTPTransport;
- при конструировании модуля SCA можно использовать возможности потока передачи для динамического создания нового заголовка идентификации HTTP и указать в заголовке имя пользователя и пароль.

Примечание: Набор стратегий имеет приоритет перед значением, указанным в заголовке. Если требуется использовать значение из заголовка идентификации HTTP во время выполнения, не прикрепляйте набор стратегий, содержащий стратегию HTTPTransport. В особенности, не используйте стандартный набор стратегий BPMHTTPBasicAuthentication и, если набор стратегий определен, убедитесь, что в нем отсутствует стратегия HTTPTransport.

Дополнительные сведения о наборах стратегий веб-служб и привязках стратегий, а также об их использовании приведены в разделе Наборы стратегий веб-служб в справочной системе WebSphere Application Server Information Center.

- Для использования поставляемого набора стратегий выполните следующие действия:
  1. Необязательно: В административной консоли создайте общую привязку стратегии клиента или измените существующую привязку, которая содержит стратегию HTTPTransport с требуемым ИД пользователя и паролем.
  2. В IBM Integration Designer создайте привязку импорта веб-службы (JAX-WS) и прикрепите набор стратегий BPMHTTPBasicAuthentication.
  3. Выполните *одно* из следующих действий:
     – В IBM Integration Designer в свойствах привязки импорта веб-службы (JAX-WS) укажите имя существующей общей привязки стратегии клиента, которая содержит стратегию HTTPTransport.
     – После развертывания модуля SCA в административной консоли выберите существующую привязку стратегии клиента или создайте привязку и свяжите ее с привязкой импорта.
  4. Необязательно: В административной консоли сервера процессов измените выбранную привязку набора стратегий, указав требуемые ИД пользователя и пароль.

- Для того чтобы указать имя пользователя и пароль в заголовке идентификации HTTP выполните следующие действия:
  - с помощью примитива передачи Задать заголовок HTTP в IBM Integration Designer создайте заголовок идентификации HTTP и укажите имя пользователя и пароль;
  - если требуется дополнительная логика, используйте код Java в пользовательском примитиве передачи (как показано в следующем примере) для выполнения следующих действий:
    1. создания заголовка идентификации HTTP;
    2. ввода имени пользователя и пароля;
    3. добавления нового заголовка идентификации HTTP в HTTPControl;
    4. возврата обновленного HTTPControl в службу Контекст.

```
//Получить HeaderInfoType из contextService
ContextService contextService = (ContextService) ServiceManager.INSTANCE
.locateService("com/ibm/bpm/context/ContextService");
HeaderInfoType headers = contextService.getHeaderInfo();
if(headers == null){
    headers = ContextObjectFactory.eINSTANCE.createHeaderInfoType();
}
//Получить заголовок HTTP и элемент управления HTTP из HeaderInfoType
HTTPHeaderType httpHeaderType = headers.getHTTPHeader();
HTTPControl cp = httpHeaderType.getControl();
HeadersFactory factory = HeadersFactory.eINSTANCE;
if(cp == null){
    cp = factory.createHTTPControl();
}
//Создать HTTPAuthentication и задать HTTPCredentials
HTTPAuthentication authorization = factory.createHTTPAuthentication();
HTTPCredentials credentials = factory.createHTTPCredentials();
authorization.setAuthenticationType(HTTPAuthenticationType.BASIC_LITERAL);
credentials.setUserId("ИМЯ");
credentials.setPassword("ПАРОЛЬ");
authorization.setCredentials(credentials);
cp.setAuthentication(authorization);
httpHeaderType.setControl(cp);
//Возвратить информацию заголовка в текущий контекст выполнения.
contextService.setHeaderInfo(headers);
```

Проверка конфигурации сервера: Advanced

Во время развертывания приложения с привязкой веб-службы (JAX-WS) необходимо убедиться, что в конфигурации сервера, на котором развертывается приложение, не выбрана опция **Запускать компоненты по необходимости**.

Проверить, выбрана ли эта опция, можно путем выполнения следующих действий в административной консоли:

1. Выберите пункт **Серверы > Типы серверов > Серверы приложений WebSphere**.
2. Щелкните на имени сервера.
3. На вкладке Конфигурация определите, выбран ли параметр **Запускать компоненты по необходимости**.
4. Выполните одно из следующих действий:
   - Если параметр **Запускать компоненты по необходимости** выбран, отмените выбор и нажмите кнопку **Применить**.
   - Если параметр **Запускать компоненты по необходимости** на выбран, нажмите кнопку **Отмена**.

Вложения в сообщениях SOAP: Advanced

Можно отправлять и получать сообщения SOAP, включающие двоичные данные (такие как файлы PDF или изображения JPEG) в виде вложений. Вложения могут быть *ссылочными*, то есть явным образом

представленными как фрагменты сообщения в интерфейсе службы, или *нессылочными*, в которые может быть включено произвольное число вложений различных типов.

Ссылочное вложение представляется одним из следующих способов:

- Вложения MTOM используют кодировку, заданную механизмом MTOM протокола SOAP (http://www.w3.org/TR/soap12-mtom/). Для включения поддержки вложений MTOM необходимо настроить параметр привязок импорта и экспорта. Это рекомендуемый способ кодировки вложений в новых приложениях.
- Как элемент с типом wsi:swaRef в схеме сообщения

  Вложения, определенные с типом wsi:swaRef, соответствуют спецификации Web Services Interoperability Organization (WS-I) *Attachments Profile Version 1.0* (http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html), где описано, как связаны элементы сообщения и фрагменты MIME.
- Как главный фрагмент сообщения с использованием двоичного типа схемы

  Вложения, являющиеся главным фрагментом сообщения, соответствуют спецификации *сообщений SOAP с вложениями* (http://www.w3.org/TR/SOAP-attachments).

  Вложения, представленные как фрагмент сообщения верхнего уровня, можно настроить так, чтобы документ WSDL и сообщения, генерируемые привязкой, соответствовали спецификации WS-I *Attachments Profile Version 1.0* и WS-I *Basic Profile Version 1.1* (http://www.ws-i.org/Profiles/BasicProfile-1.1.html).

Нессылочное вложение в сообщении SOAP никак не представлено в схеме сообщения.

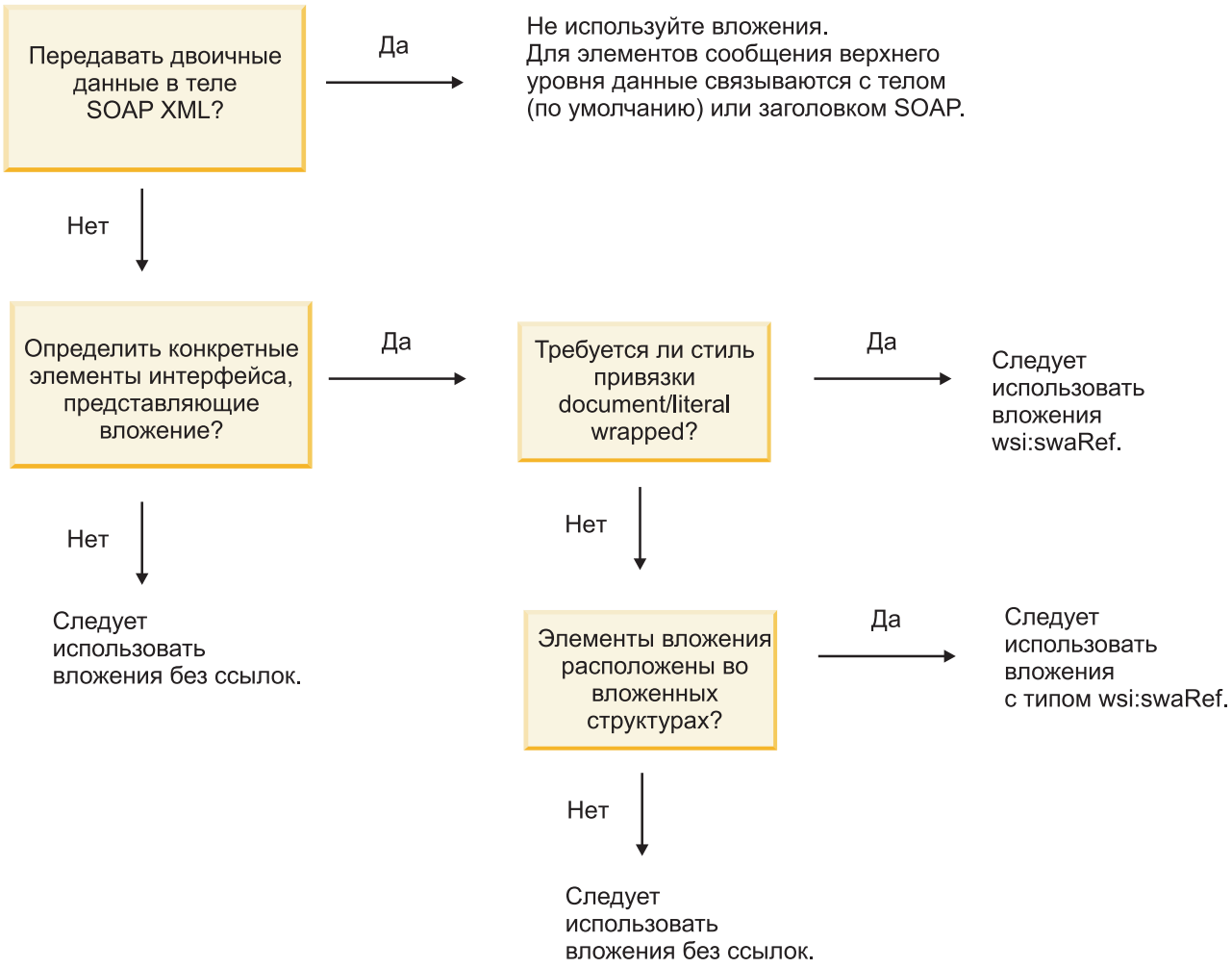
Во всех случаях, кроме вложений MTOM, привязка WSDL SOAP должна включать привязку MIME для вложений, а максимальный размер вложений не должен превышать 20 МБ..

Примечание: Для того чтобы отправлять сообщения SOAP с вложениями или получать их, необходимо использовать один из вариантов привязки веб-служб, основанный на Java API for XML Web Services (JAX-WS).

Способы выбора стиля соответствующего вложения: Advanced

При проектировании нового интерфейса службы, включающего двоичные данные, необходимо спланировать, как двоичные данные будут передаваться в сообщениях SOAP, отправляемых и получаемых службой.

Для вложений должен использоваться механизм MTOM, если он поддерживается подключенным приложением веб-службы. В противном случае используйте следующую диаграмму для выбора стиля вложения. Используйте следующий перечень вопросов, чтобы определить необходимый тип вложения:



Вложения MTOM: фрагменты сообщений верхнего уровня:

Поддерживается отправка и прием сообщений веб-служб с вложениями MTOM. В сообщении SOAP с несколькими фрагментами MIME тело SOAP является первым фрагментом сообщения, а вложение или вложения передаются в последующих фрагментах.

При отправке или приеме ссылочного вложения в сообщении SOAP двоичные данные вложения (часто большого размера) хранятся отдельно от тела сообщения SOAP, поэтому их не требуется обрабатывать как данные XML. При этом производительность обработки по сравнению с обработкой данных в элементах XML повышается.

Ниже приведен пример сообщения MTOM SOAP:

```
... другие транспортные заголовки ...
Content-Type: multipart/related; boundary=MIMEBoundaryurn_uuid_0FE43E4D025F0BF3DC11582467646812; type="application/xop+xml"
<0.urn:uuid:0FE43E4D025F0BF3DC11582467646813@apache.org>"; start-info="text/xml"; charset=UTF-8

--MIMEBoundaryurn_uuid_0FE43E4D025F0BF3DC11582467646812
content-type: application/xop+xml; charset=UTF-8; type="text/xml";
content-transfer-encoding: binary
content-id:
   <0.urn:uuid:0FE43E4D025F0BF3DC11582467646813@apache.org>

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
      <soapenv:Header/>
```

```
      <soapenv:Body>
         <sendImage xmlns="http://org/apache/axis2/jaxws/sample/mtom">
            <input>
            <imageData><xop:Include xmlns:xop="http://www.w3.org/2004/08/xop/include"
href="cid:1.urn:uuid:0FE43E4D025F0BF3DC11582467646811@apache.org"/></imageData>
            </input>
         </sendImage>
      </soapenv:Body>
   </soapenv:Envelope>
--MIMEBoundaryurn_uuid_0FE43E4D025F0BF3DC11582467646812
content-type: text/plain
content-transfer-encoding: binary
content-id:
         <1.urn:uuid:0FE43E4D025F0BF3DC11582467646811@apache.org>

... двоичные данные ...
--MIMEBoundaryurn_uuid_0FE43E4D025F0BF3DC11582467646812--
```

Обратите внимание, что в примере MTOM параметр content-type в конверте SOAP равен **application/xop+xml**, а двоичные данные заменены элементом **xop:Include** следующего вида:

```
<xop:Include xmlns:xop="http://www.w3.org/2004/08/xop/include"
href="cid:1.urn:uuid:0FE43E4D025F0BF3DC11582467646811@apache.org"/>
```

Обработка входящих ссылочных вложений

Когда клиент передает сообщение SOAP с вложением в компонент SCA, сначала привязка экспорта веб-службы (JAX-WS) удаляет вложение. Затем она обрабатывает часть SOAP сообщения и создает бизнес-объект. Наконец, привязка создает двоичное вложение в бизнес-объекте.

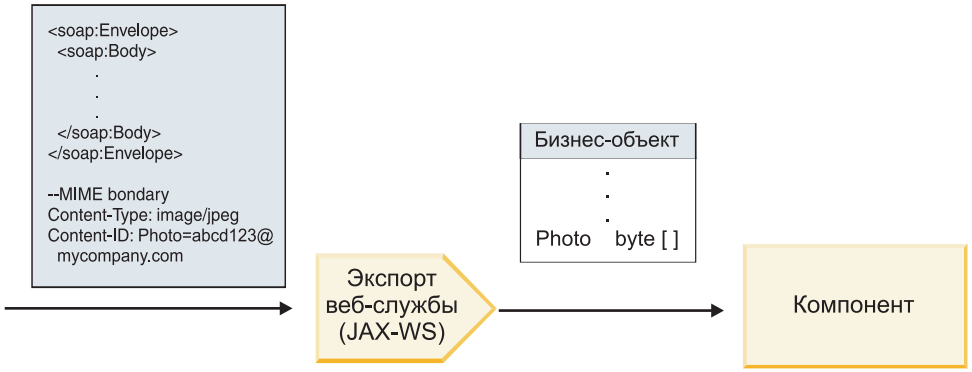


Рисунок 13. Каким образом привязка экспорта веб-службы (JAX-WS) обрабатывает сообщение SOAP со ссылочным вложением

Атрибуты вложения MTOM

- MTOM поддерживает элементы вложений во встроенных структурах.
- MTOM доступно только для типа base64Binary.
- MTOM поддерживает элементы вложений во встроенных структурах, то есть **bodyPath** вложений MTOM задает расположение **xpath** элемента, в котором хранится вложение MTOM. Логика вычисления **bodyPath** в точности следует схеме формирования расположения **xpath**, как показано в приведенных ниже примерах:
  - Для типа, отличного от массива (**maxOccurs** равно 1): /sendImage/input/imageData
  - Для массива (**maxOccurs** > 1): /sendImage/input/imageData[1]
- Смешанные типы вложений не поддерживаются, то есть если поддержка MTOM включена в привязке импорта, то вложение MTOM будет создано. Если поддержка MTOM выключена или в параметре настройки MTOM оставлено значение по умолчанию из привязки экспорта, то входящее сообщение MTOM не поддерживается.

Ссылочные вложения: элементы типа swaRef: Advanced

Можно отправлять и получать сообщения SOAP, включающие вложения, которые объявлены в интерфейсе службы как элементы типа swaRef.

Элемент типа swaRef определен в спецификации Web Services Interoperability Organization (WS-I) *Attachments Profile* Version 1.0 (http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html) и указывает, как элементы сообщений связаны с фрагментами MIME.

Тело сообщения SOAP содержит элемент типа swaRef, который задает ИД содержимого вложения.

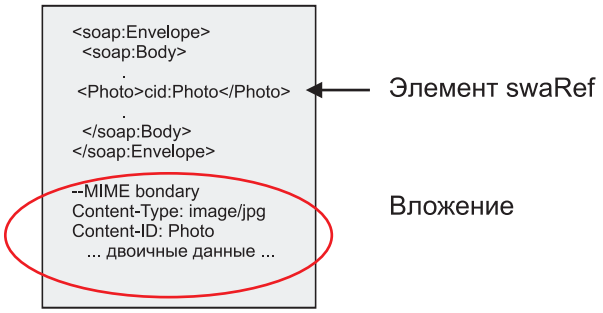


Рисунок 14. Сообщение SOAP с элементом swaRef

WSDL для этого сообщения SOAP содержит элемент с типом swaRef внутри фрагмента сообщения, определяющий вложение.

```
<element name="sendPhoto">
  <complexType>
     <sequence>
       <element name="Photo" type="wsi:swaRef"/>
     </sequence>
  </complexType>
 </element>
```

WSDL также должен включать связывание MIME, которое указывает, что используются мультифрагментные сообщения MIME.

Примечание: WSDL *не* включает связывание MIME для отдельных элементов с типом swaRef, поскольку связывание MIME применяется только к фрагменту сообщения верхнего уровня.

Вложения, представленные как элементы с типом swaRef, могут передаваться только по компонентам потока передачи. Если вложение должно быть доступно компоненту другого тип, или если его требуется передать в другой тип компонента, то используйте компонент потока передачи для перемещения вложения в расположение, доступное такому компоненту.

Обработка входящих вложений

Для настройки связывания экспорта можно использовать Integration Designer для приема сообщения. Сначала создается модуль, его интерфейс и операции, включая элемент типа swaRef. Затем создается связывание веб-службы (JAX-WS).

Примечание: Дополнительная информация приведена в разделе “Работа с вложениями” приведена в разделе Integration Designer Information Center.

Когда клиент передает сообщение SOAP с вложением типа swaRef в компонент SCA, сначала привязка экспорта веб-службы (JAX-WS) удаляет вложение. Затем оно обрабатывает часть SOAP сообщения и создает

бизнес-объект. Наконец, связывание присваивает значение content-ID вложения в бизнес-объекте.

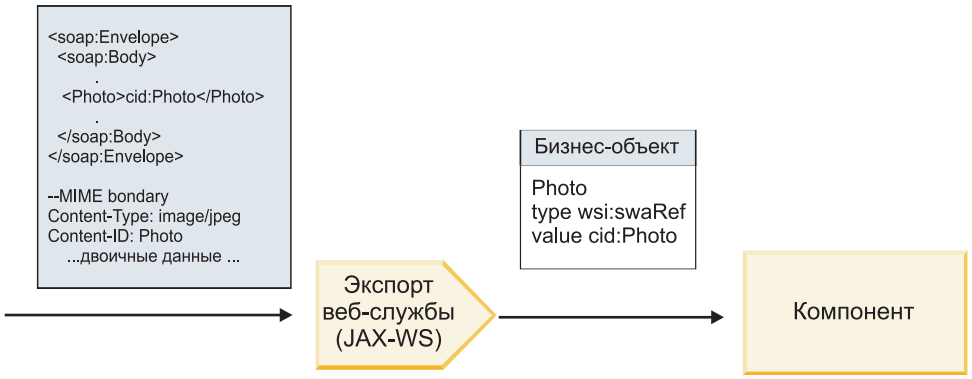


Рисунок 15. Каким образом привязка экспорта веб-службы (JAX-WS) обрабатывает сообщение SOAP с вложением типа swaRef

Доступ к мета-данным вложения в компоненте потока передачи

Как показано на рисунке рис. 16, когда компоненты обращаются к вложениям типа swaRef, идентификатор содержимого вложения представляется как элемент типа swaRef.

Каждое вложение в сообщении SOAP имеет соответствующий элемент **attachments** в SMO. При использовании типа swaRef WS-I элемент **attachments** включает тип содержимого и content-ID вложения, а также сами двоичные данные вложения.

Поэтому для того чтобы получить значение вложения swaRef, необходимо получить значение элемента с типом swaRef и затем найти элемент **attachments** с соответствующим значением **contentID**. Обратите внимание, что в значении **contentID** префикс **cid:**, как право, удаляется из значения swaRef.

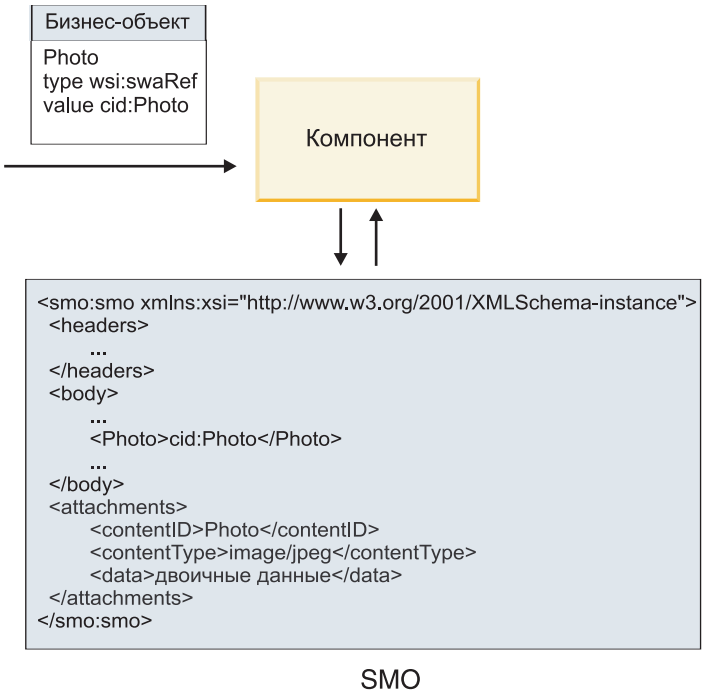


Рисунок 16. Как вложения типа swaRef представлены в SMO

Обработка исходящих

Integration Designer позволяет настроить привязку импорта веб-службы (JAX-WS) для вызова внешней веб-службы. Привязка импорта настраивается с документом WSDL, который описывает вызываемую веб-службу и определяет, какое вложение будет передаваться в веб-службу.

Когда сообщение SCA принимается привязкой веб-службы (JAX-WS), элементы с типом swaRef отправляются как вложения, если импорт связан с компонентом потока передачи и элемент типа swaRef имеет соответствующий элемент **attachments**.

В исходящих операциях элемент с типом swaRef всегда отправляются с их значениями content-ID, однако модуль передачи должен обеспечить наличие соответствующего элемента **attachments** с совпадающим значением **contentID**.

Примечание: Для соответствия спецификации WS-I Attachments Profile значение **content ID** должно следовать правилам для "content-id part encoding", описанным в разделе 3.8 спецификации WS-I *Attachments Profile* 1.0.

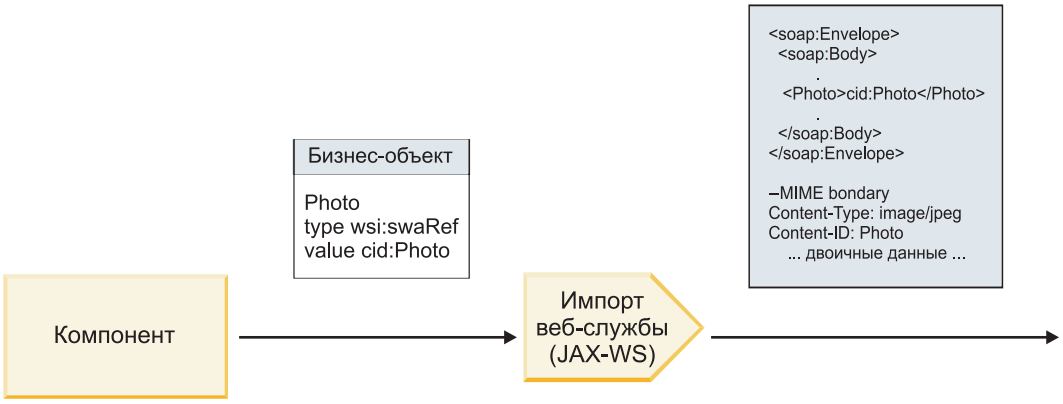


Рисунок 17. Каким образом привязка импорта веб-службы (JAX-WS) создает сообщение SOAP с вложением типа swaRef

Настройка мета-данных вложения в компоненте потока передачи

Если в SMO есть элемент с типом swaRef и элемент **attachments**, то связывание создает сообщение SOAP с вложением и отправляет его получателю.

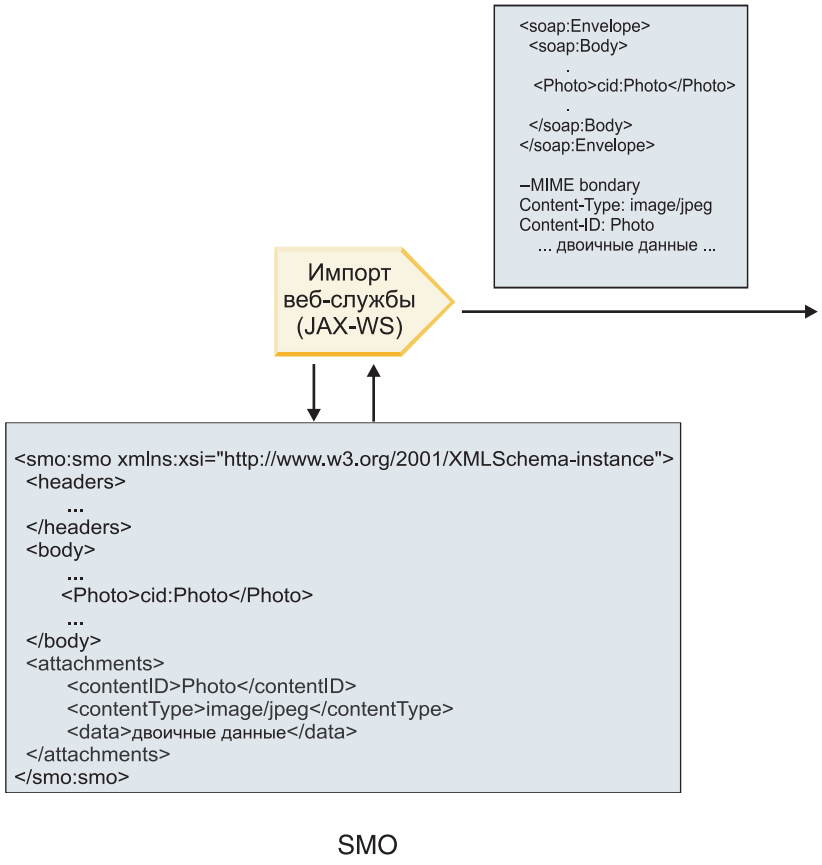


Рисунок 18. Как осуществляется доступ к вложению типа swaRef в SMO для создания сообщения SOAP

Элемент **attachments** присутствует в SMO только тогда, когда компонент потока передачи прямо соединен с импортом или экспортом. Он не передается через другие типы компонентов. Если такие значения требуется передать в модуль, содержащий другие типы компонентов, то необходимо использовать компонент потока передачи для копирования значений в расположение, где они будут доступны модулю, затем использовать другой компонент потока передачи для настройки нужных значений перед вызовом исходящей операции посредством импорта веб-службы.

Важное замечание: Как описано в разделе “Представление SMO в формате XML” примитив передачи Mapping преобразует сообщения посредством преобразования XSLT 1.0. Это преобразование работает с сериализованным SMO в формате XML. Примитив передачи Mapping позволяет задавать корневой элемент сериализации, который соответствует корневому элементу документа XML.

При отправке сообщений SOAP с вложениями выбранный корневой элемент определяет, как передаются сообщения.

- Если корневой элемент в карте XML - это “/body”, то по умолчанию все вложения передаются по карте.
- Если корневой элемент в карте - это “/”, то передачей вложений можно управлять.

Ссылочные вложения: фрагменты сообщений верхнего уровня: Advanced

Можно отправлять и получать сообщения SOAP, включающие двоичные вложения, которые объявлены как фрагменты в интерфейсе службы.

В сообщении SOAP с несколькими фрагментами MIME тело SOAP является первым фрагментом сообщения, а вложение или вложения передаются в последующих фрагментах.

В чем преимущество отправки или получения ссылочного вложения в сообщении SOAP? Двоичные данные вложения (часто большого размера) хранятся отдельно от тела сообщения SOAP, поэтому их не требуется обрабатывать как данные XML. При этом производительность обработки по сравнению с обработкой данных в элементах XML повышается.

Типы сообщений SOAP с ссылочными вложениями

Начиная с IBM Business Process Manager версии 7.0.0.3 пользователь может выбрать способ, каким генерируется сообщение SOAP:

- **Сообщения, соответствующие спецификации WS-I**

  Среда выполнения может создавать сообщения SOAP, которые соответствуют спецификации WS-I *Attachments Profile Version 1.0* и WS-I *Basic Profile Version 1.1*. В сообщении SOAP, совместимом с этими профилями, только один фрагмент сообщения связан с телом SOAP. Если фрагмент сообщения связан как вложение, то элемент content-id (как описано в спецификации WS-I *Attachments Profile Version 1.0*) используется для сопоставления вложения с фрагментом сообщения.
- **Сообщения, не соответствующие спецификации WS-I**

  Среда выполнения может создавать сообщения SOAP, которые не соответствуют спецификации профилей WS-I, но совместимы с сообщениями, созданными в версиях 7.0 или 7.0.0.2 продукта IBM Business Process Manager. Сообщения SOAP используют элементы верхнего уровня с именем, совпадающим с именем фрагмента сообщения с атрибутом **href**, содержащим **content-id** вложения, но кодировка фрагмента content-id не используется (как описано в спецификации WS-I *Attachments Profile Version 1.0*).

Настройка совместимости экспорта веб-службы с WS-I

Для настройки связывания экспорта можно использовать Integration Designer. Сначала создается модуль, его интерфейс и операции. Затем создается связывание веб-службы (JAX-WS). На странице Ссылочные вложения показаны все двоичные фрагменты из созданной операции, из которых можно выбрать вложения. Затем на странице Настроить совместимость с WS-I AP 1.0 Integration Designer выберите один из вариантов:

- **Использовать сообщение SOAP, соответствующее спецификации WS-I AP 1.0**

  Если выбрана эта опция, то также можно указать фрагмент сообщения, связанный с телом SOAP.

  Примечание: Эту опцию можно использовать, только если соответствующий файл WSDL также совместим с WS-I.

  Файл WSDL, создаваемый в Integration Designer 7.0.0.3, совместим с WS-I. Однако при импорте файла WSDL, несовместимого с WS-I, эта опция недоступна.
- **Использовать сообщение SOAP, не соответствующее спецификации WS-I AP 1.0**

  Если выбрана эта опция (значение по умолчанию), то первый фрагмент сообщения связан с телом SOAP.

Примечание: Только фрагменты сообщения верхнего уровня, то есть элементы, определенные в portType WSDL как части входящего или исходящего сообщения, имеющие двоичный тип (base64Binary или hexBinary), могут отправляться или приниматься как ссылочные вложения.
Дополнительная информация приведена в разделе “Работа с вложениями” в Integration Designer Information Center.

Для сообщений, совместимых с WS-I, элемент content-ID, генерируемый для сообщения SOAP, составляется из следующих элементов:

- Значение атрибута **name** элемента **wsdl:part**, на который ссылается **mime:content**
- Символ **=**
- Глобально уникальное значение, такое как UUID
- Символ **@**
- Допустимое имя домена

Обработка входящих ссылочных вложений

Когда клиент передает сообщение SOAP с вложением в компонент SCA, сначала связывание экспорта веб-службы (JAX-WS) удаляет вложение. Затем оно обрабатывает часть SOAP сообщения и создает бизнес-объект. Наконец, связывание создает двоичное вложение в бизнес-объекте.

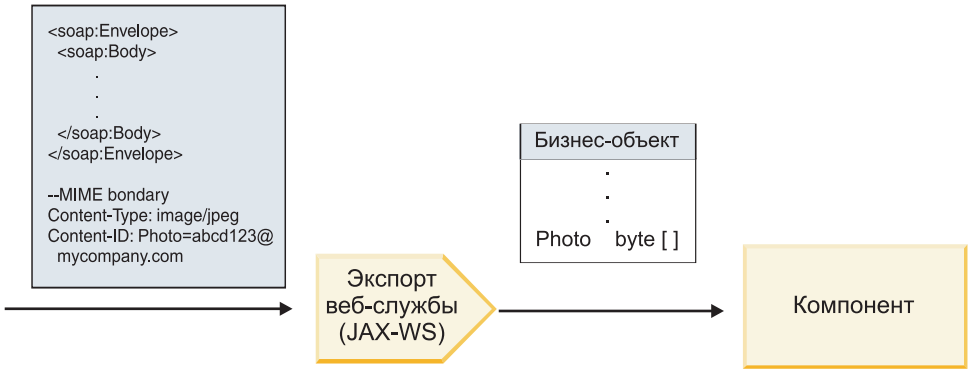


Рисунок 19. Как связывание экспорта веб-службы (JAX-WS) обрабатывает сообщение SOAP, совместимое с WS-I, с ссылочным вложением

Доступ к мета-данным вложения в компоненте потока передачи

Как показано на рисунке рис. 19, когда компоненты обращаются к ссылочным вложениям, данные вложения представляются как байтовый массив.

Каждое ссылочное вложение в сообщении SOAP имеет соответствующий элемент **attachments** в SMO. Элемент **attachments** включает тип содержимого вложения и путь к элементу тела сообщения, где хранится вложение.

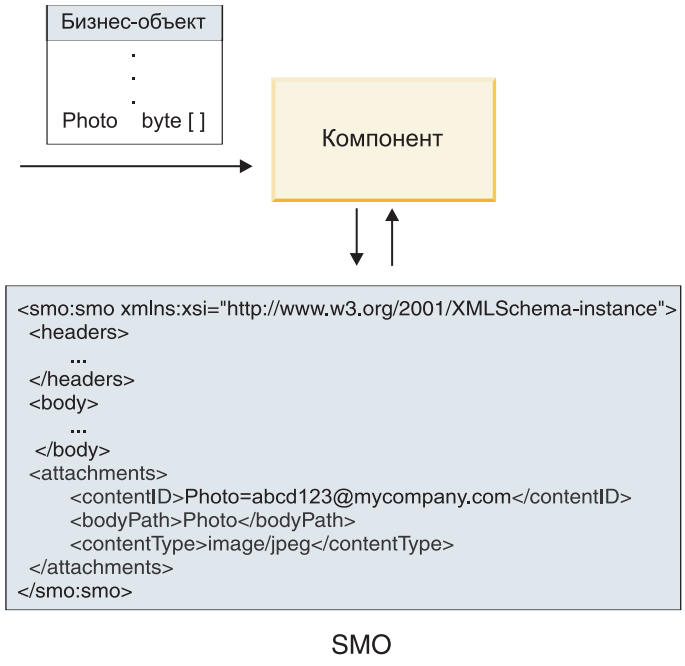


Рисунок 20. Как ссылочные вложения представлены в SMO

Важное замечание: Путь к телу сообщения не обновляется автоматически, если сообщение преобразуется и вложение перемещается. Для обновления пути в элементе **attachments**, например, в ходе преобразования или с помощью отдельного объекта составления сообщения, можно использовать поток передачи.

Как составляются исходящие сообщения SOAP

Integration Designer позволяет настроить связывание импорта веб-службы (JAX-WS) для вызова внешней веб-службы. Связывание импорта настраивается с документом WSDL, который описывает вызываемую веб-службу и определяет, какие фрагменты сообщения будут передаваться как вложения. Затем на странице Настроить совместимость с WS-I AP 1.0 Integration Designer можно выбрать один из вариантов:

- **Использовать сообщение SOAP, соответствующее спецификации WS-I AP 1.0**

  Если выбрана эта опция, то также можно указать фрагмент сообщения, связанный с телом SOAP. Все прочие фрагменты связываются с вложениями или заголовками. Сообщения, отправляемые связыванием, не включают в тело SOAP элементы, ссылающиеся на вложения. Это отношение выражается посредством content-ID вложения, включающего имя фрагмента сообщения.
- **Использовать сообщение SOAP, не соответствующее спецификации WS-I AP 1.0**

  Если выбрана эта опция (значение по умолчанию), то первый фрагмент сообщения связан с телом SOAP. Все прочие фрагменты связываются с вложениями или заголовками. Сообщения, отправляемые связыванием, включают в тело SOAP один или несколько элементов, ссылающихся на вложения посредством атрибута **href**.

Примечание: Компонент, представляющий вложение, согласно определению WSDL, должен иметь простой тип (base64Binary или hexBinary). Компонент составного типа не может быть связан как вложение.

Обработка исходящих ссылочных вложений

Привязка импорта использует информацию SMO для определения способа отправки двоичных фрагментов сообщения верхнего уровня в виде вложений.

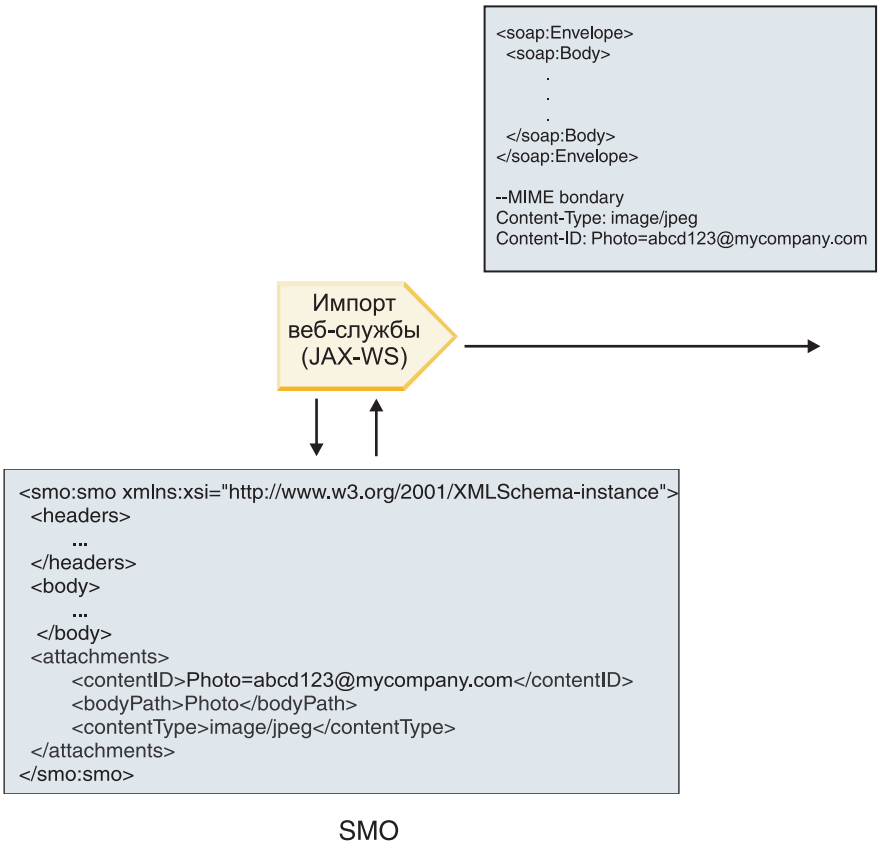


Рисунок 21. Как осуществляется доступ к ссылочному вложению в SMO для создания сообщения SOAP

Элемент **attachments** присутствует в SMO только тогда, когда компонент потока передачи прямо соединен с импортом или экспортом. Он не передается через другие типы компонентов. Если такие значения требуется передать в модуль, содержащий другие типы компонентов, то необходимо использовать компонент потока передачи для копирования значений в расположение, где они будут доступны модулю, затем использовать другой компонент потока передачи для настройки нужных значений перед вызовом исходящей операции посредством импорта веб-службы.

Для определения способа отправки сообщения в связывании анализируются следующие условия:

- Есть ли связывание MIME WSDL для двоичного фрагмента сообщения верхнего уровня, и если да, то как определен тип содержимого
- Есть ли элемент **attachments** в SMO, у которого значение **bodyPath** ссылается на двоичный фрагмент верхнего уровня

Как составляются вложения, если элемент attachment существует в SMO

В следующей таблице показано, как составляется и отправляется вложение, если SMO содержит элемент **attachment** с **bodyPath**, соответствующим части с именем сообщения:

Таблица 32. Как генерируется вложение

Состояние связывания MIME WSDL для главного двоичного фрагмента сообщения	**Как составляется и отправляется сообщение**
Выполняется одно из следующих условий: • Не определен тип содержимого для фрагмента сообщения • Определено несколько типов содержимого • Определен тип содержимого с символом подстановки	Фрагмент сообщения отправляется как вложение. Для **Content-Id** используется значение в элементе attachments, если оно задано, в противном случае оно генерируется. Для **Content-Type** используется значение в элементе attachments, если оно задано, в противном случае - **application/octet-stream**.
Присутствует, фрагмент сообщения содержит одну часть без символов подстановки	Фрагмент сообщения отправляется как вложение. Для **Content-Id** используется значение в элементе attachments, если оно задано, в противном случае оно генерируется. Для **Content-Type** используется значение в элементе attachments, если оно задано, в противном случае используется тип, определенный в элементе содержимого MIME WSDL.
Отсутствует	Фрагмент сообщения отправляется как вложение. Для **Content-Id** используется значение в элементе attachments, если оно задано, в противном случае оно генерируется. Для **Content-Type** используется значение в элементе attachments, если оно задано, в противном случае - **application/octet-stream**. **Примечание:** Отправка фрагментов сообщения в виде вложений, не определенных как таковые в WSDL, может нарушить соответствие спецификации WS-I Attachments Profile 1.0, и это следует избегать по мере возможности.

Как составляются вложения, если элемент attachment не существует в SMO

В следующей таблице показано, как составляется и отправляется вложение, если SMO не содержит элемент **attachment** с **bodyPath**, соответствующим части с именем сообщения:

Таблица 33. Как генерируется вложение

Состояние связывания MIME WSDL для главного двоичного фрагмента сообщения	**Как составляется и отправляется сообщение**
Выполняется одно из следующих условий: • Не определен тип содержимого для фрагмента сообщения • Определено несколько типов содержимого • Определен тип содержимого с символом подстановки	Фрагмент сообщения отправляется как вложение. Генерируется **Content-Id**. Для **Content-Type** присваивается значение **application/octet-stream**.
Присутствует, фрагмент сообщения содержит одну часть без символов подстановки	Фрагмент сообщения отправляется как вложение. Генерируется **Content-Id**. Для **Content-Type** присваивается тип, заданный в элементе содержимого MIME WSDL.
Отсутствует	Фрагмент сообщения не отправляется как вложение.

Важное замечание: Как описано в разделе “Представление SMO в формате XML” примитив передачи Mapping преобразует сообщения посредством преобразования XSLT 1.0. Это преобразование работает с сериализованным SMO в формате XML. Примитив передачи Mapping позволяет задавать корневой элемент сериализации, который соответствует корневому элементу документа XML.

При отправке сообщений SOAP с вложениями выбранный корневой элемент определяет, как передаются сообщения.

- Если корневой элемент в карте XML - это “/body”, то по умолчанию все вложения передаются по карте.
- Если корневой элемент в карте - это “/”, то передачей вложений можно управлять.

Нессылочные вложения: Advanced

Можно отправлять и получать *нессылочные вложения*, то есть вложения, не объявленные в интерфейсе службы.

В сообщении SOAP с несколькими фрагментами MIME тело SOAP является первым фрагментом сообщения, а вложения передаются в последующих фрагментах. В тело SOAP не включается ссылка на вложение.

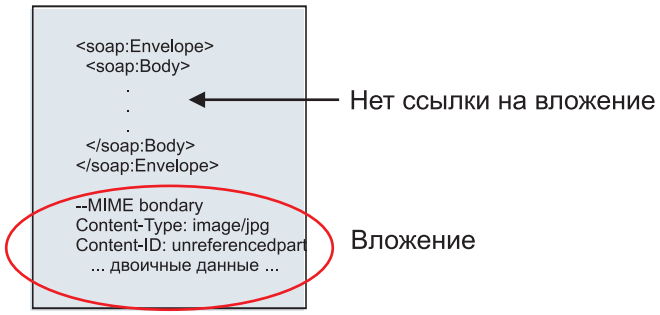


Рисунок 22. Сообщение SOAP с нессылочным вложением

Сообщение SOAP с нессылочным вложением можно отправить через экспорт веб-службы в импорт веб-службы. Исходящее сообщение, отправляемое в целевую веб-службу, содержит вложение.

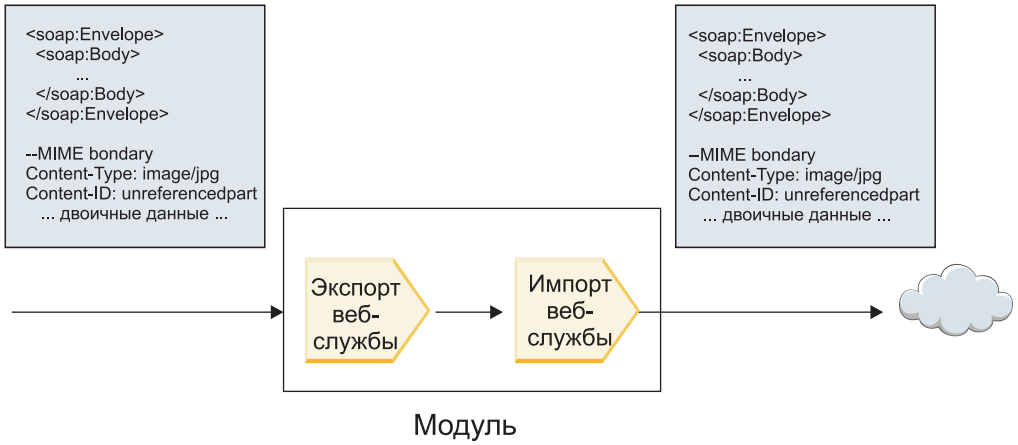


Рисунок 23. Прохождение вложения через модуль SCA

В рис. 23 сообщение SOAP, содержащее вложение, проходит без изменений.

Изменить сообщение SOAP можно с помощью компонента потока передачи. Так, компонент потока передачи может извлечь данные из сообщения SOAP (двоичные данные в теле сообщения) и создать

сообщение SOAP с вложением. Данные обрабатываются как часть элемента вложения объекта сообщения службы (SMO).

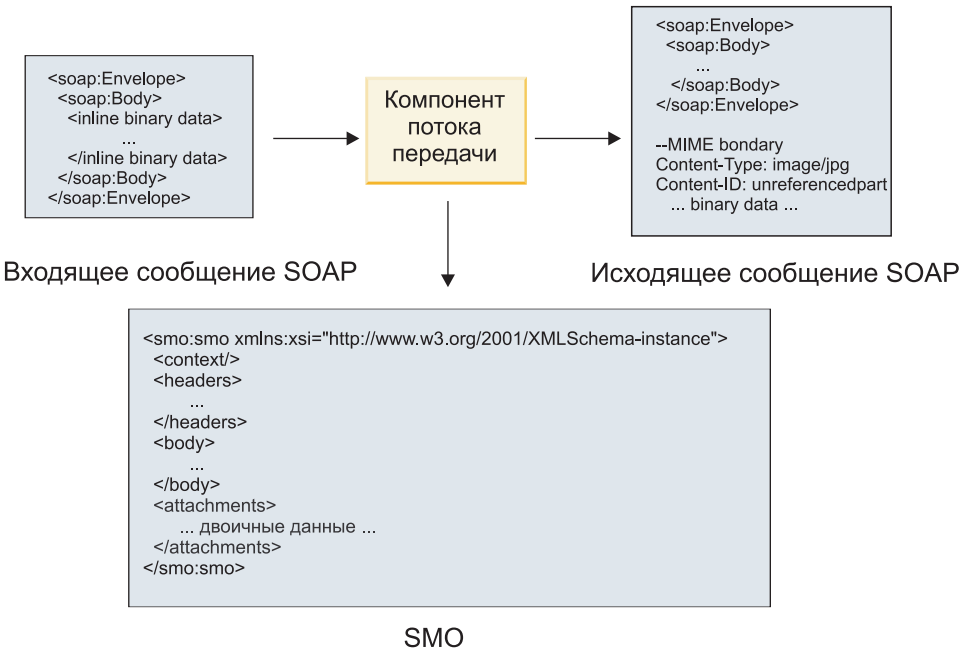


Рисунок 24. Сообщение, обработанное компонентом потока передачи

Компонент потока передачи также может преобразовывать входящее сообщение, извлекая из него вложение, кодируя его и передавая сообщение без вложений.

Вместо того чтобы извлекать данные из входящего сообщения SOAP, данные вложения можно получить из внешнего источника, например, из базы данных, и создать сообщение SOAP с вложениями.

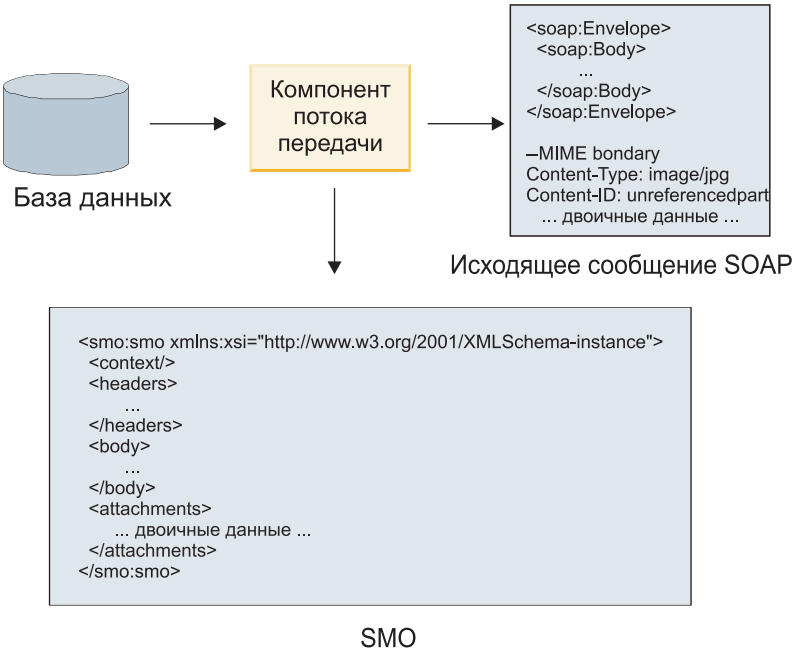


Рисунок 25. Вложение, полученное из базы данных и добавленное в сообщение SOAP

Напротив, компонент потока передачи может извлечь вложение из входящего сообщения SOAP и обработать сообщение, например, сохранив его в базе данных.

Нессылочные вложения могут передаваться только по компонентам потока передачи. Если вложение должно быть доступно компоненту другого тип, или если его требуется передать в другой тип компонента, то используйте компонент потока передачи для перемещения вложения в расположение, доступное такому компоненту.

Важное замечание: Как описано в разделе “Представление SMO в формате XML” примитив передачи Mapping преобразует сообщения посредством преобразования XSLT 1.0. Это преобразование работает с сериализованным SMO в формате XML. Примитив передачи Mapping позволяет задавать корневой элемент сериализации, который соответствует корневому элементу документа XML.

При отправке сообщений SOAP с вложениями выбранный корневой элемент определяет, как передаются сообщения.

- Если корневой элемент в карте XML - это “/body”, то по умолчанию все вложения передаются по карте.
- Если корневой элемент в карте - это “/”, то передачей вложений можно управлять.

Использование связывания стиля документа WSDL с многофрагментными сообщениями: Advanced

Организация Web Services Interoperability Organization (WS-I) определила правила для описания веб-служб посредством WSDL и способы формирования соответствующих сообщений SOAP, обеспечивающие возможность взаимодействия.

Эти правила описаны в спецификации WS-I *Basic Profile Version 1.1* (http://www.ws-i.org/Profiles/BasicProfile-1.1.html). В спецификации WS-I Basic Profile 1.1 R2712 говорится: "Связывание document-literal НЕОБХОДИМО сериализовать как ENVELOPE с soap:Body, дочерний элемент которого является экземпляром объявления глобального элемента, на который ссылается соответствующий фрагмент wsdl:message".

Это означает, что при использовании стиля документа для связывания SOAP в операциях с сообщениями (ввод, вывод или ошибка), для которых определено несколько фрагментов, только один из этих фрагментов будет связан с телом SOAP для обеспечения совместимости со спецификацией WS-I Basic Profile 1.1.

Далее в спецификации WS-I Attachments Profile 1.0 R2941 говорится: "wsdl:binding в DESCRIPTION ДОЛЖЕН связывать каждый из элементов wsdl:part в wsdl:message из wsdl:portType, к которому он относится, с одним из элементов soapbind:body, soapbind:header, soapbind:fault, soapbind:headerfault или mime:content".

Это означает, что при использовании стиля document для связывания SOAP в операциях с сообщениями (ввод, вывод или ошибка), для которых определено несколько фрагментов, все фрагменты, за исключением связанного с телом SOAP, должны быть связаны как вложения или заголовки.

Следующая процедура применяется, когда описания WSDL создаются для экспортов со связыванием веб-службы (JAX-WS и JAX-RPC):

- Если в сообщении есть несколько элементов с типом, отличным от двоичного, то можно выбрать, какой фрагмент сообщения будет связан с телом SOAP. Если есть только один элемент с типом, отличным от двоичного, то он автоматически будет связан с телом SOAP.
- Для связывания JAX-WS все прочие фрагменты сообщения с типом "hexBinary" или "base64Binary" связываются как ссылочные вложения. См. раздел “Ссылочные вложения: фрагменты сообщений верхнего уровня” на стр. 89.
- Все прочие фрагменты сообщения связываются как заголовки SOAP.

Связывание импорта JAX-RPC и JAX-WS учитывает связывание SOAP в существующем документе WSDL с сообщениями в стиле multipart-document, даже если эти множественные фрагменты не связаны с телом SOAP. Однако создавать клиентов веб-службы для таких документов WSDL в Rational Application Developer будет невозможно.

Примечание: Связывание JAX-RPC не поддерживает вложения.

Поэтому рекомендуется следующая процедура для работы с многофрагментными сообщениями в операциях со связыванием SOAP стиля document:

1. Используйте оболочечный стиль document/literal. В этом случае сообщение всегда будет иметь один фрагмент, однако вложения будут нессылочными (как описано в разделе “Нессылочные вложения” на стр. 95) или иметь тип swaRef (как описано в разделе “Ссылочные вложения: элементы типа swaRef” на стр. 86).
2. Используйте стиль RPC/literal. В этом случае связывание WSDL не налагает никаких ограничений на число фрагментов, связанных с телом SOAP. Итоговое сообщение SOAP будет иметь один дочерний элемент, который представляет вызываемую операцию, а фрагменты сообщения будут дочерними по отношению к этому элементу.
3. Для связывания JAX-WS создавайте не более одного фрагмента сообщения с типом, отличающимся от "hexBinary" или "base64Binary", в противном случае остальные недвоичные фрагменты будут связываться с заголовками SOAP.
4. Все прочие случаи подпадают под описанное поведение.

Примечание: При использовании сообщений SOAP, не соответствующих спецификации WS-I *Basic Profile Version 1.1*, существуют дополнительные ограничения.

- Первый фрагмент сообщения не должен быть двоичным.
- При приеме сообщений SOAP в стиле multipart-document со ссылочными вложениями связывание JAX-WS ожидает, что каждое ссылочное вложение будет представлено дочерним элементом тела SOAP с атрибутом href, который идентифицирует вложение по его content-ID. Связывание JAX-WS отправляет ссылочные вложения для таких сообщений аналогичным образом. Такое поведение не соответствует спецификации WS-I Basic Profile.

  Для того чтобы сообщения соответствовали спецификации Basic Profile, следуйте процедуре 1 или 2 из предыдущего списка или вместо ссылочных вложений используйте в таких сообщениях нессылочные вложения или вложения с типом swaRef.

Привязки HTTP:

Привязка HTTP предназначена для поддержки соединения SCA с HTTP. Она дает возможность встроить существующие и новые приложения HTTP в среду SCA.

Протокол HTTP широко используется для передачи информации в Интернете. Для работы с внешним приложением, которое использует протокол HTTP, необходима привязка HTTP. Привязка HTTP преобразует данные, полученные в сообщении во внутреннем формате, в бизнес-объект в приложении SCA. Также привязка HTTP преобразует данные, возвращенные в виде бизнес-объекта, во внутренний формат, который поддерживается внешним приложением.

Примечание: Для взаимодействия с клиентами и службами, использующими протокол SOAP/HTTP веб-служб, рекомендуется использовать одну из привязок веб-служб, которая предоставляет более широкий набор возможностей для обеспечения стандартного качества обслуживания веб-службы.

Ниже описаны некоторые стандартные сценарии использования привязки HTTP:

- Службы SCA могут вызывать приложения HTTP, используя объект импорта HTTP.
- Службы SCA могут экспортировать себя в виде приложений HTTP, которые могут применяться клиентами HTTP, с помощью объектов экспорта HTTP.

- IBM Business Process Manager и Process Server могут взаимодействовать друг с другом, используя инфраструктуру HTTP, поэтому пользователи могут управлять этим взаимодействием в соответствии с корпоративными стандартами.
- IBM Business Process Manager и Process Server могут выполнять роль посредников в обмене данными по протоколу HTTP, которые преобразуют и маршрутизируют сообщения для улучшения интеграции приложений.
- IBM Business Process Manager и Process Server могут выполнять функции моста между протоколом HTTP и другими протоколами, такими как веб-службы SOAP/HTTP, адаптеры ресурсов Java Connector Architecture (JCA), JMS и т. п.

Подробная информация о создании привязок импорта и экспорта HTTP приведена в справочной системе Integration Designer Information Center. Обратитесь к разделам в категории **Разработка приложений интеграции > Доступ к внешним службам по HTTP>**.

Обзор привязок HTTP:

Привязка HTTP обеспечивает связь с приложениями HTTP. Она передает данные между приложениями HTTP и позволяет вызывать приложения HTTP в модуле.

Привязки импорта HTTP

Привязка импорта HTTP обеспечивает передачу исходящих данных из приложений SCA на сервер или в приложения HTTP.

Объект импорта вызывает URL конечной точки HTTP. URL может быть указан одним из трех способов:
- URL может настраиваться динамически в заголовках HTTP с помощью динамически переопределяемого URL.
- URL может настраиваться динамически в элементе целевого адреса SMO.
- URL может быть задан в свойстве конфигурации объекта импорта.

Такой вызов по сути всегда является синхронным.

Хотя вызов HTTP всегда выполняется по схеме запрос-ответ, объект импорта HTTP поддерживает как однонаправленные, так и двунаправленные операции, игнорируя ответ в случае однонаправленной операции.

Привязки экспорта HTTP

Привязка экспорта HTTP обеспечивает передачу входящих данных из приложений HTTP в приложение SCA.

URL задается в объекте экспорта HTTP. Если приложению HTTP необходимо отправить сообщение с запросом в объект экспорта, то они вызывают объект экспорта, используя этот URL.

Объект экспорта HTTP также поддерживает вызовы ping.

Привязки HTTP во время выполнения

Во время выполнения объект импорта с привязкой HTTP отправляет запрос, который может и не содержать данных в теле сообщения, из приложения SCA во внешнюю веб-службу. Приложение SCA отправляет запрос внешней веб-службе, как показано на рисунке рис. 26 на стр. 100.

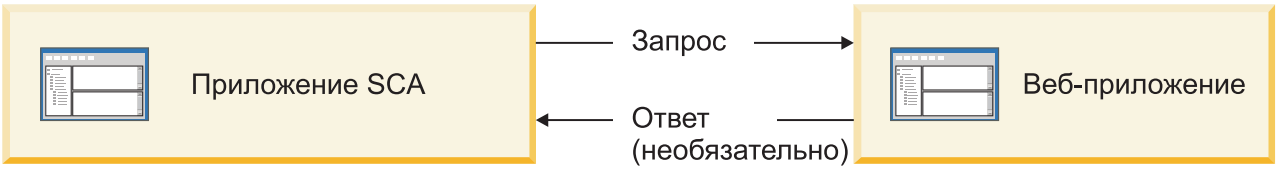


Рисунок 26. Передача запроса из приложения SCA в веб-приложение

При необходимости объект импорта с привязкой HTTP получает данные, возвращенные веб-приложением в ответ на запрос.

В случае экспорта приложение-клиент запрашивает веб-службу, как показано на рисунке рис. 27.

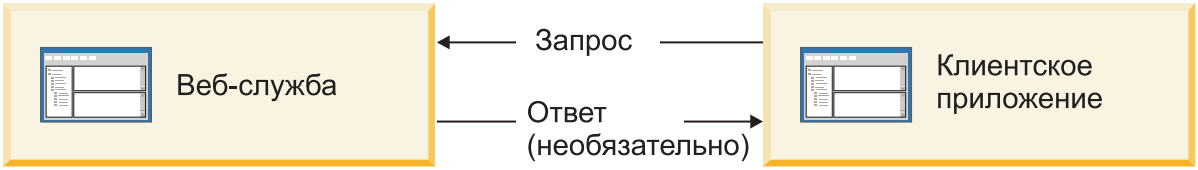


Рисунок 27. Передача запроса от клиентского приложения в веб-службу.

Веб-служба - это веб-приложение, работающее на сервере. Объект экспорта реализуется в веб-приложении в виде сервлета, поэтому клиент отправляет запрос на URL-адрес. Сервлет передает запрос в приложение SCA в среде выполнения.

При необходимости объект экспорта отправляет в приложение-клиент данные в ответ на запрос.

Заголовки HTTP:

В привязках импорта и экспорта HTTP можно настраивать заголовки HTTP и их значения, применяемые в исходящих сообщениях. В объектах импорта HTTP эти заголовки применяются для запросов, а в объектах экспорта HTTP - для ответов.

Статические заголовки и управляющая информация имеют больший приоритет, чем значения, которые динамически настраиваются во время выполнения. Однако значения метода, версии и URL для динамического переопределения используются вместо статических значений, которые во всех остальных случаях применяются по умолчанию.

Привязка поддерживает динамическую настройку URL импорта HTTP, определяя значение метода, версии и целевого URL HTTP во время выполнения. Для определения этих значений извлекается значение ссылки на конечную точку, URL для динамического переопределения, версии и метода.

- Для настройки ссылку на конечную точку используйте API com.ibm.websphere.sca.addressing.EndpointReference или укажите поле /headers/SMOHeader/Target/address в заголовке SMO.
- Для настройки URL для динамического переопределения, версии и метода используйте раздел управляющих параметров HTTP в сообщении SCA. Обратите внимание, что URL для динамического переопределения имеет более высокий приоритет, чем ссылка на целевую конечную точку, однако ссылка на конечную точку не зависит от привязки, поэтому рекомендуется использовать именно ее.

Привязки экспорта и импорта HTTP обрабатывают управляющую информацию и заголовки исходящих сообщений в следующем порядке:

1. Заголовок и управляющая информация, включая URL для динамического переопределения, версию и метод из сообщения SCA (наименее приоритетные)
2. Изменения, внесенные на уровне объекта импорта или экспорта в административной консоли

3. Изменения, внесенные на уровне метода объекта импорта или экспорта в административной консоли
4. Целевой адрес, заданный в ссылке на конечную точку или заголовке SMO
5. URL для динамического переопределения, версия и метод из сообщения SCA
6. Заголовки и управляющая информация из обработчика данных или привязки данных (наиболее приоритетные)

Объекты импорта и экспорта HTTP заполняют заголовки и управляющие параметры при приеме данных, используя содержимое входящего сообщения (HTTPExportRequest или HTTPImportResponse), если параметр передачи заголовков протокола равен **True**. И наоборот, объекты импорта и экспорта HTTP считывают и обрабатывают заголовки и управляющие параметры в исходящих данных (HTTPExportResponse или HTTPImportRequest), если параметр передачи заголовков протокола равен **True**.

Примечание: Изменения заголовков и управляющих параметров в ответах импорта и запросах экспорта с помощью обработчиков данных или привязок данных не изменяют порядок обработки сообщения в привязках импорта или экспорта и могут использоваться только для передачи измененных значений в нижележащие компоненты SCA.

Служба контекста отвечает за передачу информации о контексте (в том числе заголовков протокола, таких как заголовок HTTP, и контекста пользователя, такого как ИД учетной записи) вместе с путем для вызова SCA. Во время разработки в IBM Integration Designer для управления передачей контекста можно использовать свойства объектов импорта и экспорта. Более подробные сведения приведены в описании привязок импорта и экспорта в справочной системе IBM Integration Designer Information Center.

Структуры заголовков HTTP и их поддержка

Табл. 34 содержит перечень параметров для запросов и ответов объектов импорта и экспорта HTTP.

Таблица 34. Предоставляемая информация заголовка HTTP

Имя управляющего параметра	**Запрос импорта HTTP**	**Ответ импорта HTTP**	**Запрос экспорта HTTP**	**Ответ экспорта HTTP**
URL	Игнорируется	Не задано	Считывается из сообщения с запросом. **Примечание:** Строка запроса также является частью управляющего параметра URL.	Игнорируется
Версия (возможные значения: 1.0, 1.1; значение по умолчанию - 1.1)	Игнорируется	Не задано	Считывается из сообщения с запросом	Игнорируется
Метод	Игнорируется	Не задано	Считывается из сообщения с запросом	Игнорируется

Таблица 34. Предоставляемая информация заголовка HTTP (продолжение)

Имя управляющего параметра	Запрос импорта HTTP	Ответ импорта HTTP	Запрос экспорта HTTP	Ответ экспорта HTTP
URL для динамического переопределения	Если задан в обработчике данных или привязке данных, то переопределяет URL импорта HTTP. Сохраняется в сообщении в строке запроса. **Примечание:** Строка запроса также является частью управляющего параметра URL.	Не задано	Не задано	Игнорируется
Версия для динамического переопределения	Если задан, то переопределяет версию импорта HTTP. Сохраняется в сообщении в строке запроса.	Не задано	Не задано	Игнорируется
Метод для динамического переопределения	Если задан, то переопределяет метод импорта HTTP. Сохраняется в сообщении в строке запроса.	Не задано	Не задано	Игнорируется
Тип среды передачи (этот управляющий параметр содержит фрагмент значения заголовка HTTP Content-Type)	Если задан, то сохраняется в сообщении в заголовке Content-Type. **Примечание:** Это значение управляющего элемента должно предоставляться обработчиком данных или привязкой данных.	Считывается из заголовка Content-Type сообщения с ответом	Считывается из заголовка Content-Type сообщения с запросом	Если задан, то сохраняется в сообщении в заголовке Content-Type. **Примечание:** Это значение управляющего элемента должно предоставляться обработчиком данных или привязкой данных.
Набор символов (по умолчанию: UTF-8)	Если задан, то сохраняется в сообщении в заголовке Content-Type. **Примечание:** Это значение управляющего элемента должно предоставляться привязкой данных.	Считывается из заголовка Content-Type сообщения с ответом	Считывается из заголовка Content-Type сообщения с запросом	Поддерживается; сохраняется в сообщении в заголовке Content-Type. **Примечание:** Это значение управляющего элемента должно предоставляться привязкой данных.

Таблица 34. Предоставляемая информация заголовка HTTP (продолжение)

Имя управляющего параметра	Запрос импорта HTTP	Ответ импорта HTTP	Запрос экспорта HTTP	Ответ экспорта HTTP
Кодировка передачи (возможные значения: chunked, identity; значение по умолчанию - identity)	Если задан, то сохраняется в сообщении в виде заголовка и определяет кодировку при преобразовании сообщения.	Считывается из сообщения с ответом	Считывается из сообщения с запросом	Если задан, то сохраняется в сообщении в виде заголовка и определяет кодировку при преобразовании сообщения.
Кодировка содержимого (возможные значения: gzip, x-gzip, deflate, identity; значение по умолчанию - identity)	Если задан, то сохраняется в сообщении в виде заголовка и определяет кодировку данных.	Считывается из сообщения с ответом	Считывается из сообщения с запросом	Если задан, то сохраняется в сообщении в виде заголовка и определяет кодировку данных.
Длина содержимого	Игнорируется	Считывается из сообщения с ответом	Считывается из сообщения с запросом	Игнорируется
Код состояния (по умолчанию: 200)	Не поддерживается	Считывается из сообщения с ответом	Не поддерживается	Если задан, то сохраняется в сообщении в строке ответа
Причина (по умолчанию: OK)	Не поддерживается	Считывается из сообщения с ответом	Не поддерживается	Это управляющее значение игнорируется. Значение в строке ответа генерируется с учетом кода состояния.
Идентификация (содержит несколько свойств)	Если задан, то используется для создания заголовка простой идентификации. **Примечание:** Значение этого заголовка кодируется только в протоколе HTTP. В SCA оно раскодируется и передается в виде обычного текста.	Неприменимо	Считывается из заголовка Basic Authentication сообщения с запросом. Наличие этого заголовка не говорит о том, что пользователь был идентифицирован. Для управления идентификацией должна применяться конфигурация сервлета. **Примечание:** Значение этого заголовка кодируется только в протоколе HTTP. В SCA оно раскодируется и передается в виде обычного текста.	Неприменимо
Proxy (содержит несколько свойств: Хост, Порт, Идентификация)	Если задан, то используется для установления соединения через Proxy.	Неприменимо	Неприменимо	Неприменимо

Таблица 34. Предоставляемая информация заголовка HTTP (продолжение)

Имя управляющего параметра	Запрос импорта HTTP	Ответ импорта HTTP	Запрос экспорта HTTP	Ответ экспорта HTTP
SSL (содержит несколько свойств: Хранилище ключей, Пароль хранилища ключей, Хранилище доверенных сертификатов, Пароль хранилища доверенных сертификатов, Идентификация клиента)	Если параметр задан, и целевой URL использует HTTPS, то параметр применяется для установления соединения SSL.	Неприменимо	Неприменимо	Неприменимо

Привязки данных HTTP:

Для каждой схемы преобразования сообщений SCA в сообщения протокола HTTP необходимо настроить обработчик данных или привязку данных HTTP. Обработчики данных предоставляют интерфейс, не зависящий от типа привязки, который может использоваться в любых привязках транспортных протоколов, и поэтому рекомендуется использовать именно их. Привязки данных предназначены для определенной привязки транспортного протокола. В составе продукта предусмотрены классы привязок данных для протокола HTTP, и их можно дополнить собственными обработчиками или привязками данных.

Примечание: Три описанных здесь класса привязок данных HTTP (HTTPStreamDataBindingSOAP, HTTPStreamDataBindingXML и HTTPServiceGatewayDataBinding) устарели и больше не используются в IBM Business Process Manager версии 7.0. Вместо описанных в этом разделе привязок данных рекомендуется использовать следующие обработчики данных:

- SOAPDataHandler вместо HTTPStreamDataBindingSOAP.
- UTF8XMLDataHandler вместо HTTPStreamDataBindingXML
- GatewayTextDataHandler вместо HTTPServiceGatewayDataBinding

Для использования в объектах импорта и экспорта HTTP предусмотрены следующие привязки данных: привязка двоичных данных, привязка данных XML и привязка данных SOAP. Привязка данных ответа не требуется для однонаправленных операций. Для представления привязки данных используется имя класса Java, экземпляры которого могут выполнять преобразование данных HTTP в ServiceDataObject и обратное преобразование. Для определения необходимой привязки данных и выполняемой операции в объекте экспорта должен применяться селектор функций в сочетании с привязками методов. Предоставляются следующие привязки данных:

- Привязки двоичных данных, рассматривающие тело как неструктурированные двоичные данные. Ниже приведена реализация схемы XSD привязки двоичных данных:

```
<xsd:schema elementFormDefault="qualified"
  targetNamespace="http://com.ibm.websphere.http.data.bindings/schema"
  xmlns:tns="http://com.ibm.websphere.http.data.bindings/schema"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:complexType name="HTTPBaseBody">
    <xsd:sequence/>
  </xsd:complexType>

  <xsd:complexType name="HTTPBytesBody">
    <xsd:complexContent>
      <xsd:extension base="tns:HTTPBaseBody">
        <xsd:sequence>
          <xsd:element name="value" type="xsd:hexBinary"/>
```

```
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```

- Привязки данных XML, которые поддерживают тело, содержащее данные XML. Реализация привязки данных XML аналогична реализации привязки данных JMS XML и не накладывает никакие ограничения на схему интерфейса.
- Привязки данных SOAP, поддерживающие тело, содержащее данные SOAP. Реализация привязки данных SOAP не накладывает никакие ограничения на схему интерфейса.

Реализация пользовательских привязок данных HTTP

В этом разделе приведена информация о том, как реализовать пользовательскую привязку данных HTTP.

Примечание: Рекомендуется реализовать пользовательский обработчик данных вместо привязки, поскольку его можно использовать в разных привязках транспортных протоколов.

HTTPStreamDataBinding - это интерфейс для обработки пользовательских сообщений HTTP. Этот интерфейс разработан для поддержки обработки данных большого размера. Однако для его работы необходимо, чтобы привязка данных возвращала управляющую информацию и заголовки перед записью сообщения в поток.

Пользовательская привязка данных должна реализовывать следующие методы в указанном порядке.

Для настройки привязки данных создайте класс, реализующий HTTPStreamDataBinding. Привязка данных должна содержать четыре частных свойства:

- private DataObject pDataObject
- private HTTPControl pCtrl
- private HTTPHeaders pHeaders
- private yourNativeDataType nativeData

Привязка HTTP будет вызывать методы привязки данных в следующем порядке:

- Обработка исходящих данных (преобразование DataObject во внутренний формат):
  1. setDataObject(...)
  2. setHeaders(...)
  3. setControlParameters(...)
  4. setBusinessException(...)
  5. convertToNativeData()
  6. getControlParameters()
  7. getHeaders()
  8. write(...)
- Обработка входящих данных (преобразование из внутреннего формата в DataObject):
  1. setControlParameters(...)
  2. setHeaders(...)
  3. convertFromNativeData(...)
  4. isBusinessException()
  5. getDataObject()
  6. getControlParameters()
  7. getHeaders()

Метод setDataObject(...) должен вызываться в методе convertFromNativeData(...) для настройки значения dataObject, которое необходимо преобразовать из внутреннего формата в содержимое частного свойства "pDataObject".

```
public void setDataObject(DataObject dataObject)
   throws DataBindingException {
  pDataObject = dataObject;

}
public void setControlParameters(HTTPControl arg0) {
  this.pCtrl = arg0;
}

public void setHeaders(HTTPHeaders arg0) {
  this.pHeaders = arg0;
}
/*
* Добавить заголовок http "IsBusinessException" в pHeaders.
* Выполняется в два шага:
* 1. Удалить все заголовки с именем IsBusinessException (без учета регистра).
*    Это гарантирует наличие только одного заголовка.
* 2. Добавить новый заголовок "IsBusinessException"
*/
public void setBusinessException(boolean isBusinessException) {
  //удалить все заголовки с именем IsBusinessException (в любом регистре)
  //это обеспечит наличие только одного заголовка
  //добавить новый заголовок "IsBusinessException", например:
  HTTPHeader header=HeadersFactory.eINSTANCE.createHTTPHeader();
  header.setName("IsBusinessException");
  header.setValue(Boolean.toString(isBusinessException));
  this.pHeaders.getHeader().add(header);

}
public HTTPControl getControlParameters() {
  return pCtrl;
}
public HTTPHeaders getHeaders() {
  return pHeaders;
}
public DataObject getDataObject() throws DataBindingException {
  return pDataObject;
}
/*
* Получить заголовок "IsBusinessException" из pHeaders, вернуть его булевское значение
*/
public boolean isBusinessException() {
  String headerValue = getHeaderValue(pHeaders,"IsBusinessException");
  boolean result=Boolean.parseBoolean(headerValue);
  return result;
}
public void convertToNativeData() throws DataBindingException {
  DataObject dataObject = getDataObject();
  this.nativeData=realConvertWorkFromSDOToNativeData(dataObject);
}
public void convertFromNativeData(HTTPInputStream arg0){
  //Пользовательский метод для
  //чтения данных из HTTPInputStream
  //Преобразовать их в DataObject
  DataObject dataobject=realConvertWorkFromNativeDataToSDO(arg0);
  setDataObject(dataobject);
}
public void write(HTTPOutputStream output) throws IOException {
  if (nativeData != null)
  output.write(nativeData);
}
```

Привязки EJB:

Привязки импортов Enterprise JavaBeans (EJB) позволяют компонентам архитектуры компонентов служб (SCA) вызывать службы, предоставляемые бизнес-логикой Java EE, работающей на сервере Java EE. Привязки экспортов EJB позволяют компонентам SCA экспортироваться как EJB, чтобы бизнес-логика Java EE могла вызывать компоненты SCA, которые в противном случае недоступны ей.

Привязки импортов EJB:

Привязки импортов EJB позволяют модулю SCA вызывать реализации EJB определением способа привязки использующего модуля к внешнему EJB. Импорт служб из внешней реализации EJB позволяет пользователям встроить свою бизнес-логику в среду IBM Business Process Manager и и принимать участие в бизнес-процессе.

Для создания привязок импортов EJB используется Integration Designer. Для создания привязок можно использовать любую из следующих процедур:

- Создание импорта EJB с помощью мастера внешних служб

  Для создания импорта EJB на основе существующей реализации можно использовать мастер внешних служб в Integration Designer. Мастер внешних служб создает службы на основе заданных критериев. Затем он создает бизнес-объекты, интерфейсы и файлы импорта на основе найденных служб.
- Создание импорта EJB с помощью редактора сборки

  Импорт EJB можно создать в пределах диаграммы сборки с помощью редактора сборки Integration Designer. Для создания привязки EJB из палитры можно использовать Импорт или класс Java.

Созданный импорт имеет привязки данных, которые устанавливают связь Java-WSDL вместо вызова компонента моста Java. Можно напрямую соединить компонент со ссылкой языка описания веб-служб (WSDL) с импортом, который связывается со службой на основе EJB с помощью интерфейса Java.

Импорт EJB может взаимодействовать с бизнес-логикой Java EE с помощью программной модели EJB 2.1 или EJB 3.0.

Вызов бизнес-логики Java EE может быть локальным (только для EJB 3.0) или удаленным.

- Локальный вызов используется для вызова бизнес-логики Java EE, которая находится на том же сервере, что и импорт.

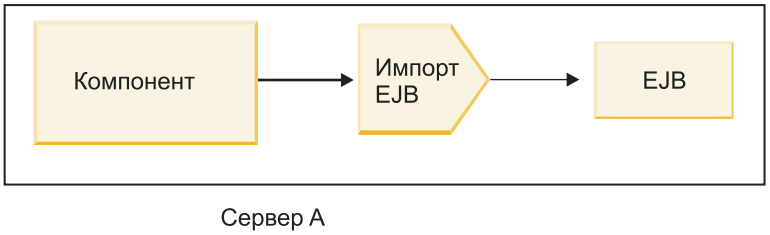


Рисунок 28. Локальный вызов EJB (только для EJB 3.0)

- Удаленный вызов используется для вызова бизнес-логики Java EE, которая не находится на том же сервере, что и импорт.

  Например, на следующем рисунке импорт EJB использует удаленный метод вызова через протокол Internet InterORB (RMI/IIOP) для вызова метода EJB на другом сервере.

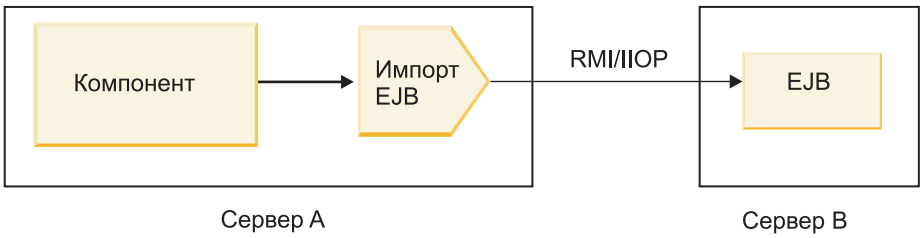


Рисунок 29. Удаленный вызов EJB

При настройке привязки EJB Integration Designer использует имя JNDI для определения уровня программной модели EJB и типа вызова (локального или удаленного).

Привязки импортов EJB содержат следующие основные компоненты:

- Обработчик данных JAX-WS
- Селектор ошибок EJB
- Селектор функций импорта EJB

Если пользовательский сценарий не основан на преобразовании JAX-WS, то могут потребоваться обработчик пользовательских данных, селектор функций и селектор по умолчанию для выполнения задач, которые в противном случае выполняются компонентами привязок импортов EJB. Это включает преобразование, обычно выполняемое пользовательским алгоритмом преобразования.

Привязки экспортов EJB:

Внешние приложения Java EE могут вызывать компонент SCA при помощи привязки экспорта EJB. Экспорт EJB позволяет экспортировать компоненты SCA, чтобы внешние приложения Java EE могли вызывать эти компоненты с помощью программной модели EJB.

Примечание: Экспорт EJB является EJB без сохранения состояния.

Для создания привязок EJB используется Integration Designer. Для создания привязок можно использовать любую из следующих процедур:

- Создание привязок экспортов EJB с помощью мастера внешних служб

  Для создания службы экспорта EJB на основе существующей реализации можно использовать мастер внешних служб в Integration Designer. Мастер внешних служб создает службы на основе заданных критериев. Затем он создает бизнес-объекты, интерфейсы и файлы экспорта на основе найденных служб.
- Создание привязок экспортов EJB с помощью редактора сборки

  Экспорт EJB можно создать с помощью редактора сборки Integration Designer.

Важное замечание: Клиент Java 2 Platform, Standard Edition (J2SE) не может вызывать клиент экспорта EJB, который создан в Integration Designer.

Привязку можно создать из существующего компонента SCA или можно создать экспорт с привязкой EJB для интерфейса Java.

- При создании экспорта для существующего компонента SCA с существующим интерфейсом WSDL экспорту назначается интерфейс Java.
- При создании экспорта для интерфейса Java для экспорта можно выбрать интерфейс WSDL или Java.

  Примечание: Интерфейс Java, используемый для создания экспорта EJB, имеет следующие ограничения в отношении объектов (параметров ввода и вывода и исключительных ситуаций), передаваемых в качестве параметров по удаленному вызову:
  - Они должны быть конкретного типа (а не интерфейсом или абстрактным типом).

– Они должны соответствовать спецификации EJB (Enterprise JavaBean). Они должны быть сериализуемы и иметь безаргументный конструктор по умолчанию, а все свойства должны быть доступны через методы getter и setter.

  Информация о спецификации EJB (Enterprise JavaBean) приведена на веб-сайте Sun Microsystems, Inc., по адресу http://java.sun.com.

Кроме того, исключительная ситуация должна быть проверенной исключительной ситуацией, унаследованной из java.lang.Exception, и она должна быть единственной (то есть выброс проверенных исключительных ситуаций нескольких типов не поддерживается).

Отметим также, что бизнес-интерфейс Java EnterpriseBean является простым интерфейсом Java и не должен расширять javax.ejb.EJBObject или javax.ejb.EJBLocalObject. Методы бизнес-интерфейса не должны выбрасывать java.rmi.Remote.Exception.

Привязки экспорта EJB могут взаимодействовать с бизнес-логикой Java EE с помощью программной модели EJB 2.1 или EJB 3.0.

Вызов может быть локальным (только для EJB 3.0) или удаленным.

- Локальный вызов используется, когда бизнес-логика Java EE вызывает компонент SCA, который находится на том же сервере, что и экспорт.
- Удаленный вызов используется, когда бизнес-логика Java EE не находится на том же сервере, что и экспорт.

  Например, на следующем рисунке EJB использует RMI/IIOP для вызова компонента SCA на другом сервере.

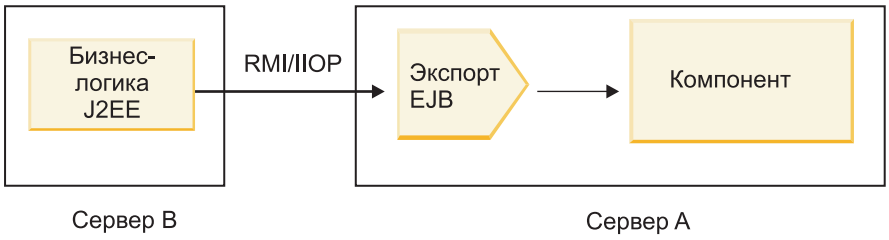


Рисунок 30. Удаленный вызов из клиента компонента SCA при помощи экспорта EJB

При настройке привязки EJB Integration Designer использует имя JNDI для определения уровня программной модели EJB и типа вызова (локального или удаленного).

Привязки экспортов EJB содержат следующие основные компоненты:

- Обработчик данных JAX-WS
- Селектор функций экспорта EJB

Если пользовательский сценарий не основан на преобразовании JAX-WS, то могут потребоваться обработчик пользовательских данных и селектор функций для выполнения задач, которые в противном случае выполняются компонентами привязок экспортов EJB. Это включает преобразование, обычно выполняемое пользовательским алгоритмом преобразования.

Свойства привязки EJB:

Привязки импортов EJB используют настроенные для них имена JNDI для определения программной модели EJB и типа вызова (локального или удаленного). Привязки импортов и экспортов EJB используют обработчик данных JAX-WS для преобразования данных. Привязки импортов EJB использует селектор функций импорта EJB и селектор ошибок EJB, а привязка экспорта EJB использует селектор функций экспорта EJB.

Имена JNDI и привязки импортов EJB:

При настройке привязки EJB на импорте Integration Designer использует имя JNDI для определения уровня программной модели EJB и типа вызова (локального или удаленного).

Если имя JNDI не задано, то используется привязка интерфейса EJB по умолчанию. Создаваемые имена по умолчанию зависят от вызываемого объекта: EJB 2.1 JavaBean или EJB 3.0 JavaBean.

Примечание: Более подробная информация о соглашении об именах содержится в разделе "Обзор связывания приложений EJB 3.0" справочной системе WebSphere Application Server Information Center.

- EJB 2.1 JavaBean

  Имя JNDI по умолчанию, предварительно выбираемое Integration Designer , представляет собой привязку EJB 2.1 по умолчанию, which takes the которое принимает вид **ejb/** плюс домашний интерфейс, разделенные косыми чертами.

  Например, для домашнего интерфейса EJB 2.1 JavaBean для com.mycompany.myremotebusinesshome привязкой по умолчанию является:

  ```
  ejb/com/mycompany/myremotebusinesshome
  ```

  Для EJB 2.1 поддерживается только удаленный вызов EJB.

- EJB 3.0 JavaBean

  Именем JNDI по умолчанию, предварительно выбираемым Integration Designer для локального JNDI является полное имя класса локального интерфейса, перед которым стоит **ejblocal:**. Например, для полного имени интерфейса локального интерфейса com.mycompany.mylocalbusiness предварительно выбранным EJB 3.0 JNDI является:

  ```
  ejblocal:com.mycompany.mylocalbusiness
  ```

  Для удаленного интерфейса com.mycompany.myremotebusiness предварительно выбранным EJB 3.0 JNDI является полное имя интерфейса:

  ```
  com.mycompany.myremotebusiness
  ```

  Привязки приложений EJB 3.0 по умолчанию описываются в следующем расположении: Привязки приложений EJB 3.0 – обзор.

  Integration Designer будет использовать "короткое" имя в качестве расположения JNDI по умолчанию для EJB, использующих программную модель версии 3.0.

  Примечание: Если развернутая ссылка JNDI целевого EJB отличается от расположения привязки JNDI по умолчанию из-за использования пользовательского преобразования или настройки, то целевое имя JNDI должно быть задано правильно. Можно задать имя в Integration Designer до развертывания или для привязки импорта можно изменить имя в административной консоли (после развертывания) для соответствия имени JNDI целевого EJB.

Дополнительная информация о создании привязок EJB приведена в разделе Работа с привязками EJB справочной системы Integration Designer Information Center.

Обработчик данных JAX-WS:

Привязка импорта Enterprise JavaBean (EJB) использует обработчик данных JAX-WS для превращения бизнес-объекта запроса в параметры объекта Java и для превращения возвращенного значения объекта Java в бизнес-объект ответа. Привязка экспорта EJB использует обработчик данных JAX-WS для превращения EJB запросов в бизнес-объекты запросов и для превращения бизнес-объекта ответа в возвращенное значение.

Этот обработчик данных преобразует данные из интерфейса WSDL со спецификацией SCA в целевой интерфейс EJB Java (и наоборот) с помощью Java API для спецификации XML Web Services (JAX-WS) и Java Architecture для спецификации XML Binding (JAXB).

Примечание: В настоящее время поддерживаются только спецификации JAX-WS 2.1.1 и JAXB 2.1.3.

Обработчик данных, указанный на уровне привязки EJB, используется для обработки запросов, ответов, ошибок и исключительных ситуаций во время выполнения.

Примечание: Для ошибок с помощью свойства конфигурации faultBindingType можно определить конкретный обработчик данных для каждой ошибки. Это переопределяет значение, заданное на уровне привязки EJB.

Обработчик данных JAX-WS используется по умолчанию, когда привязка EJB содержит интерфейс WSDL. Этот обработчик данных не может использоваться для преобразования сообщения SOAP, представляющего вызов JAX-WS объекта данных.

Привязка импорта EJB использует обработчик данных для преобразования объекта данных в массив Java Object (Object[]). Во время исходящей связи происходит следующая обработка:

1. Привязка EJB задает ожидаемый тип, ожидаемый элемент и имя целевого метода в BindingContext согласно заданным в WSDL.
2. Привязка EJB вызывает метод преобразования для объекта данных, требующего преобразования данных.
3. Обработчик данных возвращает Object[], представляющий параметры метода (в порядке их определения в методе).
4. Привязка EJB использует Object[] для вызова метода на целевом интерфейсе EJB.

Привязка также подготавливает Object[] к обработке ответа из вызова EJB.

- Первым элементом в Object[] является возвращенное значение из вызова метода Java.
- Последующие значения представляют параметры ввода для метода.

Это требует поддержки типов In/Out и Out параметров.

Для параметров типа Out значения должны возвращаться в объект данных ответа.

Обработчик данных обрабатывает и преобразует значения в Object[] и затем возвращает ответ объекту данных.

Обработчик данных поддерживает xs:AnyType, xs:AnySimpleType и xs:Any наряду с другими типами данных XSD. Для включения поддержки xs:Any используйте **@XmlAnyElement (lax=true)** для свойства EJB (JavaBean) в коде Java, как показано в следующем примере:

```
public class TestType {
 private Object[] object;

 @XmlAnyElement (lax=true)
 public Object[] getObject() {
  return object;
 )

 public void setObject (Object[] object) {
  this.object=object;
 )
)
```

При этом создается объект свойства в TestType и поле xs:any. Значение класса Java, используемое в поле xs:any, должно иметь аннотацию **@XmlAnyElement**. Например, если адресом является класс Java, используемый для заполнения массива объекта, то класс Адрес должен иметь аннотацию **@XmlRootElement**.

Примечание: Для настройки преобразования из типа XSD в типы Java, определенные в спецификации JAX-WS, измените аннотации JAXB в соответствии с требованиями вашего бизнеса. Обработчик данных JAX-WS поддерживает xs:any, xs:anyType и xs:anySimpleType.

Для обработчика данных JAX-WS применяются следующие ограничения:

- Обработчик данных не включает поддержку аннотации **@WebParam** атрибута заголовка.
- Пространство имен для файлов схем бизнес-объектов (файлов XSD) не включает преобразование по умолчанию из имен пакета Java. Аннотация **@XMLSchema** в package-info.java тоже не работает. Единственным путем создания XSD с пространством имен является использование аннотаций **@XmlType** и **@XmlRootElement**. **@XmlRootElement** определяет целевое пространство имен для глобального элемента в типах JavaBean.
- Мастер импорта EJB не создает файлов XSD для несвязанных классов. Версия 2.0 не поддерживает аннотацию **@XmlSeeAlso**, поэтому если дочерний класс не указывается непосредственно из родительского класса, то XSD не создается. Решением этой проблемы является выполнение SchemaGen для каждого дочернего класса.

  SchemaGen – это утилита командной строки (в каталоге *установочный-каталог-WPS*/bin), предусмотренная для создания файлов XSD для данного EJB. Для того чтобы это решение работало, эти XSD необходимо вручную скопировать в модуль.

Селектор ошибок EJB:

Селектор ошибок EJB определяет, приведет ли вызов EJB к ошибке, исключительной ситуации во время выполнения или к успешному ответу.

При обнаружении ошибки селектор ошибок EJB возвращает собственное имя ошибки среде выполнения привязки, чтобы обработчик данных JAX-WS мог преобразовать объект исключительной ситуации в бизнес-объект ошибки.

При успешном ответе (без ошибок) привязка импорта EJB собирает массив объекта Java (Object[]) для возврата значений.

- Первым элементом в Object[] является возвращенное значение из вызова метода Java.
- Последующие значения представляют параметры ввода для метода.

Это требует поддержки типов In/Out и Out параметров.

Для сценариев с исключительной ситуацией привязка собирает Object[], и первый элемент представляет исключительную ситуацию, выброшенную методом.

Селектор ошибок может возвращать любое из следующих значений:

Таблица 35. Возвращаемые значения

Тип	**Возвращаемое значение**	**Описание**
Ошибка	**ResponseType.FAULT**	Возвращается, когда переданный Object[] содержит объект исключительной ситуации.
Исключительная ситуация во время выполнения	**ResponseType.RUNTIME**	Возвращается, если объект исключительной ситуации не совпадает ни с одним из типов исключительных ситуаций, объявленных в методе.
Нормальный ответ	**ResponseType.RESPONSE**	Возвращается во всех остальных случаях.

Если селектор ошибок возвращает значение **ResponseType.FAULT**, то возвращается собственное имя ошибки. Это собственное имя ошибки используется привязкой для определения соответствующего имени ошибки WSDL из модели и для вызова соответствующего обработчика данных ошибок.

Селектор функций EJB:

Привязки EJB используют селектор функций импорта (для исходящей обработки) или селектор функций экспорта (для входящей обработки) для определения вызываемого метода EJB.

Селектор функций импорта

Для исходящей обработки селектор функций импорта извлекает тип метода EJB на основании имени операции, вызванной компонентом SCA, который соединен с импортом EJB. Селектор функций ищет аннотацию @WebMethod на созданном Integration Designer JAX-WS, аннотированном классом Java, для определения имени связанной целевой операции.

- Если аннотация @WebMethod присутствует, то селектор функций использует аннотацию @WebMethod для определения соответствующего преобразования метода Java для метода WSDL.
- Если аннотация @WebMethod отсутствует, то селектор функций предполагает, что имя метода Java совпадает с именем вызванной операции.

Примечание: Этот селектор функций действует только для интерфейса типа WSDL на импорте EJB, но не для интерфейса типа Java на импорте EJB.

Селектор функций возвращает объект java.lang.reflect.Method, который представляет метод интерфейса EJB.

Селектор функций использует массив Java Object (Object[]) для ответа из целевого метода. Первым элементом в Object[] является метод Java с именем WSDL, а вторым элементом Object[] является бизнес-объект ввода.

Селектор функций экспорта

Для входящей обработки селектор функций экспорта извлекает целевой метод, вызываемый из метода Java.

Селектор функций экспорта преобразует имя операции Java, вызванной клиентом EJB, в имя операции в интерфейсе целевого компонента. Имя метода возвращается как строка и обрабатывается средой выполнения SCA в зависимости от типа интерфейса целевого компонента.

Привязки EIS:

Привязки информационной системы предприятия (EIS) обеспечивают связь между компонентами SCA и внешней EIS. Эта связь достигается с помощью экспортов и импортов EIS, которые поддерживают адаптеры ресурсов JCA 1.5 и адаптеры Websphere.

Компоненты SCA могут требовать передачи данных в или из внешней EIS. При создании модуля SCA, требующего такой связи включается (кроме компонента SCA) импорт или экспорт с привязкой EIS для связи с конкретной внешней EIS.

Адаптеры ресурсов в IBM Integration Designer используются в контексте импорта или экспорта. Импорт или экспорт разрабатывается с помощью мастера внешней службы и в процессе разработки включается адаптер ресурсов. Импорт EIS, который позволяет приложению вызывать службу в системе EIS, или экспорт EIS, который позволяет приложению в системе EIS вызывать службу, разработанную в IBM Integration Designer, создается с помощью адаптера ресурсов. Например, можно создать импорт с помощью адаптера JD Edwards для вызова службы в системе JD Edwards.

При использовании мастера внешней службы создается информация о привязке EIS. Для добавления или изменения информации о привязке также можно использовать другой инструмент – редактор сборки. Дополнительная информация приведена в разделе Обращение к внешним службам с помощью адаптеров.

После развертывания модуля с привязкой EIS на сервере административную консоль можно использовать для просмотра информации о привязке или для ее настройки.

Обзор привязок EIS:

Привязка EIS (enterprise information system - информационная система предприятия), используемая совместно с адаптером ресурсов JCA, позволяет получать доступ к службам информационной системы предприятия или предоставлять свои службы для использования в EIS.

В следующем примере показано, каким образом модуль SCA с именем ContactSyncModule синхронизирует информацию о контактах между системами Siebel и SAP.

1. Компонент SCA с именем ContactSync отслеживает (с помощью экспорта приложения EIS с именем Siebel Contact) изменения в контактах Siebel.
2. Компонент SCA ContactSync использует приложение SAP (посредством импорта приложения EIS) для обновления информации о контактах SAP.

Поскольку в системах Siebel и SAP контакты хранятся в разных структурах данных, компонент SCA ContactSync должен выполнять преобразование.

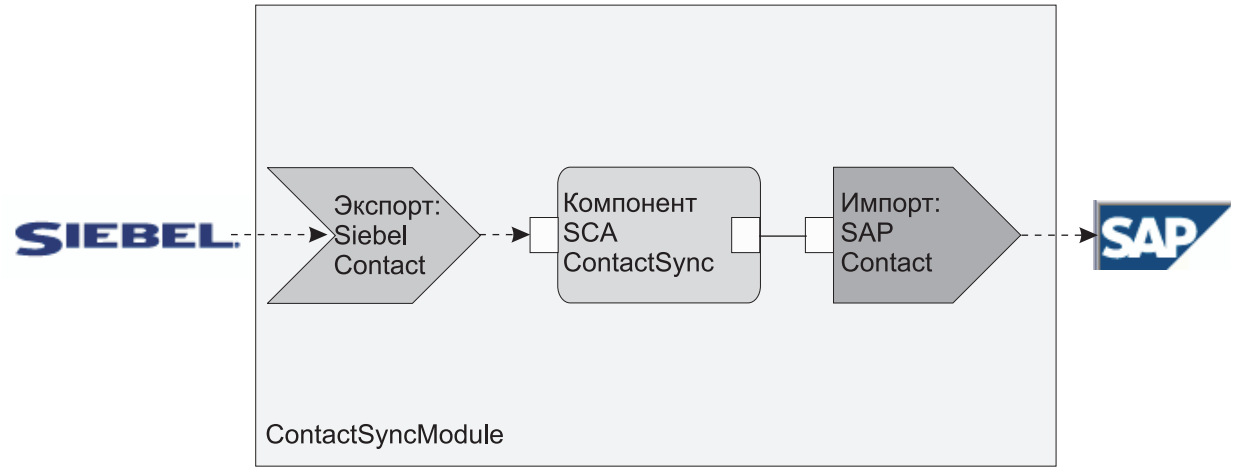


Рисунок 31. Передача данных из системы Siebel в систему SAP

Для экспорта Siebel Contact и импорта SAP Contact настроены соответствующие адаптеры ресурсов.

Основные компоненты привязок EIS:

Импорт EIS является импортом архитектуры компонентов служб (SCA), который позволяет компонентам в модуле SCA использовать приложения EIS, определенные вне модуля SCA. Импорт EIS используется для передачи данных из компонента SCA внешней EIS; экспорт EIS используется для передачи данных из внешней EIS модулю SCA.

Импорты

Ролью импорта EIS является связывание компонентов SCA и внешних систем EIS. Внешние приложения могут рассматриваться как импорт EIS. В этом случае импорт EIS передает данные внешней EIS и может принимать данные в ответ.

Импорт EIS предоставляет компонентам SCA единообразную панель приложений, внешних по отношению к модулю. Это позволяет компонентам связываться с внешней EIS, например SAP, Siebel или PeopleSoft, с помощью совместимой модели SCA.

На стороне клиента импорта имеется интерфейс, предоставляемый приложением импорта EIS, с одним или более методами, каждый из которых получает объекты данных как аргументы и возвращает значения. На стороне реализации имеется общий клиентский интерфейс (CCI), реализуемый адаптером ресурсов.

Динамическая реализация импорта EIS связывает интерфейс стороны клиента с CCI. Импорт преобразует вызов метода на интерфейсе в вызов на CCI.

Привязки создаются на трех уровнях: привязке интерфейса, которая затем использует содержащиеся привязки метода, которые, в свою очередь, используют привязки данных.

Привязка интерфейса связывает интерфейс импорта для соединения с системой EIS, предоставляющей приложение. Это отражает тот факт, что набор приложений, представляемый интерфейсом, предоставляется конкретным экземпляром EIS, а соединение предоставляет доступ к этому экземпляру. Элемент привязки содержит свойства с достаточной информацией для создания соединения (эти свойства являются частью экземпляра javax.resource.spi.ManagedConnectionFactory).

Привязка метода связывает метод с конкретным взаимодействием с системой EIS. Для JCA это взаимодействие характеризуется набором свойств реализации интерфейса javax.resource.cci.InteractionSpec. Элемент взаимодействия привязки метода содержит эти свойства вместе с именем класса, предоставляя таким образом достаточную информацию для осуществления взаимодействия. Привязка метода использует привязки данных, описывающие преобразование аргумента и результат метода интерфейса для представления EIS.

Сценарий выполнения для импорта EIS следующий:

1. Метод на интерфейсе импорта вызывается с помощью программной модели SCA.
2. Запрос, достигающий импорта EIS, содержит имя метода и его аргументы.
3. Импорт сначала создает реализацию привязки интерфейса, а затем, с помощью данных из привязки импорта, создает ConnectionFactory и связывает их. Таким образом, импорт вызывает setConnectionFactory на привязке интерфейса.
4. Создается реализация привязки метода, соответствующая вызываемому методу.
5. Создается и заполняется экземпляр javax.resource.cci.InteractionSpec; затем привязки данных используются для связывания аргументов метода с форматом, понятным адаптеру ресурсов.
6. Интерфейс CCI используется для осуществления взаимодействия.
7. Когда вызов возвращается, привязка данных используется для создания результата вызова, и результат возвращается инициатору.

Экспорты

Ролью экспорта EIS является связывание компонента SCA и внешней EIS. Внешние приложения могут рассматриваться как экспорт EIS. В этом случае внешнее приложение отправляет свои данные в виде периодических уведомлений. Экспорт EIS можно рассматривать как приложение подписки, получающее внешний запрос от EIS. Компонент SCA, который использует экспорт EIS, видит его как локальное приложение.

Экспорт EIS предоставляет компонентам SCA единообразную панель приложений, внешних по отношению к модулю. Это позволяет компонентам связываться с EIS, например SAP, Siebel или PeopleSoft, с помощью совместимой модели SCA.

Экспорт имеет реализацию получателя, получающего запросы от EIS. Получатель реализует интерфейс получателя конкретного адаптера ресурсов. Экспорт также содержит компонент, реализующий интерфейс, предоставляемый EIS через экспорт.

Динамическая реализация экспорта EIS соединяет получателя с компонентом, реализующим интерфейс. Экспорт преобразует запрос EIS в запрос соответствующей операции на компоненте. Привязки создаются на трех уровнях: привязке получателя, которая затем использует содержащуюся привязку внутреннего метода, которая, в свою очередь, используют привязку данных.

Привязка получателя связывает получателя, получающего запросы, с компонентом, предоставляемым через экспорт. Определение экспорта содержит имя компонента; среда выполнения находит его и направляет запросы ему.

Привязка внутреннего метода связывает внутренний метод или тип события, получаемого получателем, с операцией, реализуемой компонентом, предоставленным посредством экспорта. Нет взаимосвязи между методом, вызываемым на получателе, и типом события; все события поступают через один или несколько методов получателя. Привязка внутреннего метода использует определенный в экспорте селектор функций для извлечения имени внутреннего метода из входящих данных и привязок данных для связывания формата данных EIS с форматом, понятным компоненту.

Сценарий выполнения для экспорта EIS следующий:

1. Запрос EIS запускает вызов метода на реализации получателя.
2. Получатель находит и вызывает экспорт, передавая ему все аргументы вызова.
3. Экспорт создает реализацию привязки получателя.
4. Экспорт создает экземпляр селектора функций и задает его на привязке получателя.
5. Экспорт инициализирует привязки внутренних методов и добавляет их в привязку получателя. Для каждой привязки внутреннего метода также инициализируются привязки данных.
6. Экспорт вызывает привязку получателя
7. Привязка получателя находит экспортированные компоненты и использует селектор функций для извлечения имени внутреннего метода.
8. Это имя используется для поиска привязки внутреннего метода, которая затем вызывает целевой компонент.

Стиль взаимодействия с адаптером позволяет привязке EIS вызывать целевой компонент либо асинхронно (по умолчанию), либо синхронно.

Адаптеры ресурсов

Импорт или экспорт разрабатывается с помощью мастера внешней службы и в процессе разработки включается адаптер ресурсов. Адаптеры, которые поставляются с IBM Integration Designer, используемым для доступа к системам CICS, IMS, JD Edwards, PeopleSoft, SAP и Siebel, предназначаются только для целей разработки и тестирования. Это означает, что они используются для разработки и тестирования приложений.

После развертывания приложения необходимы лицензионные динамические адаптеры для выполнения приложения. Однако при создании службы можно вложить адаптер со службой. Лицензирование адаптера может разрешать использовать вложенный embedded adapter адаптер как лицензионный динамический адаптер. Эти адаптеры согласуются с архитектурой коннекторов Java EE Connector Architecture (JCA 1.5). Открытый стандарт JCA является стандартом Java EE для связи с EIS. JCA предоставляет управляемую среду, то есть Quality of Service (QoS) обеспечивается сервером приложений, который предоставляет управление жизненным циклом и защиту транзакциям. Они также совместимы со спецификацией поиска метаданных предприятия, кроме IBM CICS ECI Resource Adapter и IBM IMS Connector for Java.

Более старый набор адаптеров бизнес-интеграции WebSphere также поддерживается мастером.

Ресурсы EE

Модуль EIS, как модуль SCA, который следует шаблону модуля EIS, можно развернуть на платформе Java EE.

Развертывание модуля EIS на платформе Java EE позволяет получить приложение, которое готово к выполнению, упаковано как файл EAR и развернуто на сервере. Создаются все артефакты и ресурсы Java EE; приложение настраивается и готово к выполнению.

Динамические свойства спецификации взаимодействия и спецификации соединения JCA:

Привязка EIS принимает ввод для спецификаций InteractionSpec и ConnectionSpec, заданный в виде корректно определенного дочернего объекта данных, связанного с полезными данными. Это позволяет взаимодействовать с адаптером ресурсов в режиме запрос-ответ с помощью InteractionSpec или выполнять идентификацию компонента с помощью ConnectionSpec.

javax.cci.InteractionSpec содержит информацию о том, каким образом следует обработать запрос на взаимодействие с адаптером ресурсов. Кроме того, в нем может содержаться информация о ходе взаимодействия после отправки запроса. Такое двустороннее взаимодействие иногда называется *диалогом*.

Привязка EIS ожидает, что полезные данные, которые передаются в качестве аргумента в адаптер ресурсов, содержат дочерний объект с именем **properties**. Этот объект свойств должен содержать пары имя-значение, причем имена свойств спецификации взаимодействия должны быть заданы в определенном формате. Действуют следующие правила форматирования:

- Имена должны начинаться с префикса **IS**, за которым должно следовать имя свойства. Например, если у спецификации взаимодействия есть свойство JavaBeans, которое называется **InteractionId**, то имя этого свойства должно быть указано как **ISInteractionId**.
- Пара имя-значение задает имя и значение простого типа для свойства спецификации взаимодействия.

В этом примере в определении интерфейса указано, что вводом для операции служит объект данных **Account**. Этот интерфейс вызывает приложение привязки импорта EIS для отправки и получения динамического свойства InteractionSpec с именем **workingSet** и значением **xyz**.

Хранящиеся на сервере бизнес-график или бизнес-объекты содержат бизнес-объект **properties**, поддерживающий отправку данных в формате протокола вместе с полезными данными. Это встроенный бизнес-объект **properties**, который не требуется указывать в схеме XML при создании бизнес-объекта. Его можно создать и сразу же использовать. Если на основе схемы XML определены пользовательские типы данных, то укажите элемент **properties**, содержащий ожидаемые пары имя-значение.

```
BOFactory dataFactory = (BOFactory) \
serviceManager.locateService("com/ibm/websphere/bo/BOFactory");
 //Wrapper для интерфейсов с оболочкой doc-lit,
 //перейдите к полезным данным, если doc-lit не используется
 DataObject docLitWrapper = dataFactory.createByElement /
 ("http://mytest/eis/Account", "AccountWrapper");
```

Создать полезные данные.

```
DataObject account = docLitWrapper.createDataObject(0);
  DataObject accountInfo = account.createDataObject("AccountInfo");
  //Настройка полезных данных


  //Создание свойств для динамического взаимодействия

  DataObject properties = account.createDataObject("properties");
```

Задать ожидаемое значение **xyz** для свойства с именем workingSet.

```
properties.setString("ISworkingSet", "xyz");


  //Вызвать службу с аргументом

  Service accountImport = (Service) \
  serviceManager.locateService("AccountOutbound");
  DataObject result = accountImport.invoke("createAccount", docLitWrapper);

  //Получить возвращенное свойство
```

```
DataObject retProperties = result.getDataObject("properties");

String workingset = retProperties.getString("ISworkingSet");
```

Свойства ConnectionSpec можно использовать для динамической идентификации компонента. При этом действуют указанные выше правила, однако префикс имени свойства должен быть равен **CS** (а не **IS**). Свойства ConnectionSpec нельзя передавать в обоих направлениях. Один объект данных **properties** может содержать свойства IS и CS.

Для применения свойств ConnectionSpec укажите в параметре **resAuth** привязки импорта значение **Application**. Кроме того, адаптер ресурсов должен поддерживать идентификацию компонентов. За дополнительной информацией обратитесь к главе 8 публикации J2EE Connector Architecture Specification.

Внешние клиенты с привязками EIS:

Сервер может обмениваться сообщениями с внешними клиентами через привязки EIS.

Внешний клиент, такой как веб-портал или EIS, должен отправить сообщение модулю SCA на сервере, или же его необходимо вызвать в компоненте на сервере.

Клиент вызывает импорт EIS как и любое другое приложение, используя интерфейс DII или интерфейс Java.

1. Внешний клиент создает экземпляр ServiceManager и выполняет поиск импорта EIS по имени. Результатом поиска является реализация интерфейса службы.
2. Клиент создает входной аргумент - простой объект данных, создаваемый динамически с помощью схемы объекта данных. Это действие выполняется с помощью реализации интерфейса DataFactory объекта Service Data Object.
3. Внешний клиент вызывает EIS и получает необходимые результаты.

Вместо этого клиент может вызвать импорт EIS, используя интерфейс Java.

1. Клиент создает экземпляр ServiceManager и выполняет поиск импорта EIS по имени. Результатом поиска является интерфейс Java импорта EIS.
2. Клиент создает входной аргумент и типизированный объект данных.
3. Клиент вызывает EIS и получает необходимые результаты.

Интерфейс экспорта EIS определяет интерфейс экспортированного компонента SCA, который доступен внешним приложениям EIS. Можно считать, что это тот интерфейс, который внешнее приложение (такое как SAP или PeopleSoft) вызывает через реализацию среды выполнения приложения экспорта EIS.

Экспорт использует привязку EISExportBinding для связывания экспортированных служб с внешним приложением EIS. Это дает возможность настроить приложение, содержащее модуль SCA, для отслеживания запросов службы EIS. Привязка экспорта EIS задает правила преобразования определения входящих событий в том формате, который поддерживается адаптером ресурсов (через интерфейсы Java EE Connector Architecture), в вызовы операций SCA.

Для применения EISExportBinding необходимо, чтобы внешние службы EIS использовали контракты на входящие соединения Java EE Connector Architecture 1.5. При использовании EISExportBinding необходимо, чтобы обработчик данных или привязка данных были заданы на уровне привязки или на уровне метода.

Привязки JMS:

Провайдер службы сообщений Java (JMS) включает обмен сообщениями на основании API и программной модели службы сообщений Java. Он предоставляет фабрики соединений JMS для создания целевых объектов JMS и для отправки и получения сообщений.

Привязки JMS можно использовать для взаимодействия с привязкой провайдера шины интеграции служб (Service Integration Bus, SIB). Они совместимы с JMS и JCA 1.5.

Привязки импортов и экспортов JMS обеспечивают модулю архитектуры компонентов служб (SCA) возможность обращаться к и получать сообщения из внешних систем JMS.

Привязки импортов и экспортов JMS обеспечивают интеграцию с приложениями JMS с помощью провайдера SIB JMS на основе JCA 1.5, который включен в WebSphere Application Server. Другие адаптеры ресурсов JMS на основе JCA 1.5 не поддерживаются.

Привязки JMS – обзор:

Привязки JMS обеспечивают взаимодействие между средой архитектуры компонентов служб (SCA) и системами JMS.

Привязки JMS

Основные компоненты привязок импортов и экспортов JMS следующие:

- Адаптер ресурсов: обеспечивает управляемую двухстороннюю связь между модулем SCA и внешними системами JMS
- Соединения: охватывают виртуальное соединение между клиентом и приложением провайдера
- Целевые объекты: используются клиентом для определения целевого объекта сообщений, которые он отправляет, или исходного объекта сообщений, которые он получает
- Идентификационные данные: используются для защиты доступа к привязке

Основные компоненты привязок JMS

Специальные заголовки

Свойства специальных заголовков используются в импортах и экспортах JMS для информирования целевого объекта о том, как обрабатывать сообщение.

Например, TargetFunctionName выполняет преобразование из внутреннего метода в рабочий метод.

Ресурсы EE

При развертывании импортов и экспортов JMS в среду Java EE создаются многие ресурсы Java EE.

`ConnectionFactory`
Используется клиентами для соединения с провайдером JMS.

`ActivationSpec`
Используется импортами для получения ответа на запрос; используется экспортами при настройке конечных точек сообщений, которые представляют получателей сообщений в их взаимодействиях с системой обмена сообщениями.

`Целевые объекты`

- Назначение отправления: в случае импорта – это куда отправляется запрос или исходящее сообщение; в случае экспорта – это получатель, которому будет отправлено ответное сообщение в отсутствие замещающего поля заголовка JMSReplyTo во входящем сообщении.
- Назначение получения: это целевой объект размещения входящего сообщения; в случае импорта – это ответ, а в случае экспорта – это запрос.
- Назначение обратного вызова: целевая система SCA JMS для хранения информации о зависимостях. Не производите чтение или запись в это назначение.

Задача установки создает ConnectionFactory и три объекта назначения. Она также создает ActivationSpec для включения динамического получателя сообщений для прослушивания ответов в назначении получения. Свойства этих ресурсов определяются в файле импорта или экспорта.

Интеграция и адаптеры ресурсов JMS:

Служба сообщений Java (JMS) обеспечивает интеграцию через доступный адаптер ресурсов JMS на основе JCA 1.5. Полная поддержка интеграции JMS предусматривается для адаптера ресурсов JMS шины интеграции служб (SIB).

Используйте провайдера JMS для адаптера ресурсов JCA 1.5, когда требуется интеграция с внешней системой JMS, совместимой с JCA 1.5. Внешние службы, совместимые с JCA 1.5, могут получать сообщения и отправлять сообщения для интеграции с вашими компонентами архитектуры компонентов служб (SCA) с использованием адаптера ресурсов SIB JMS.

Использование адаптеров ресурсов JCA 1.5 других провайдеров не поддерживается.

Привязки импорта и экспорта JMS:

С помощью привязок импорта и экспорта JMS можно настроить модули SCA взаимодействовать со службами, предоставляемыми внешними приложениями JMS.

Привязки импортов JMS

Соединения со связанным провайдером JMS целевых объектов JMS устанавливаются с помощью фабрики соединений JMS. Административные объекты фабрики соединений используются для управления фабриками соединений JMS для провайдера обмена сообщениями по умолчанию.

Взаимодействие с внешними системами JMS включает использование целевых объектов для отправки запросов и получения ответов.

Для привязки импорта JMS поддерживаются два типа сценариев использования в зависимости от типа вызываемой операции:

- Односторонний: Импорт JMS помещает сообщение на целевой объект отправления, настроенный в привязке импорта. Поле replyTo заголовка JMS остается пустым.
- Двухсторонний (запрос-ответ): Импорт JMS помещает сообщение на целевой объект отправления и затем сохраняет ответ, который он получает из компонента SCA.

  Привязку импорта можно настроить (с помощью поля **Схема зависимости ответа** в Integration Designer) на ожидание ИД зависимости ответного сообщения, скопированного из ИД сообщения-запроса (по умолчанию) или из ИД зависимости сообщения-запроса. Привязку импорта также можно настроить на использование временного динамического целевого объекта ответа, чтобы связать ответы с запросами. Временный целевой объект создается для каждого запроса, и импорт использует этот целевой объект для получения ответа.

  Целевой объект получения задается в свойстве заголовка replyTo исходящего сообщения. Получатель сообщения развертывается для прослушивания на целевом объекте получения, и по получении ответа получатель сообщения передает этот ответ назад компоненту.

Как для одностороннего, так и двухстороннего сценария могут быть заданы динамические и статические свойства заголовков. Статические свойства можно задать из привязки метода импорта JMS. Некоторые из этих свойств имеют специальные значения для выполнения SCA JMS.

Важно отметить, что JMS является асинхронной привязкой. Если вызывающий компонент вызывает импорт JMS синхронно (в двухсторонней операции), то вызывающий компонент блокируется до возвращения ответа службой JMS.

рис. 32 на стр. 121 иллюстрирует, как импорт связывается с внешней службой.

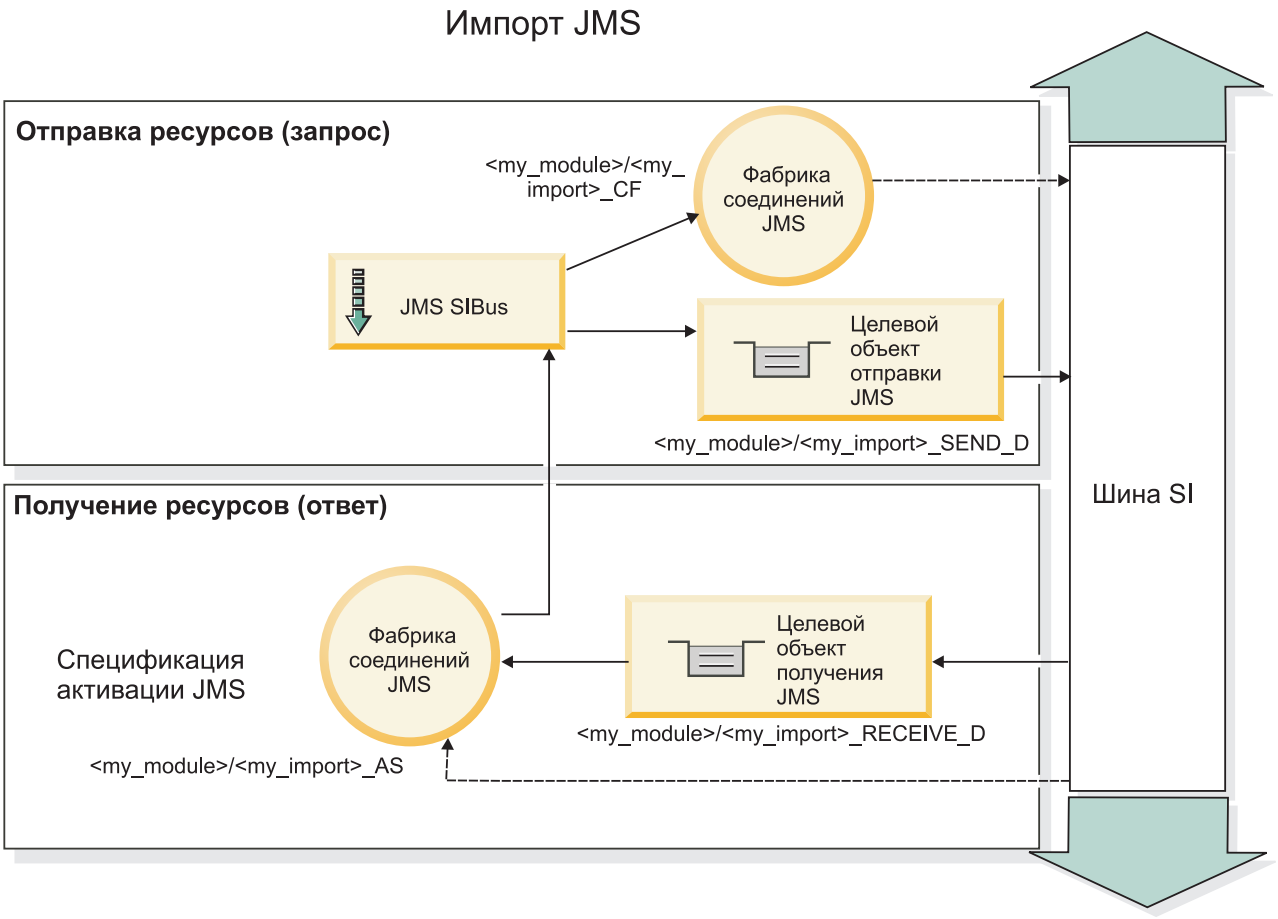


Рисунок 32. Ресурсы привязок импортов JMS

Привязки экспортов JMS

Привязки экспортов JMS обеспечивают средства для модуля SCA по предоставлению служб внешним приложениям JMS.

Соединение, включенное в экспорт JMS, является настраиваемой спецификацией активации.

Экспорт JMS имеет целевые объекты отправления и получения.

- Целевой объект получения – это куда должно быть помещено входящее сообщение для целевого компонента.
- Целевой объект отправления – это куда будет отправлен ответ, если входящее сообщение не переопределило его с помощью свойства заголовка replyTo.

Получатель сообщений развертывается для прослушивания запросов, приходящих на целевой объект получения, указанный в привязке экспорта. Целевой объект, заданный в поле send, используется для отправки ответа на входящий запрос, если вызванный компонент предоставляет ответ. Целевой объект, заданный в поле replyTo входящего сообщения, переопределяет целевой объект, указанный в поле send.

На рисунке рис. 33 на стр. 122 показано, как внешний инициатор запроса связывается с экспортом.

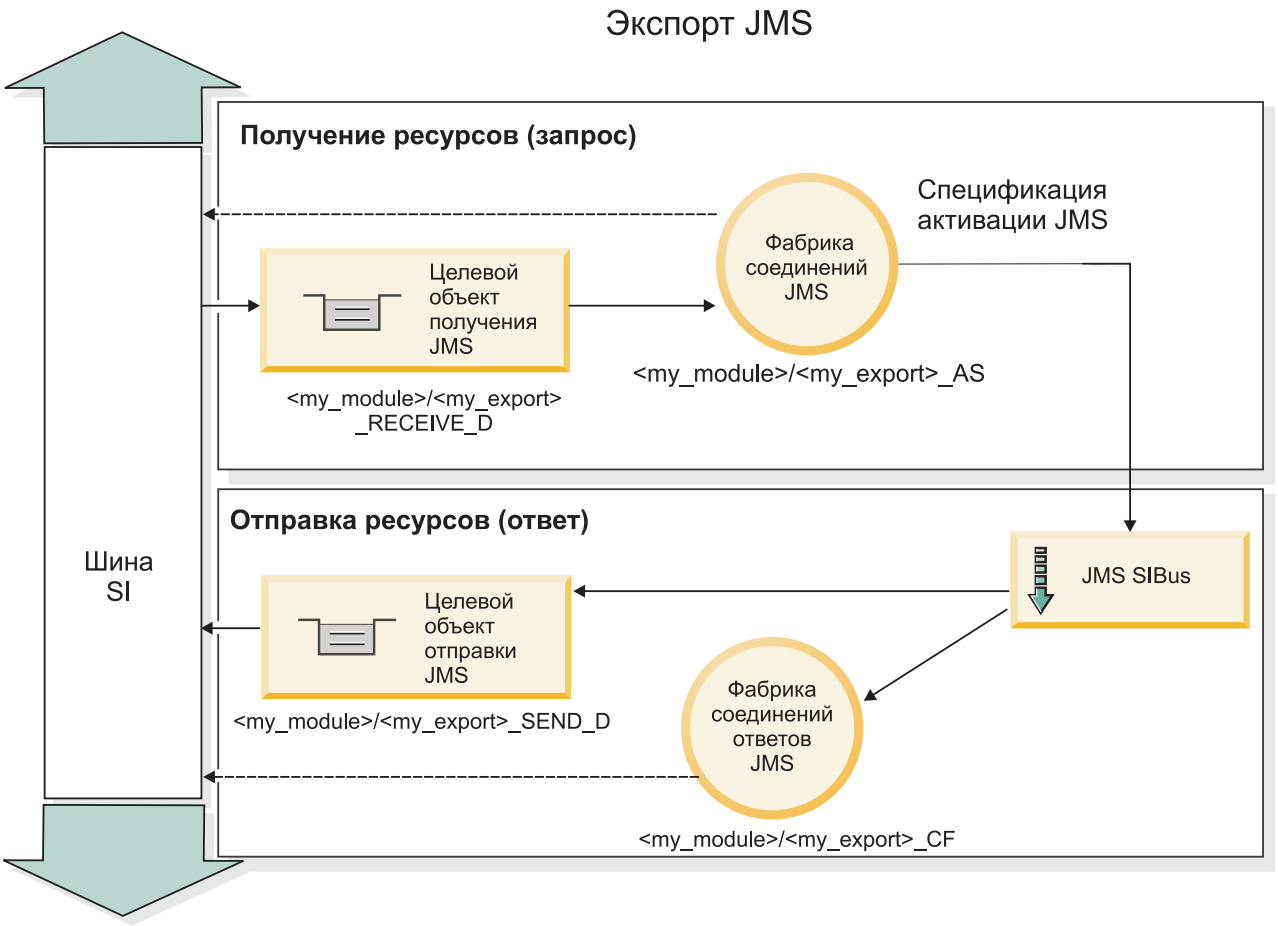


Рисунок 33. Ресурсы привязок экспортов JMS

Заголовки JMS:

Сообщение JMS содержит заголовки двух типов: системный заголовок JMS и несколько свойств JMS. Оба типа заголовков могут быть доступны либо в модуле передачи объекта сообщения службы (SMO), либо с помощью API ContextService.

Системный заголовок JMS

Системный заголовок JMS представляется в SMO элементом JMSHeader, который содержит все поля, типичные для заголовка JMS. Несмотря на то, что они могут изменяться в передаче (или ContextService), некоторые поля системного заголовка JMS, заданные в SMO, не будут передаваться в исходящем сообщении JMS, поскольку они переопределяются системными или статическими значениями.

Основные поля в системном заголовке JMS, которые могут обновляться в передаче (или ContextService), следующие:

- **JMSType** и **JMSCorrelationID** - значения конкретных уже заданных свойств заголовка сообщения
- **JMSDeliveryMode** - значения для режима доставки (постоянные или временные; по умолчанию - постоянные)
- **JMSPriority** - значение приоритета (от 0 до 9; по умолчанию применяется приоритет-JMS-по-умолчанию)

Свойства JMS

Свойства JMS представляются в SMO как записи в списке Свойства. Эти свойства можно добавлять, обновлять или удалять в передаче или с помощью API ContextService.

Свойства также можно задавать статически в привязке JMS. Свойства, которые заданы статически, переопределяют значения (с тем же именем), которые задаются динамически.

Пользовательские свойства, передаваемые из других привязок (например, привязки HTTP), будут выводиться в привязке JMS как свойства JMS.

Параметры распространения заголовков

Распространением системного заголовка и свойств JMS либо из входящего сообщения JMS в последующие компоненты, либо из предыдущих компонентов в исходящее сообщение JMS можно управлять с помощью флага Распространять заголовок протокола на привязке.

Когда флаг Распространять заголовок протокола включен, информация заголовка может передаваться сообщению или целевому компоненту согласно следующему списку:

- Запрос экспорта JMS

  Заголовок JMS, полученный в сообщении, будет распространяться целевым компонентам при помощи контекстной службы. Свойства JMS, полученные в сообщении, будут распространяться целевым компонентам при помощи контекстной службы.
- Ответ экспорта JMS

  Все поля заголовка JMS, заданные в контекстной службе, будут использоваться в исходящем сообщении, если они не переопределяются статическими свойствами, заданными в привязке экспорта JMS. Все свойства, заданные в контекстной службе, будут использоваться в исходящем сообщении, если они не переопределяются статическими свойствами, заданными в привязке экспорта JMS.
- Запрос импорта JMS

  Все поля заголовка JMS, заданные в контекстной службе, будут использоваться в исходящем сообщении, если они не переопределяются статическими свойствами, заданными в привязке импорта JMS. Все свойства, заданные в контекстной службе, будут использоваться в исходящем сообщении, если они не переопределяются статическими свойствами, заданными в привязке импорта JMS.
- Ответ импорта JMS

  Заголовок JMS, полученный в сообщении, будет распространяться целевым компонентам при помощи контекстной службы. Свойства JMS, полученные в сообщении, будут распространяться целевым компонентам при помощи контекстной службы.

Схема зависимостей временного динамического целевого объекта ответа JMS:

Схема зависимостей временного динамического целевого объекта ответа вызывает уникальную динамическую очередь или раздел, создаваемый для каждого отправления запроса.

Статический целевой объект ответа, задаваемый в импорте, используется для получения свойств очереди или раздела временного динамического целевого объекта. Это задается в поле **ReplyTo** запроса, и импорт JMS прослушивает ответы на этом целевом объекте. По получении ответа он снова помещается в очередь статического целевого объекта ответа для асинхронной обработки. Поле **CorrelationID** ответа не используется и не требует определения.

Операционные неполадки

Когда используется временный динамический целевой объект, ответ должен использоваться в той же нити, что и отправленный запрос. Запрос должен отправляться вне глобальной транзакции и должен быть зафиксирован до получения конечной службой и возврата ответа.

Сохранность

Временные динамические очереди являются короткоживущими сущностями и не гарантируют того же уровня сохранности как статическая очередь или раздел. Временная динамическая очередь или раздел не сохраняется после перезапуска сервера, а также и содержащиеся сообщения. После повторного помещения сообщения в очередь статического целевого объекта ответа оно сохраняется в ней в течение времени, определенного в сообщении.

Тайм-аут

Импорт ожидает получения ответа на временном динамическом целевом объекте ответа в течение некоторого фиксированного времени. Этот период времени берется из спецификатора срока действия ответа SCA, если он задан, в противном случае по умолчанию применяется значение 60 секунд. Если время ожидания превышается, то импорт выбрасывает `ServiceTimeoutRuntimeException`.

Внешние клиенты:

Сервер может обмениваться сообщениями с внешними клиентами через привязки JMS.

Внешний клиент (такой как веб-портал или информационная система предприятия) должен отправить сообщение модулю SCA на сервере, или же его необходимо вызвать в компоненте на сервере.

Компоненты экспорта JMS разворачивают получатели сообщений для отслеживания запросов, адресованных целевому получателю, указанному в привязке экспорта. Если вызванное приложение предоставило ответ, то он возвращается получателю, указанному в поле отправки. Благодаря этому внешний клиент может вызывать приложения с привязкой экспорта.

Импорты JMS взаимодействуют с внешними клиентами, отправляя сообщения в очереди JMS или получая сообщения.

Работа со внешними клиентами:

Внешнему клиенту (расположенному вне сервера) может потребоваться взаимодействие с приложением, установленным на сервере.

Рассмотрим очень простой сценарий, в котором внешнему клиенту требуется взаимодействовать с приложением на сервере. На изображении показан обычный простой сценарий.

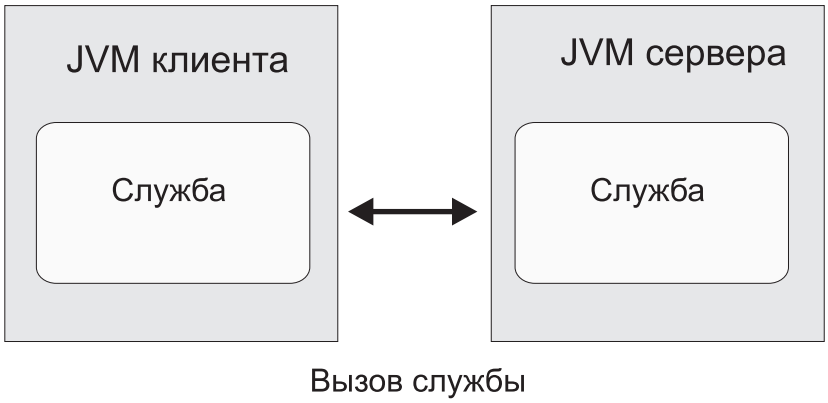


Рисунок 34. Простой сценарий варианта использования: внешний клиент взаимодействует с приложением сервера

Приложение SCA содержит экспорт с привязкой JMS. Таким образом приложение становится доступным для внешних клиентов.

Если внешний клиент расположен на виртуальной машине Java (JVM), которая расположена за пределами сервера, то для установки соединения и обеспечения взаимодействия с экспорт JMS необходимо выполнить некоторые действия. Клиент получает InitialContext с правильными значениями и ищет ресурсы с помощью JNDI. Затем клиент использует клиент спецификации JMS 1.1 для доступа к целевым расположениям и сообщениям, отправляемым и получаемым ими.

Стандартные имена JNDI ресурсов, созданные автоматически средой выполнения, приведены в подразделе о конфигурации в данном разделе. Однако если ресурсы созданы заранее, используйте их имена JNDI.

1. Настройте целевые расположения JMS и фабрику соединений для отправки сообщения.
2. Убедитесь, что контекст JNDI, порт адаптера ресурсов SIB и порт начальной загрузки модуля обмена сообщениями указаны без ошибок.

   Сервер использует некоторые стандартные порты, но если в системе установлено несколько серверов, то во время установки создаются альтернативные порты для того чтобы избежать конфликтов с другими экземплярами сервера. Для того чтобы определить, какие порты использует сервер, можно использовать административную консоль. Перейдите на страницу **Серверы > Серверы приложений >** ***сервер*** **> Конфигурация** и выберите **Порты** в разделе **Связь**. Теперь используемый порт можно изменить.
3. Клиент получает исходный контекст с правильными значениями и ищет ресурсы с помощью JNDI.
4. С помощью спецификаций JMS 1.1 клиент получает доступ к целевым расположениям и к сообщениям, отправляемым и получаемым ими.

Устранение неполадок привязок JMS:

Неполадки привязок JMS можно диагностировать и устранять.

Исключительные ситуации реализации

В ответ на различные ошибки реализация импорта и экспорта JMS может возвратить исключительные ситуации одного из двух типов:

- исключительная ситуация бизнес-службы: эта исключительная ситуация выбрасывается, если происходит сбой, указанный в бизнес-интерфейсе службы (тип порта WSDL);
- исключительная ситуация рабочей среды: выбрасывается во всех остальных случаях. В большинстве случаев исключительная ситуация cause будет содержать исходную исключительную ситуацию (JMSException).

  Например, импорт ожидает только одно ответное сообщение для каждого сообщения запроса. Если получено несколько ответов или ответ запоздал (время истечения срока ожидания ответа SCA истекло), то выбрасывается исключительная ситуация рабочей среды. Выполняется откат транзакции, и ответное сообщение удаляется из очереди или обрабатывается диспетчером недоставленных событий.

Основные условия сбоя

Основные условия сбоя привязок JMS определяются семантикой транзакций, конфигурацией поставщика JMS или ссылкой на фактическое поведение других компонентов. В число основных условий сбоя входят следующие.

- Сбой установки соединения с поставщиком JMS или целевым расположением.

  Сбой установки соединения с поставщиком JMS для получения сообщений приведет к сбою запуска обработчика событий сообщений. Это событие будет записано в протокол WebSphere Application Server. Постоянные сообщения останутся в целевом расположении до момента их успешного получения (или истечения срока).

  Сбой установки соединения с поставщиком JMS для отправки исходящих сообщений приведет к откату транзакции, которая управляет отправкой.

- Сбой анализа входящего сообщения или создания исходящего сообщения.

  Сбой связывания данных или обработчика данных приводит к откату транзакции, которая управляет работой.
- Сбой отправки исходящего сообщения.

  Сбой отправки исходящего сообщения приводит к откату соответствующей транзакции.
- Получение нескольких ответных сообщений или непредвиденное запаздывание ответного сообщения.

  Импорт ожидает только одно ответное сообщение для каждого сообщения запроса. Кроме того, допустимый период времени получения ответа определяется спецификатором Время истечения срока ответа SCA в запросе. В случае получения ответа или истечения срока будет удалена запись об отношении. Если ответные сообщения приходят непредвиденно или запаздывают, то выбрасывается исключительная ситуация рабочей среды.
- Исключительная ситуация рабочей среды в случае тайм-аута службы, которая выбрасывается по причине опоздания ответа при использовании схемы зависимостей назначений временного динамического ответа.

  Тайм-аут импорта JMS произойдет через промежуток времени, определенный спецификатором времени истечения срока ответа SCA. Если он не задан, то будет использовано значение по умолчанию, равное 60 секундам.

Сообщения SCA на основе JMS, которые не выводятся в диспетчере событий с ошибками

В случае сбоя сообщений SCA, полученных по причине сбоя взаимодействия с JMS, ожидается, что они будут показаны в диспетчере недоставленных сообщений. Если эти сообщения не выводятся в диспетчере недоставленных событий, убедитесь, что максимальное число сбоев доставки, заданное для базового назначения SIB назначения JMS, больше **1**. Если указать значение **2** или большее значение, то взаимодействие с диспетчером недоставленных событий во время вызовов SCA для привязок JMS будет возможным.

Обработка исключительных ситуаций:

В конфигурации привязки настраивается способ обработки исключительных ситуаций, сгенерированных обработчиками данных или привязками данных. На действия системы при возникновении такой исключительной ситуации также влияет тип потока передачи.

В обработчике данных или привязке данных, вызванных из привязки, может возникнуть неполадка. Например, обработчик данных может получить сообщение с поврежденными данными или попытаться прочитать сообщение в неверном формате.

Способ обработки таких исключительных ситуаций в привязке зависит от того, каким образом реализован обработчик данных или привязка данных. Рекомендуется, чтобы привязка данных генерировала исключительную ситуацию **DataBindingException**.

При возникновении любой исключительной ситуации времени выполнения, включая **DataBindingException**, выполняются следующие действия:

- Если поток передачи настроен для выполнения транзакций, то по умолчанию в администраторе событий со сбоем сохраняется сообщение JMS , которое можно обработать или удалить вручную.

  Примечание: В привязке можно настроить такой режим исправления, чтобы сообщение не сохранялось в администраторе событий со сбоем, а вместо этого выполнялся откат.
- Если поток передачи не поддерживает транзакции, то исключительная ситуация регистрируется в протоколе, а сообщение удаляется.

В случае обработчика данных ситуация полностью аналогична. Поскольку обработчик данных вызывается из привязки данных, любая исключительная ситуация обработчика данных инкапсулируется в исключительную ситуацию привязки данных. По этой причине об исключительной ситуации **DataHandlerException** сообщается как о **DataBindingException**.

Базовые привязки JMS:

Базовые привязки JMS обеспечивают связь с внешними провайдерами, совместимыми с JMS 1.1. Работа базовых привязок JMS аналогична привязкам JMS.

Служба, предоставляемая через привязку JMS, позволяет модулю архитектуры компонентов служб (SCA) делать вызовы или получать сообщения от внешних систем. Этой системой может быть внешняя система JMS.

Базовая привязка JMS обеспечивает интеграцию с провайдерами JMS, не совместимыми с JCA 1.5, которые поддерживают JMS 1.1 и реализуют необязательный JMS Application Server Facility. Базовая привязка JMS поддерживает таких провайдеров JMS (включая Oracle AQ, TIBCO, SonicMQ, WebMethods и BEA WebLogic), которые не поддерживают JCA 1.5, но поддерживают Application Server Facility спецификации JMS 1.1. Встроенный в WebSphere провайдер JMS (SIBJMS), который является провайдером JCA 1.5 JMS, не поддерживается этой привязкой; в случае использования этого провайдера используйте “Привязки JMS” на стр. 118.

Используйте эту базовую привязку при интеграции с системой на основе JMS, не совместимой с JCA 1.5, в среде SCA. Затем целевые внешние приложения могут получать и отправлять сообщения для интеграции с компонентом SCA.

Базовые привязки JMS – обзор:

Базовые привязки JMS являются привязками JMS не JCA, которые обеспечивают взаимодействие между средой архитектуры компонентов служб (SCA) и системами JMS, которые совместимы с JMS 1.1 и которые реализуют необязательный JMS Application Server Facility.

Общие привязки JMS

Основные аспекты базовых привязок импорта и экспорта JMS включают следующие:

- Порт получателя: позволяет провайдерам JMS не на основе JCA получать и отправлять сообщения управляемому сообщениями EJB (MDB)
- Соединения: охватывают виртуальное соединение между клиентом и приложением провайдера
- Целевые объекты: используются клиентом для определения целевого объекта сообщений, которые он отправляет, или исходного объекта сообщений, которые он получает
- Идентификационные данные: используются для защиты доступа к привязке

Базовые привязки импортов JMS

Базовые привязки импортов JMS позволяют компонентам в пределах модуля SCA связываться со службами, предоставляемыми внешними провайдерами JMS, не совместимыми с JCA 1.5.

Обеспечивающей соединение частью импорта JMS является фабрика соединений. Фабрика соединений, являющаяся объектом, который клиент использует для установления соединения с провайдером, включает набор параметров конфигурации соединений, определяемый администратором. Каждая фабрика соединений является экземпляром интерфейса ConnectionFactory, QueueConnectionFactory или TopicConnectionFactory.

Взаимодействие с внешними системами JMS включает использование целевых объектов для отправки запросов и получения ответов.

Для базовой привязки импорта JMS поддерживаются два типа сценариев использования в зависимости от типа вызываемой операции:

- Односторонний: базовый импорт JMS помещает сообщение на целевой объект отправления, настроенный в привязке импорта. В поле replyTo заголовка JMS не отправляется ничего.

- Двухсторонний (запрос-ответ): Базовый импорт JMS помещает сообщение на целевой объект отправления и затем сохраняет ответ, который он получает из компонента SCA.

  Целевой объект получения задается в свойстве заголовка replyTo исходящего сообщения. Управляемый сообщениями EJB (MDB) развертывается для прослушивания на целевом объекте получения, и по получении ответа MDB передает этот ответ назад компоненту.

  Привязку импорта можно настроить (с помощью поля **Схема зависимости ответа** в Integration Designer) на ожидание ИД зависимости ответного сообщения, скопированного из ИД сообщения-запроса (по умолчанию) или из ИД зависимости сообщения-запроса.

Как для одностороннего, так и двухстороннего сценария могут быть заданы динамические и статические свойства заголовков. Статические свойства можно задать из привязки метода базового импорта JMS. Некоторые из этих свойств имеют специальные значения для выполнения SCA JMS.

Важно отметить, что базовая JMS является асинхронной привязкой. Если вызывающий компонент вызывает базовый импорт JMS синхронно (в двухсторонней операции), то вызывающий компонент блокируется до возвращения ответа службой JMS.

рис. 35 иллюстрирует, как импорт связывается с внешней службой.

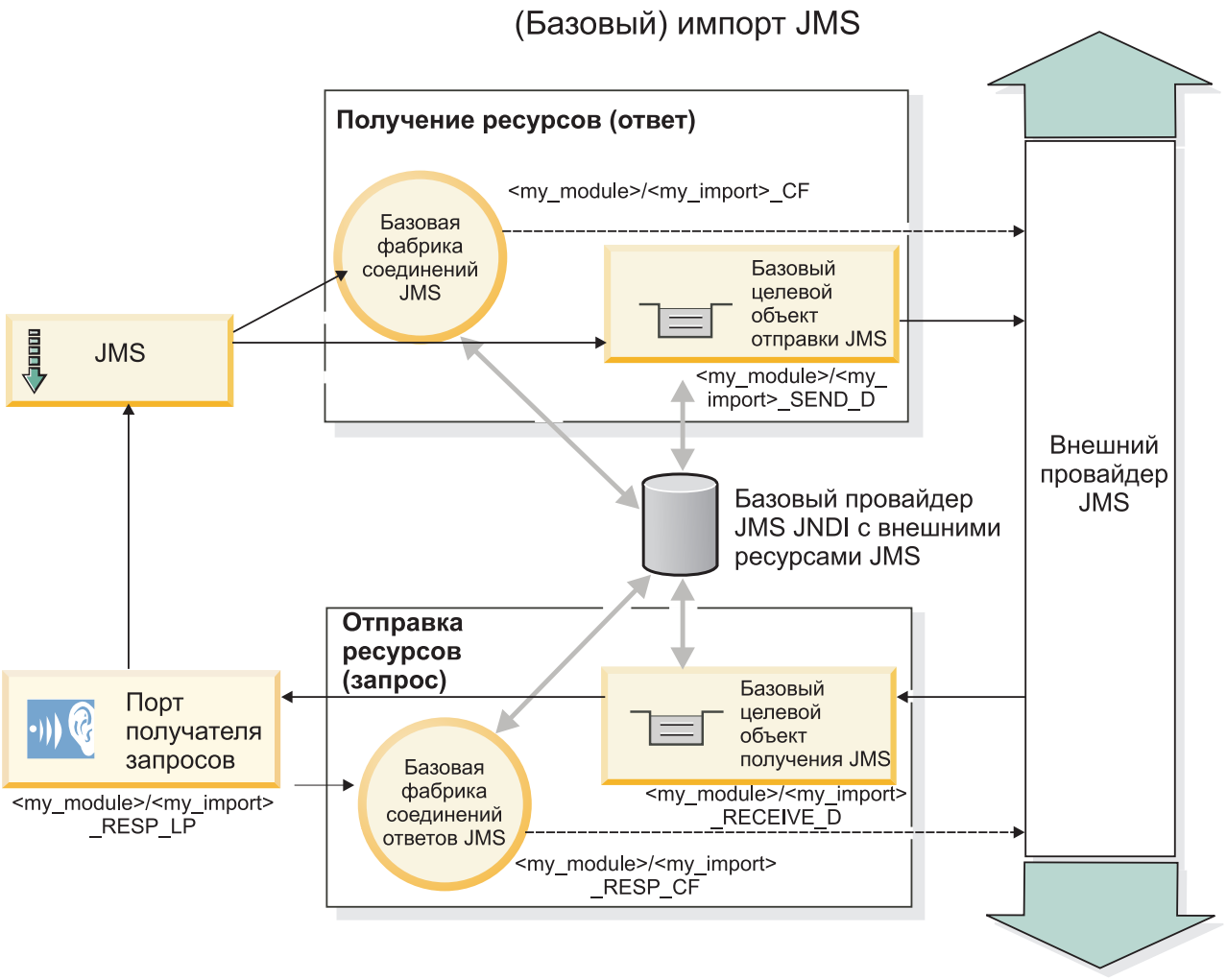


Рисунок 35. Ресурсы привязок базовых импортов JMS

Базовые привязки экспортов JMS

Базовые привязки экспортов JMS обеспечивают средства для модуля SCA по предоставлению служб внешним приложениям JMS.

Обеспечивающей соединение частью экспорта JMS являются ConnectionFactory и ListenerPort.

Общий экспорт JMS имеет целевые объекты отправления и получения.

- Целевой объект получения – это куда должно быть помещено входящее сообщение для целевого компонента.
- Целевой объект отправления – это куда будет отправлен ответ, если входящее сообщение не переопределило его с помощью свойства заголовка replyTo.

Развертывается MDB прослушивания запросов, приходящих на целевой объект получения, указанный в привязке экспорта.

- Целевой объект указывается в поле send и используется для отправки ответа на входящий запрос, если вызванный компонент предоставляет ответ.
- Целевой объект, заданный в поле replyTo входящего сообщения, переопределяет целевой объект, указанный в поле send.
- Для сценариев запрос/ответ привязку импорта можно настроить (с помощью поля **Схема зависимости ответа** в Integration Designer) на ожидание ответа для копирования ИД сообщения-запроса в поле ИД зависимости ответного сообщения (по умолчанию), или ответ может скопировать ИД зависимости запроса в поле ИД зависимости ответного сообщения.

На рисунке рис. 36 на стр. 130 показано, как внешний инициатор запроса связывается с экспортом.

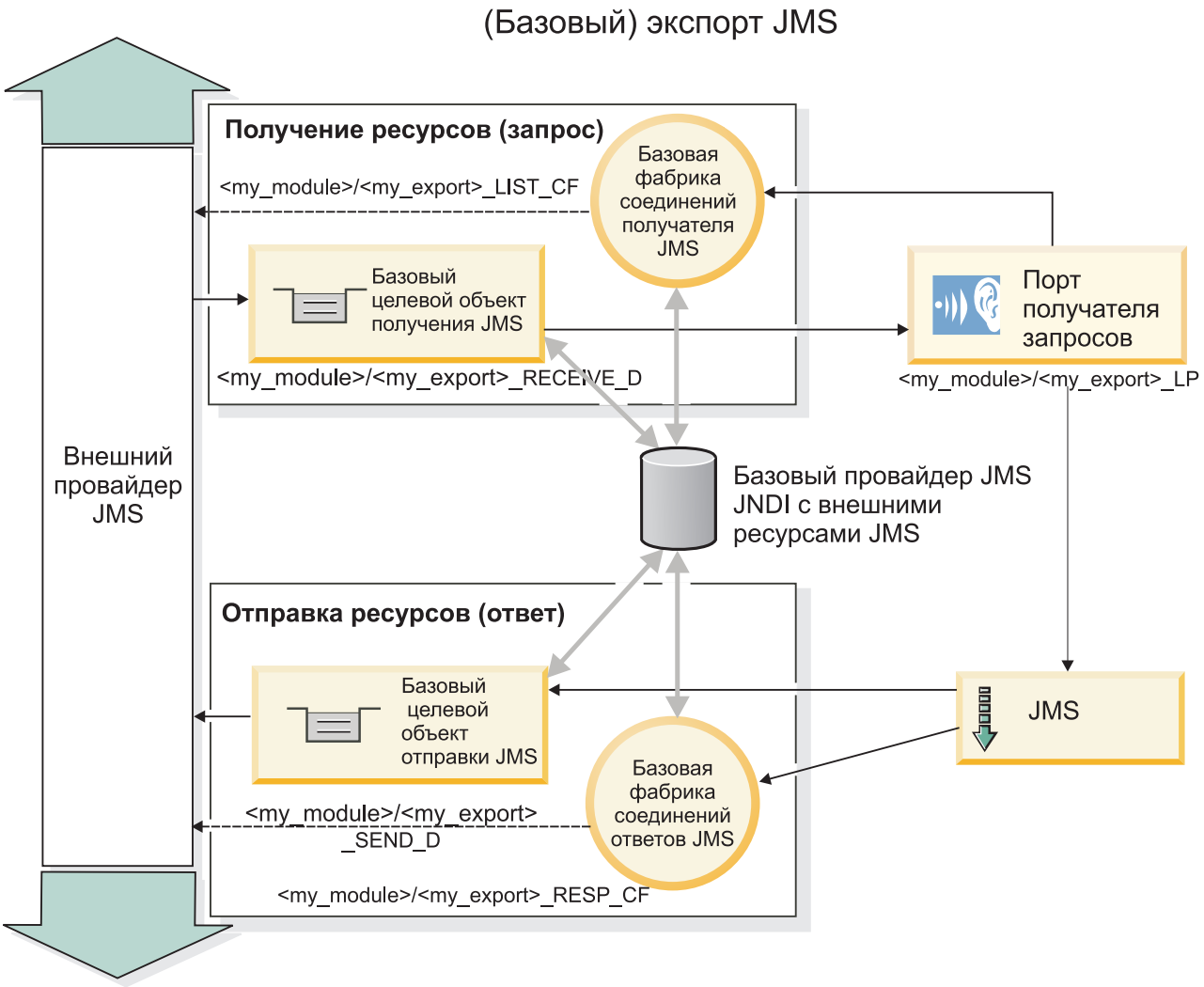


Рисунок 36. Ресурсы базовых привязок экспортов JMS

Основные компоненты привязок Generic JMS:

Компоненты привязки импорта и экспорта Generic JMS соответствуют компонентам встроенных привязок импортов WebSphere JMS и MQ JMS. Основные компоненты включают определения заголовков и доступ к существующим ресурсам Java EE. Однако, ввиду их общего характера, опции соединения с конкретным провайдером JMS отсутствуют, и способность этой привязки создавать ресурсы во время развертывания и установки ограничена.

Базовые импорты

Аналогично приложению импорта MQ JMS, реализация Общая JMS асинхронна и поддерживает три вызова: односторонний, двухсторонний (также называемый запрос-ответ) и обратный вызов.

При развертывании импорта JMS также развертывается управляемый сообщениями EJB (MDB), предоставляемый средой выполнения. MDB прослушивает ответы на сообщение-запрос. MDB связывается (прослушивает) назначением, передаваемым с запросом в поле заголовка replyTo сообщения JMS.

Базовые экспорты

Привязки базовых экспортов JMS отличаются от привязок экспортов EIS обработкой возврата результата. Базовый экспорт JMS явно отправляет ответ в назначение replyTo, указанное во входящем сообщении. Если оно не указано, то используется назначение отправления.

При развертывании базового экспорта JMS также развертывается управляемый сообщениями EJB (MDB, отличающийся от используемого для базовых импортов JMS). Он прослушивает входящие запросы в назначении получения и затем направляет запросы на обработку средой выполнения SCA.

Специальные заголовки

Свойства специальных заголовков используются в базовых импортах и экспортах JMS для информирования целевой привязки о том, как обрабатывать сообщение.

Например, свойство TargetFunctionName используется селектором функций по умолчанию для определения имени операции, вызываемой в интерфейсе экспорта.

Примечание: Привязка импорта может быть настроена на задание имени операции в заголовке TargetFunctionName для каждой операции.

Ресурсы EE

При развертывании привязки JMS в среду Java EE создаются многие ресурсы Java EE.

- Порт получателя для прослушивания назначения получения (ответа) (только при двусторонней связи) для импортов и назначения получения (запроса) для экспортов
- Базовая фабрика соединений JMS для outboundConnection (импорт) и inboundConnection (экспорт)
- Базовое назначение JMS для назначений отправления (импорт) и получения (экспорт, только для двусторонней связи)
- Базовая фабрика соединений JMS для responseConnection (только для двусторонней связи и необязательно; в противном случае outboundConnection используется для импортов, а inboundConnection – для экспортов)
- Базовое назначение JMS для назначения получения (импорт) и отправления (экспорт) (только для двухсторонней связи)
- Назначение JMS обратных вызовов провайдера обмена сообщениями по умолчанию, используемое для доступа к назначению очереди SIB (только для двухсторонней связи)
- Фабрика соединений JMS обратных вызовов провайдера обмена сообщениями по умолчанию, используемая для доступа к назначению JMS обратных вызовов (только для двухсторонней связи)
- Назначение обратных вызовов SIB, используемое для хранения информации о сообщении-запросе для использования во время обработки ответов (только для двухсторонней связи)

Задача установки создает ConnectionFactory, три назначения и ActivationSpec из информации в файлах импорта и экспорта.

Базовые заголовки JMS:

Базовые заголовки JMS - это объекты данных службы (SDO), содержащие все свойства сообщения базового JMS. Это могут быть свойства из входящего сообщения или свойства, применяемые к исходящему сообщению.

Сообщение JMS содержит заголовки двух типов: системный заголовок JMS и несколько свойств JMS. Заголовки обоих типов можно получить из модуля передачи, расположенного в объекте сообщения службы (SMO), или с помощью API ContextService.

Следующие свойства статически настраиваются в methodBinding:

- JMSType
- JMSCorrelationID

- JMSDeliveryMode
- JMSPriority

Привязка базового JMS поддерживает динамическое изменение заголовков и свойств JMS, как и другие привязки JMS и JMS MQ.

Некоторые поставщики базового JMS ограничивают набор свойств, которые разрешено настраивать в приложении, а также возможные сочетания этих свойств. За дополнительной информацией обратитесь к документации по продукту другого поставщика. В methodBinding было добавлено свойство ignoreInvalidOutboundJMSProperties, позволяющее передавать информацию о любых исключительных ситуациях.

Заголовки и свойства сообщения базового JMS используются только в том случае, если включен базовый коммутатор привязок SCDL-SCA. Если этот коммутатор включен, то передается информация о контексте. По умолчанию этот коммутатор включен. Для того чтобы запретить передачу информации о контексте, измените значение на **false**.

Если передача контекста разрешена, то информацию из заголовков разрешается передавать в сообщение или в целевой компонент. Для включения или выключения передачи информации о контексте укажите значение **true** или **false** в атрибуте contextPropagationEnabled привязки импорта или экспорта. Например:

```
<esbBinding xsi:type="eis:JMSImportBinding" contextProgagationEnabled="true">
```

Значение по умолчанию равно **true**.

Устранение неполадок привязок Generic JMS:

Неполадки привязок Generic JMS можно диагностировать и устранять.

Исключительные ситуации реализации

В ответ на различные ошибки реализация импорта и экспорта Generic JMS может возвратить исключительные ситуации одного из двух типов:

- исключительная ситуация бизнес-службы: эта исключительная ситуация выбрасывается, если происходит сбой, указанный в бизнес-интерфейсе службы (тип порта WSDL);
- исключительная ситуация рабочей среды: выбрасывается во всех остальных случаях. В большинстве случаев исключительная ситуация cause будет содержать исходную исключительную ситуацию (JMSException).

Устранение неполадок истечения срока сообщений Generic JMS

Сообщение запроса от поставщика JMS имеет время истечения срока.

Время истечения срока запроса означает, что поставщик JMS помечает сообщение запроса как устаревшее по прошествии времени, указанного в JMSExpiration сообщения запроса. Как и другие привязки JMS, привязка Generic JMS обрабатывает истечение срока запроса, указав в сообщении обратного вызова, которое отправлено импортом, такое же время истечения срока, как и в исходящем запросе. Уведомление об истечении срока сообщения обратного вызова указывает, что сообщение запроса устарело и клиент должен быть оповещен путем выбрасывания исключительной ситуации бизнес-службы.

Однако если целевое расположение обратного вызова перенесено на поставщик другой фирмы, этот тип истечения срока запроса не поддерживается.

Время истечения срока ответа означает, что поставщик JMS помечает сообщение ответа как устаревшее по прошествии времени, указанного в JMSExpiration сообщения ответа.

Истечение срока ответа для общей привязки JMS не поддерживается, потому что точное поведение поставщика JMS другой фирмы при истечении срока является неопределенным. Однако можно проверить, не истек ли срок ответа, в случае и во время его получения.

Для исходящих сообщений значение JMSExpiration вычисляется по времени ожидания и по значениям requestExpiration, если они заданы в asyncHeader.

Устранение неполадок фабрики соединений Generic JMS

Когда в поставщике Generic JMS определяются определенные типы фабрик соединений, во время попытки запуска приложения может быть выведено сообщение об ошибке. Для того чтобы избежать этой неполадки, можно изменить внешнюю фабрику соединений.

Во время запуска приложения может быть выведено следующее сообщение об ошибке:

```
JMSConnectionFactory порта получения запросов MDB не совпадает
с типом JMSDestination
```

Эта неполадка может возникнуть во время определения внешних фабрик соединения. В частности, исключительная ситуация может быть выброшена в случае создания JMS 1.0.2 Topic Connection Factory вместо JMS 1.1 (unified) Connection Factory (то есть, той, которая способна поддерживать связь как в двухточечном режиме, так и в режиме публикации-подписки).

Для устранения этой неполадки выполните следующие действия:

1. откройте поставщик Generic JMS, который используете.
2. замените указанную JMS 1.0.2 Topic Connection Factory на JMS 1.1 (unified) Connection Factory.

При запуске приложения с указанной JMS 1.1 Connection Factory сообщение об ошибке больше выводиться не будет.

Сообщения SCA на основе Generic JMS, которые не выводятся в диспетчере событий с ошибками

В случае сбоя сообщений SCA, полученных по причине сбоя взаимодействия с общей JMS, ожидается, что они будут показаны в диспетчере недоставленных сообщений. Если эти сообщения не выводятся в диспетчере недоставленных событий, убедитесь, что значение свойства максимального числа попыток базового порта получения запросов больше 1. Если указать значение 1 или большее значение, то взаимодействие с диспетчером недоставленных событий во время вызовов SCA для привязок общей JMS будет возможным.

Обработка исключительных ситуаций:

В конфигурации привязки настраивается способ обработки исключительных ситуаций, сгенерированных обработчиками данных или привязками данных. На действия системы при возникновении такой исключительной ситуации также влияет тип потока передачи.

В обработчике данных или привязке данных, вызванных из привязки, может возникнуть неполадка. Например, обработчик данных может получить сообщение с поврежденными данными или попытаться прочитать сообщение в неверном формате.

Способ обработки таких исключительных ситуаций в привязке зависит от того, каким образом реализован обработчик данных или привязка данных. Рекомендуется, чтобы привязка данных генерировала исключительную ситуацию **DataBindingException**.

В случае обработчика данных ситуация полностью аналогична. Поскольку обработчик данных вызывается из привязки данных, любая исключительная ситуация обработчика данных инкапсулируется в исключительную ситуацию привязки данных. По этой причине об исключительной ситуации **DataHandlerException** сообщается как о **DataBindingException**.

При возникновении любой исключительной ситуации времени выполнения, включая **DataBindingException**, выполняются следующие действия:

- Если поток передачи настроен для выполнения транзакций, то по умолчанию в администраторе событий со сбоем сохраняется сообщение JMS, которое можно обработать или удалить вручную.

  Примечание: В привязке можно настроить такой режим исправления, чтобы сообщение не сохранялось в администраторе событий со сбоем, а вместо этого выполнялся откат.
- Если поток передачи не поддерживает транзакции, то исключительная ситуация регистрируется в протоколе, а сообщение удаляется.

В случае обработчика данных ситуация полностью аналогична. Поскольку обработчик данных вызывается из привязки данных, любая исключительная ситуация обработчика данных инкапсулируется в исключительную ситуацию привязки данных. По этой причине об исключительной ситуации **DataHandlerException** сообщается как о **DataBindingException**.

Привязки WebSphere MQ JMS:

Привязка WebSphere MQ JMS обеспечивает интеграцию с внешними приложениями, которые используют провайдера WebSphere MQ на основе JMS.

Привязки экспортов и импортов WebSphere MQ JMS используются непосредственно для интеграции с внешними системами JMS или MQ JMS из вашей серверной среды. Это исключает необходимость в использовании компонента MQ Link или Client Link шины интеграции служб.

Когда компонент взаимодействует со службой WebSphere MQ на основе JMS через импорт, привязка импорта WebSphere MQ JMS использует целевой объект, которому будут отправляться данные, и целевой объект, на котором может быть получен ответ. Преобразование данных в и из сообщения JMS осуществляется через Edge Component привязки обработчика данных или данных JMS.

Когда модуль SCA предоставляет службу клиентам WebSphere MQ JMS, привязка экспорта WebSphere MQ JMS использует целевой объект, на котором может быть получен запрос и которому может быть отправлен ответ. Преобразование данных в и из сообщения JMS осуществляется через привязку обработчика данных или данных JMS.

Селектор функций обеспечивает преобразование в операцию, вызываемую на целевом компоненте.

Привязки WebSphere MQ JMS – обзор:

Привязка WebSphere MQ JMS обеспечивает интеграцию с внешними приложениями, использующими провайдер WebSphere MQ JMS.

Административные задачи WebSphere MQ

Предполагается, что системный администратор WebSphere MQ создаст основной WebSphere MQ Queue Manager, который будут использовать привязки WebSphere MQ JMS, перед выполнением приложения, содержащего эти привязки.

Привязки импортов WebSphere MQ JMS

Импорт WebSphere MQ JMS позволяет компонентам в пределах вашего модуля SCA соединяться со службами, предоставляемыми провайдерами WebSphere MQ на основе JMS. Необходимо использовать поддерживаемую версию WebSphere MQ. Подробные требования к аппаратному и программному обеспечению можно найти на страницах поддержки IBM.

Для привязок импортов WebSphere MQ JMS поддерживаются два типа сценариев использования в зависимости от типа вызываемой операции:

- Односторонний: Импорт WebSphere MQ JMS помещает сообщение на целевой объект отправления, настроенный в привязке импорта. В поле replyTo заголовка JMS не отправляется ничего.
- Двухсторонний (запрос-ответ): Импорт WebSphere MQ JMS помещает сообщение на целевой объект отправления.

  Целевой объект получения задается в поле заголовка replyTo. Управляемый сообщениями EJB (MDB) развертывается для прослушивания на целевом объекте получения, и по получении ответа MDB передает этот ответ назад компоненту.

  Привязку импорта можно настроить (с помощью поля **Схема зависимости ответа** в Integration Designer) на ожидание ИД зависимости ответного сообщения, скопированного из ИД сообщения-запроса (по умолчанию) или из ИД зависимости сообщения-запроса.

Как для одностороннего, так и двухстороннего сценария могут быть заданы динамические и статические свойства заголовков. Статические свойства можно задать из привязки метода импорта JMS. Некоторые из этих свойств имеют специальные значения для выполнения SCA JMS.

Важно отметить, что WebSphere MQ JMS является асинхронной привязкой. Если вызывающий компонент вызывает импорт WebSphere MQ JMS синхронно (в двухсторонней операции), то вызывающий компонент блокируется до возвращения ответа службой JMS.

рис. 37 иллюстрирует, как импорт связывается с внешней службой.

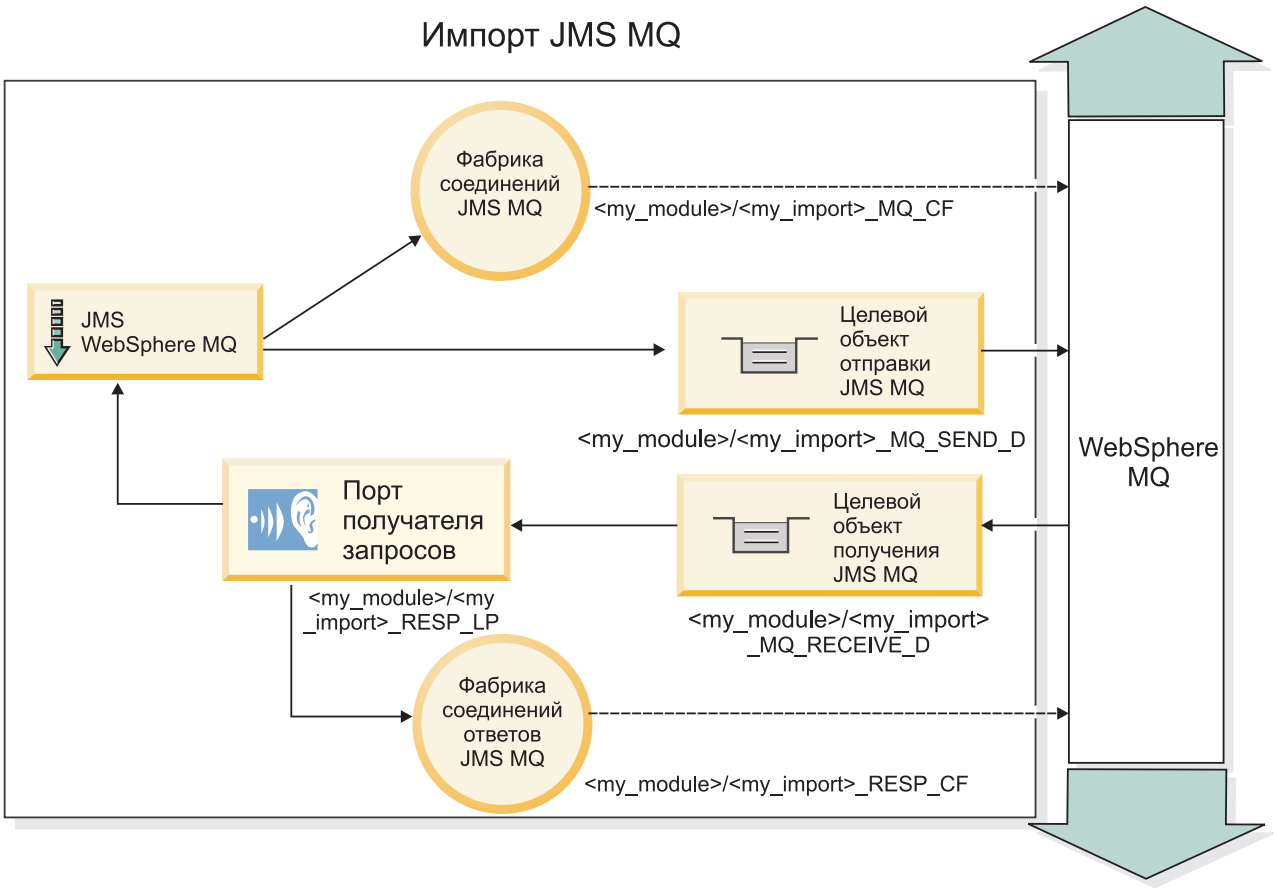


Рисунок 37. Ресурсы привязки импорта WebSphere MQ JMS

Привязки экспортов WebSphere MQ JMS

Привязка экспорта WebSphere MQ JMS обеспечивает средства для модуля SCA по предоставлению служб внешним приложениям JMS на провайдере WebSphere MQ на основе JMS.

Развертывается MDB прослушивания запросов, приходящих на целевой объект получения, указанный в привязке экспорта. Целевой объект, заданный в поле send, используется для отправки ответа на входящий запрос, если вызванный компонент предоставляет ответ. Целевой объект, заданный в поле replyTo ответного сообщения, переопределяет целевой объект, указанный в поле send.

На рисунке рис. 38 показано, как внешний инициатор запроса связывается с экспортом.

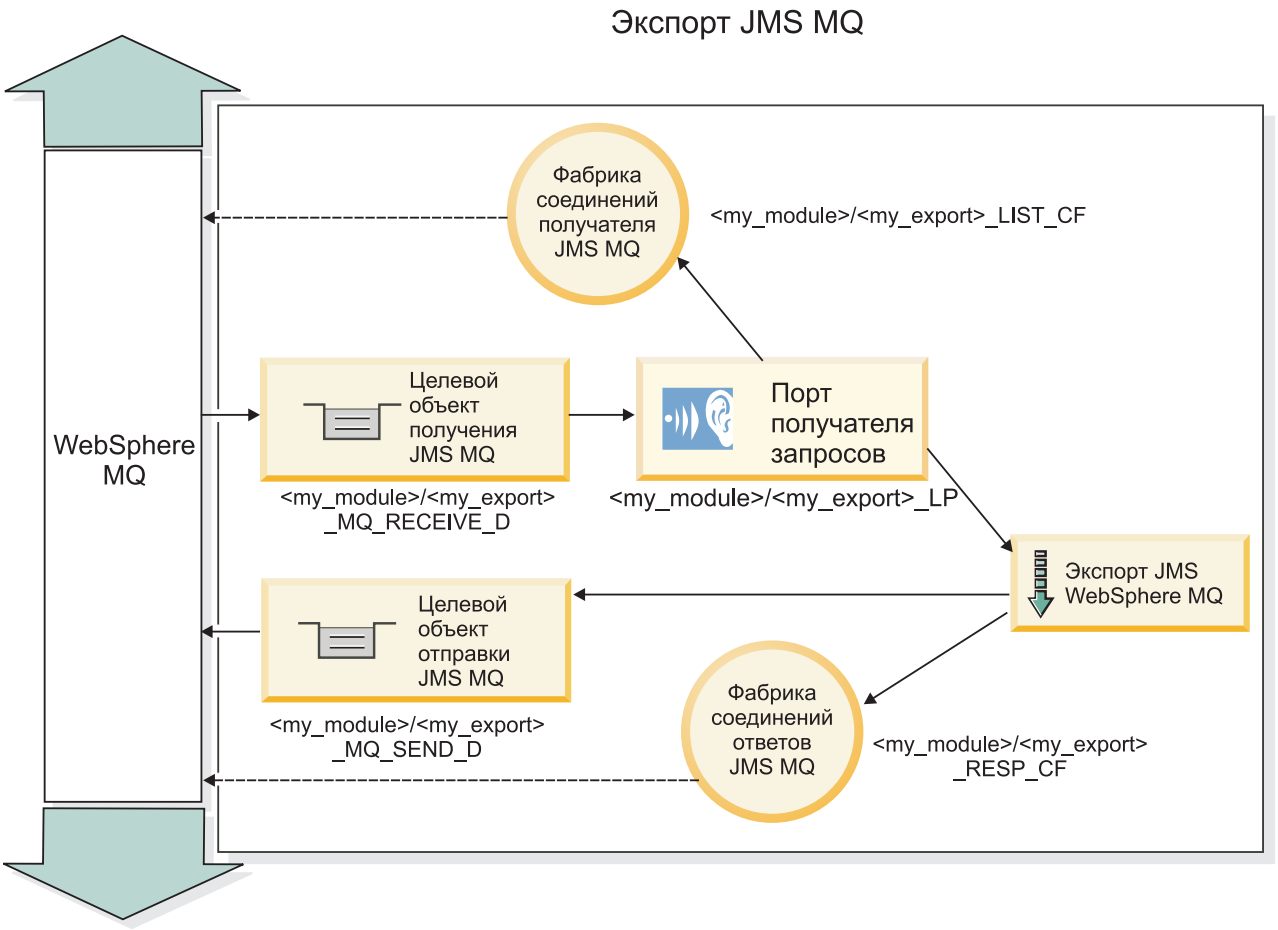


Рисунок 38. Ресурсы привязки экспорта WebSphere MQ JMS

Примечание: рис. 37 на стр. 135 и рис. 38 иллюстрируют, как приложение из предыдущей версии IBM Business Process Manager связывается с внешней службой. В случае приложений, развернутых для IBM Business Process Manager версии 7.0, используется спецификация активации вместо порта получателя и фабрики соединений.

Основные компоненты привязок WebSphere MQ JMS:

Основные компоненты привязок WebSphere MQ JMS включают заголовки, артефакты Java EE и созданные ресурсы Java EE.

Заголовки

Заголовок сообщения JMS содержит несколько уже заданных полей, содержащих значения, используемые как клиентами, так и провайдерами для идентификации и маршрутизации сообщений. Свойства привязок можно использовать для настройки этих заголовков с использованием постоянных значений, а также заголовки можно определить динамически во время выполнения.

JMSCorrelationID
Указывает на связанное сообщение. Обычно в этом поле задается строка идентификатора сообщения, на которое требуется ответ.

TargetFunctionName
Этот заголовок используется одним из предоставленных селекторов функций для идентификации вызываемой операции. Задание свойства заголовка TargetFunctionName JMS в сообщениях, отправляемых в экспорт JMS, позволяет использовать этот селектор функций. Свойство может быть задано непосредственно в приложениях-клиентах JMS или при соединении импорта с привязкой JMS с таким экспортом. В этом случае привязка импорта JMS должна быть настроена на задание заголовка TargetFunctionName для каждой операции в интерфейсе на имя операции.

Схемы зависимостей

Привязки WebSphere MQ JMS предусматривают разные схемы зависимостей, которые используются для определения того, как сообщения-запросы связаны с ответными сообщениями.

RequestMsgIDToCorrelID
JMSMessageID копируется в поле JMSCorrelationID. Это значение по умолчанию.

RequestCorrelIDToCorrelID
JMSCorrelationID копируется в поле JMSCorrelationID.

Ресурсы EE

При развертывании импорта MQ JMS в среду Java EE создаются многие ресурсы Java EE.

Параметры

Фабрика соединений MQ
Используется клиентами для соединения с провайдером MQ JMS.

Фабрика соединений ответов
Используется средой выполнения SCA MQ JMS, когда назначения отправления и получения относятся к разным администраторам очередей.

Спецификация активации
Спецификация активации MQ JMS связывается с одним или несколькими управляемыми сообщениями EJB и обеспечивает конфигурацию, необходимую им для получения сообщений.

Назначения

- Назначение отправления:
  - Импорты: Куда отправляется запрос или исходящее сообщение.
  - Экспорты: Куда будет отправлено ответное сообщение в отсутствие замещающего поля заголовка JMSReplyTo входящего сообщения.
- Назначение получения:
  - Импорты: Куда должен быть размещен ответ или входящее сообщение.
  - Экспорты: Куда должно быть размещено входящее сообщение или запрос.

Заголовки JMS:

Сообщение JMS содержит заголовки двух типов: системный заголовок JMS и несколько свойств JMS. Оба типа заголовков могут быть доступны либо в модуле передачи объекта сообщения службы (SMO), либо с помощью API ContextService.

Системный заголовок JMS

Системный заголовок JMS представляется в SMO элементом JMSHeader, который содержит все поля, типичные для заголовка JMS. Несмотря на то, что они могут изменяться в передаче (или ContextService), некоторые поля системного заголовка JMS, заданные в SMO, не будут передаваться в исходящем сообщении JMS, поскольку они переопределяются системными или статическими значениями.

Основные поля в системном заголовке JMS, которые могут обновляться в передаче (или ContextService), следующие:

- **JMSType** и **JMSCorrelationID** - значения конкретных уже заданных свойств заголовка сообщения
- **JMSDeliveryMode** - значения для режима доставки (постоянные или временные; по умолчанию - постоянные)
- **JMSPriority** - значение приоритета (от 0 до 9; по умолчанию применяется приоритет-JMS-по-умолчанию)

Свойства JMS

Свойства JMS представляются в SMO как записи в списке Свойства. Эти свойства можно добавлять, обновлять или удалять в передаче или с помощью API ContextService.

Свойства также можно задавать статически в привязке JMS. Свойства, которые заданы статически, переопределяют значения (с тем же именем), которые задаются динамически.

Пользовательские свойства, передаваемые из других привязок (например, привязки HTTP), будут выводиться в привязке JMS как свойства JMS.

Параметры распространения заголовков

Распространением системного заголовка и свойств JMS либо из входящего сообщения JMS в последующие компоненты, либо из предыдущих компонентов в исходящее сообщение JMS можно управлять с помощью флага Распространять заголовок протокола на привязке.

Когда флаг Распространять заголовок протокола включен, информация заголовка может передаваться сообщению или целевому компоненту согласно следующему списку:

- Запрос экспорта JMS

  Заголовок JMS, полученный в сообщении, будет распространяться целевым компонентам при помощи контекстной службы. Свойства JMS, полученные в сообщении, будут распространяться целевым компонентам при помощи контекстной службы.
- Ответ экспорта JMS

  Все поля заголовка JMS, заданные в контекстной службе, будут использоваться в исходящем сообщении, если они не переопределяются статическими свойствами, заданными в привязке экспорта JMS. Все свойства, заданные в контекстной службе, будут использоваться в исходящем сообщении, если они не переопределяются статическими свойствами, заданными в привязке экспорта JMS.
- Запрос импорта JMS

  Все поля заголовка JMS, заданные в контекстной службе, будут использоваться в исходящем сообщении, если они не переопределяются статическими свойствами, заданными в привязке импорта JMS. Все свойства, заданные в контекстной службе, будут использоваться в исходящем сообщении, если они не переопределяются статическими свойствами, заданными в привязке импорта JMS.

- Ответ импорта JMS

  Заголовок JMS, полученный в сообщении, будет распространяться целевым компонентам при помощи контекстной службы. Свойства JMS, полученные в сообщении, будут распространяться целевым компонентам при помощи контекстной службы.

Внешние клиенты:

Сервер может отправлять сообщения или принимать сообщения от внешних клиентов с помощью привязок WebSphere MQ JMS.

Внешний клиент (например, веб-портал или информационная система организации) может отправить сообщение компоненту SCA в приложении при помощи экспорта или он может быть вызван компонентом SCA в приложении при помощи импорта.

Привязка экспорта WebSphere MQ JMS развертывает управляемые сообщениями EJB (MDB) для прослушивания запросов, приходящих на целевой объект получения, указанный в привязке экспорта. Целевой объект указывается в поле send и используется для отправки ответа на входящий запрос, если вызванное приложение предоставляет ответ. Таким образом, внешний клиент может вызывать приложения через привязку экспорта.

Импорты WebSphere MQ JMS привязываются к внешним клиентам и могут доставлять им сообщения. Эти сообщения могут требовать или могут не требовать ответа от внешнего клиента.

Дополнительную информацию о взаимодействии с внешними клиентами с помощью WebSphere MQ можно найти в справочной системе WebSphere MQ Information Center.

Устранение неполадок привязок WebSphere MQ JMS:

Неполадки привязок WebSphere MQ JMS можно диагностировать и устранять.

Исключительные ситуации реализации

В ответ на различные ошибки реализация импорта и экспорта MQ JMS может возвратить исключительные ситуации одного из двух типов:

- исключительная ситуация бизнес-службы: эта исключительная ситуация выбрасывается, если происходит сбой, указанный в бизнес-интерфейсе службы (тип порта WSDL);
- исключительная ситуация рабочей среды: выбрасывается во всех остальных случаях. В большинстве случаев исключительная ситуация cause будет содержать исходную исключительную ситуацию (JMSException).

  Например, импорт ожидает только одно ответное сообщение для каждого сообщения запроса. Если получено несколько ответов или ответ запоздал (время истечения срока ожидания ответа SCA истекло), то выбрасывается исключительная ситуация рабочей среды. Выполняется откат транзакции, и ответное сообщение удаляется из очереди или обрабатывается диспетчером недоставленных событий.

Сообщения SCA на основе WebSphere MQ JMS, которые не выводятся в диспетчере событий с ошибками

В случае сбоя сообщений SCA, полученных при взаимодействии с WebSphere MQ JMS, ожидается, что они будут показаны в диспетчере недоставленных сообщений. Если эти сообщения не выводятся в диспетчере недоставленных событий, убедитесь, что значение свойства максимального числа попыток базового порта получения запросов больше **1**. Если указать значение **1** или большее значение, то взаимодействие с диспетчером недоставленных событий во время вызовов SCA для привязок MQ JMS будет возможным.

Сценарии неправильного употребления: сравнение с привязками WebSphere MQ

Привязка WebSphere MQ JMS предназначена для взаимодействия с приложениями JMS, развернутых на WebSphere MQ, которая предоставляет сообщения в соответствии с моделью сообщений JMS. Однако импорт и экспорт WebSphere MQ изначально предназначены для взаимодействия со стандартными приложениями WebSphere MQ и предоставляют передачам доступ ко всему содержимому тела сообщения WebSphere MQ.

Следующие сценарии следует компоновать с использованием привязки WebSphere MQ JMS, а не WebSphere MQ.

- Вызов объекта EJB, управляемого сообщениями JMS (MDB) из модуля SCA, если MDB развернут на поставщике WebSphere MQ JMS. Используйте импорт WebSphere MQ JMS.
- Разрешение вызова модуля SCA из сервлета компонента Java EE или EJB с использованием JMS. Используйте экспорт WebSphere MQ JMS.
- Транзитная передача содержимого JMS MapMessage через WebSphere MQ. Используйте экспорт и импорт WebSphere MQ в комбинации с соответствующим обработчиком или привязкой данных.

Существуют ситуации, в которых ожидается взаимодействие привязки WebSphere MQ с привязкой WebSphere MQ JMS. В частности, в случае стыковки приложений WebSphere MQ, использующих Java EE, с приложениями, не использующими Java EE, используйте экспорт WebSphere MQ и импорт WebSphere MQ JMS (или наоборот) в комбинации с соответствующими привязками данных или модулями передачи как по отдельности, так и совместно.

Обработка исключительных ситуаций:

В конфигурации привязки настраивается способ обработки исключительных ситуаций, сгенерированных обработчиками данных или привязками данных. На действия системы при возникновении такой исключительной ситуации также влияет тип потока передачи.

В обработчике данных или привязке данных, вызванных из привязки, может возникнуть неполадка. Например, обработчик данных может получить сообщение с поврежденными данными или попытаться прочитать сообщение в неверном формате.

Способ обработки таких исключительных ситуаций в привязке зависит от того, каким образом реализован обработчик данных или привязка данных. Рекомендуется, чтобы привязка данных генерировала исключительную ситуацию **DataBindingException**.

В случае обработчика данных ситуация полностью аналогична. Поскольку обработчик данных вызывается из привязки данных, любая исключительная ситуация обработчика данных инкапсулируется в исключительную ситуацию привязки данных. По этой причине об исключительной ситуации **DataHandlerException** сообщается как о **DataBindingException**.

При возникновении любой исключительной ситуации времени выполнения, включая **DataBindingException**, выполняются следующие действия:

- Если поток передачи настроен для выполнения транзакций, то по умолчанию в администраторе событий со сбоем сохраняется сообщение JMS, которое можно обработать или удалить вручную.

  Примечание: В привязке можно настроить такой режим исправления, чтобы сообщение не сохранялось в администраторе событий со сбоем, а вместо этого выполнялся откат.
- Если поток передачи не поддерживает транзакции, то исключительная ситуация регистрируется в протоколе, а сообщение удаляется.

В случае обработчика данных ситуация полностью аналогична. Поскольку обработчик данных вызывается из привязки данных, любая исключительная ситуация обработчика данных инкапсулируется в

исключительную ситуацию привязки данных. По этой причине об исключительной ситуации **DataHandlerException** сообщается как о **DataBindingException**.

Привязки WebSphere MQ:

Привязка WebSphere MQ обеспечивает возможность соединения архитектуры компонентов служб (SCA) с приложениями WebSphere MQ.

Привязки экспортов и импортов WebSphere MQ JMS используются непосредственно для интеграции с системой на основе WebSphere MQ из вашей серверной среды. Это исключает необходимость в использовании компонента MQ Link или Client Link шины интеграции служб.

Когда компонент взаимодействует со службой WebSphere MQ через импорт, привязка импорта WebSphere MQ использует очередь, в которую будут отправляться данные, и очередь, из которой может быть получен ответ.

Когда модуль SCA предоставляет службу клиентам WebSphere MQ, привязка экспорта WebSphere MQ использует очередь, из которой может быть получен запрос и в которую может быть отправлен ответ. Селектор функций обеспечивает преобразование в операцию, вызываемую на целевом компоненте.

Преобразование данных полезной нагрузки в и из сообщения MQ выполняется через обработчик данных тела MQ или привязку данных. Преобразование данных заголовка в и из сообщения MQ выполняется через привязку данных заголовка MQ.

Информация о поддерживаемых версиях WebSphere MQ приведена на веб-странице подробных системных требований.

Привязки WebSphere MQ – обзор:

Привязка WebSphere MQ обеспечивает интеграцию со стандартными приложениями на основе MQ.

Административные задачи WebSphere MQ

Предполагается, что системный администратор WebSphere MQ создаст основной WebSphere MQ Queue Manager, который будут использовать привязки WebSphere MQ, перед выполнением приложения, содержащего эти привязки.

Административные задачи WebSphere

Необходимо задать свойство **Путь к стандартной библиотеке** адаптера ресурсов MQ в Websphere для версии WebSphere MQ, поддерживаемой сервером, и перезапустить сервер. Этим обеспечивается использование библиотек поддерживаемой версии WebSphere MQ. Подробные требования к аппаратному обеспечению приведены на страницах поддержки IBM.

Привязки импортов WebSphere MQ

Привязка импорта WebSphere MQ позволяет компонентам в пределах вашего модуля SCA соединяться со службами, предоставляемыми внешними приложениями на основе WebSphere MQ. Необходимо использовать поддерживаемую версию WebSphere MQ. Подробные требования к аппаратному обеспечению приведены на страницах поддержки IBM.

Взаимодействие с внешними системами WebSphere MQ включает использование очередей для отправки запросов и получения ответов.

Для привязки импорта WebSphere MQ поддерживаются два типа сценариев использования в зависимости от типа вызываемой операции:

- Односторонний: Импорт WebSphere MQ помещает сообщение в очередь, настроенную в поле **Очередь целевого объекта отправления** привязки импорта. В поле replyTo заголовка MQMD не отправляется ничего.
- Двухсторонний (запрос-ответ): Импорт WebSphere MQ помещает сообщение в очередь, настроенную в поле **Очередь целевого объекта отправления**

  Очередь получения задается в поле заголовка replyTo MQMD. Управляемый сообщениями EJB (MDB) развертывается для прослушивания очереди получения, и по получении ответа MDB передает этот ответ назад компоненту.

  Привязку импорта можно настроить (с помощью поля **Схема зависимости ответа**) на ожидание ИД зависимости ответного сообщения, скопированного из ИД сообщения-запроса (по умолчанию) или из ИД зависимости сообщения-запроса.

Важно отметить, что WebSphere MQ является асинхронной привязкой. Если вызывающий компонент вызывает импорт WebSphere MQ синхронно (в двухсторонней операции), то вызывающий компонент блокируется до возвращения ответа службой WebSphere MQ.

рис. 39 иллюстрирует, как импорт связывается с внешней службой.

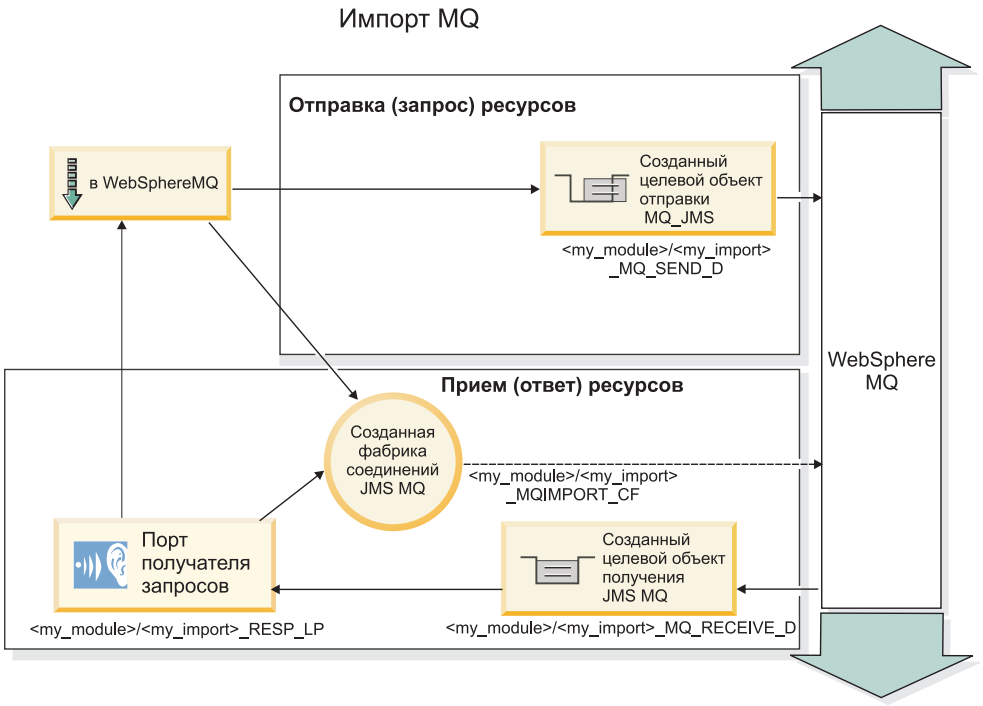


Рисунок 39. Ресурсы привязки импорта WebSphere MQ

Привязки экспортов WebSphere MQ

Привязка экспорта WebSphere MQ обеспечивает средства для модуля SCA по предоставлению служб внешним приложениям на основе WebSphere MQ.

Развертывается MDB прослушивания запросов, приходящих в **очередь целевого объекта получения**, указанную в привязке экспорта. Очередь, заданная в поле **Очередь целевого объекта отправления**, используется для отправки ответа на входящий запрос, если вызванный компонент предоставляет ответ. Очередь, заданная в поле replyTo ответного сообщения, переопределяет очередь, указанную в поле **Очередь целевого объекта отправления**.

На рисунке рис. 40 на стр. 143 показано, как внешний инициатор запроса связывается с экспортом.

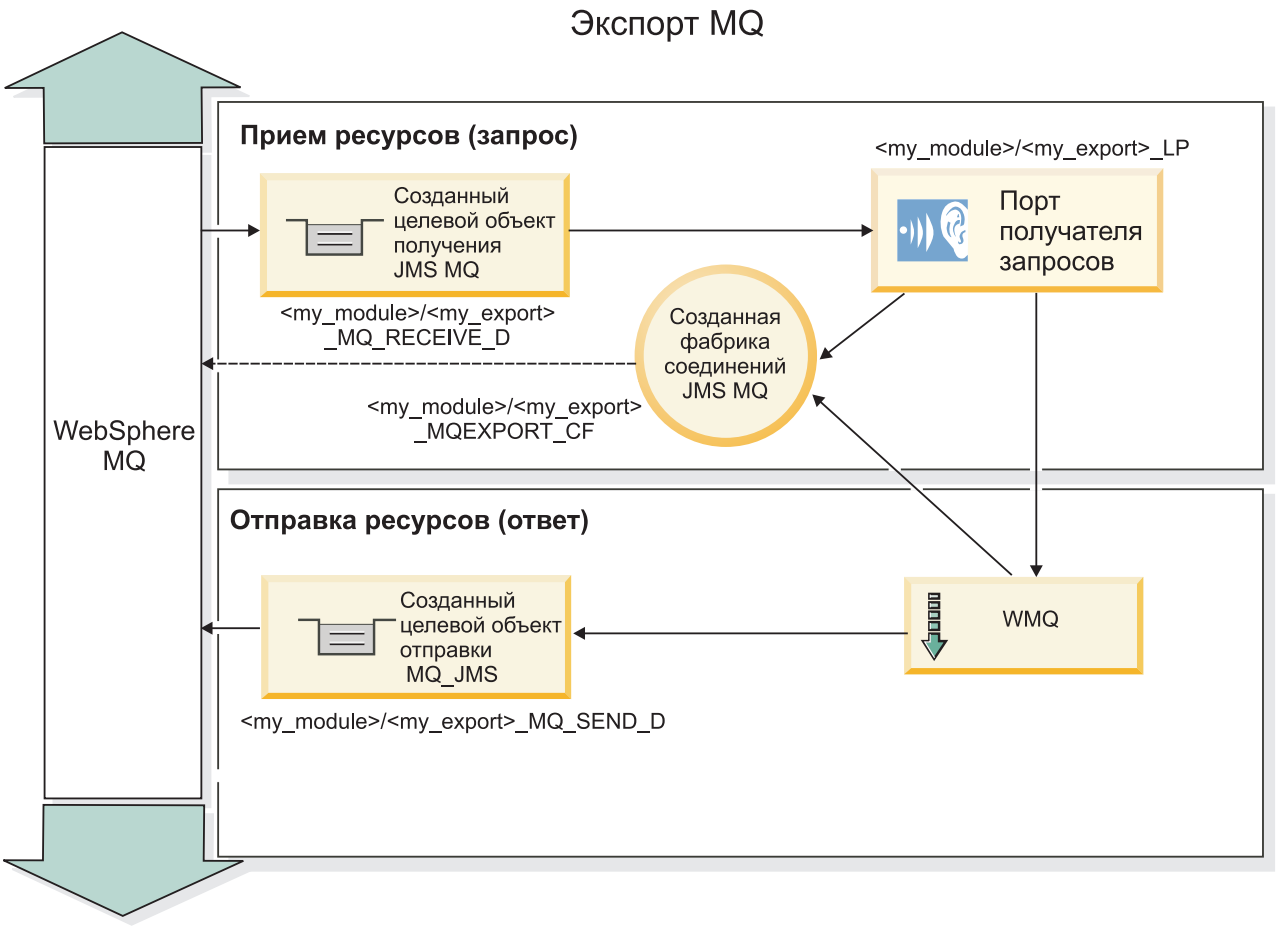


Рисунок 40. Ресурсы привязки экспорта WebSphere MQ

Примечание: рис. 39 на стр. 142 и рис. 40 иллюстрируют, как приложение из предыдущей версии IBM Business Process Manager связывается с внешней службой. В случае приложений, развернутых для IBM Business Process Manager версии 7.x или более поздней, используется спецификация активации вместо порта получателя и фабрики соединений.

Основные компоненты привязки WebSphere MQ:

Основные компоненты привязки WebSphere MQ включают заголовки, артефакты Java EE и созданные ресурсы Java EE.

Схемы зависимостей

Приложение запроса/ответа WebSphere MQ может использовать один из многих способов связывания ответных сообщений с запросами, создаваемых полями MQMD MessageID и CorrelID. В большинстве случаев инициатор разрешает администратору очередей выбрать MessageID и ожидает копирования отвечающего приложения в CorrelID ответа. В большинстве случаев инициатору и отвечающему приложению косвенно известно, как метод зависимости используется. Иногда отвечающее приложение ставит различные флаги в поле Отчет ответа, которые описывают, как обрабатывать эти поля.

Привязки экспортов для сообщений WebSphere MQ могут настраиваться на следующие опции:

Опции MsgId ответа:

`Создать MsgID`
Позволяет администратору очередей выбрать уникальный MsgId для ответа (по умолчанию).

Копировать из MsgID ответа
Копирует поле MsgId из поля MsgId ответа.

Копировать из сообщения SCA
Задает то MsgId, которое переносится в заголовках WebSphere MQ ответного сообщения SCA или позволяет администратору очередей определить новый Id, если это значение не существует.

Опции согласно отчету
Проверяет поле Отчет MQMD в запросе на указание, как обрабатывать MsgId. Опции MQRO_NEW_MSG_ID и MQRO_PASS_MSG_ID поддерживаются и действуют аналогично опциям Создать MsgId и Копировать из MsgID ответа.

Опции CorrelId ответа:

Копировать из MsgID ответа
Копирует поле CorrelId из поля MsgId ответа (по умолчанию).

Копировать из CorrelID ответа
Копирует поле CorrelId из поля CorrelId ответа.

Копировать из сообщения SCA
Задает то CorrelId, которое переносится в заголовках WebSphere MQ ответного сообщения SCA или оставляет его пустым, если это значение не существует.

Опции согласно отчету
Проверяет поле Отчет MQMD в запросе на указание, как обрабатывать CorrelId. Опции MQRO_COPY_MSG_ID_TO_CORREL_ID и MQRO_PASS_CORREL_ID поддерживаются и действуют аналогично опциям Копировать из MsgID запроса и Копировать из CorrelID ответа.

Привязки импортов для сообщений WebSphere MQ могут настраиваться на следующие опции:

Опции MsgId запроса:

Создать MsgID
Позволяет администратору очередей выбрать уникальный MsgId для запроса (по умолчанию).

Копировать из сообщения SCA
Задает то MsgId, которое переносится в заголовках WebSphere MQ сообщения-запроса SCA или позволяет администратору очередей определить новый Id, если это значение не существует.

Опции зависимости ответа:

CorrelID ответа копируется из MsgId
Ожидается, что ответное сообщение имеет поле CorrelId согласно MsgId запроса (по умолчанию).

MsgID ответа копируется из MsgId
Ожидается, что ответное сообщение имеет поле MsgId согласно MsgId запроса.

CorrelID ответа копируется из CorrelId
Ожидается, что ответное сообщение имеет поле CorrelId согласно CorrelId запроса.

Ресурсы EE

При развертывании привязки WebSphere MQbinding в среду Java EE создаются многие ресурсы Java EE.

Параметры

Фабрика соединений MQ
Используется клиентами для установления соединения с провайдером WebSphere MQ.

Фабрика соединений ответов
Используется средой выполнения SCA MQ, когда назначения отправления и получения относятся к разным администраторам очередей.

Спецификация активации
Спецификация активации MQ JMS связывается с одним или несколькими управляемыми сообщениями EJB и обеспечивает конфигурацию, необходимую им для получения сообщений.

Целевые объекты

- Назначение отправления: Куда отправляется запрос или исходящее сообщение (импорт); куда отправляется ответное сообщение (экспорт) в отсутствие замещающего поля заголовка MQMD ReplyTo во входящем сообщении.
- Назначение получения: Куда должен быть размещен ответ/запрос или входящее сообщение.

Заголовки WebSphere MQ:

В заголовках WebSphere MQ применяются определенные соглашения для преобразования в сообщения архитектуры компонентов служб (SCA).

Сообщения WebSphere MQ включают системный заголовок (MQMD), ноль или более других заголовков MQ (системных или пользовательских) и тело сообщения. При наличии нескольких заголовков сообщения важен их порядок.

Каждый заголовок содержит информацию, описывающую структуру следующего заголовка. MQMD описывает первый заголовок.

Как обрабатываются заголовки MQ

Для обработки заголовков MQ применяется связывание данных заголовка MQ. Следующие заголовки поддерживаются автоматически:

- MQRFH
- MQRFH2
- MQCIH
- MQIIH

Заголовки, начинающиеся с **MQH**, обрабатываются по-другому. Особые поля заголовка не обрабатываются, а остаются в виде набора байтов.

Для обработки прочих заголовков MQ можно подготовить пользовательское связывание данных заголовка MQ.

Как осуществляется доступ к заголовкам MQ

Доступ к заголовкам MQ в продукте может осуществляться следующими способами:

- Посредством объекта сообщения службы (SMO) в посреднике
- Посредством ContextService API

В системе заголовки MQ представляются элементом MQHeader SMO. MQHeader - это контейнер данных заголовка, который расширяет MQControl, но при этом содержит элемент anyType. Он содержит MQMD, MQControl (управляющая информация о теле сообщения MQ), а также список прочих заголовков MQ.

- MQMD представляет содержимое описания сообщения WebSphere MQ, за исключением информации, определяющей структуру и кодировку тела сообщения.
- MQControl содержит информацию, определяющую структуру и кодировку тела сообщения.
- MQHeaders содержит список объектов MQHeader.

Цепочка заголовков MQ не зацеплена, то есть внутри SMO каждый заголовок MQ несет свою собственную управляющую информацию (CCSID, Encoding, Format). Тем самым заголовки можно добавлять и удалять, не затрагивая другие данные заголовков.

Настройка полей в MQMD

Обновить MQMD можно с помощью Context API или посредством объекта сообщения службы (SMO) в посреднике. Следующие поля автоматически передаются в исходящее сообщение MQ:

- Encoding
- CodedCharacterSet
- Format
- Report
- Expiry
- Feedback
- Priority
- Persistence
- CorrelId
- MsgFlags

Настройте связывание MQ для элемента Import или Export, чтобы передавать следующие свойства в исходящее сообщение MQ:

MsgID
В поле **ИД сообщения-запроса** выберите `Скопировать из сообщения SCA`.

MsgType
Выключите переключатель **Установить тип сообщения MQMT_DATAGRAM или MQMT_REQUEST для операции запрос-ответ**.

ReplyToQ
Выключите переключатель **Переопределить ответ в очередь сообщения-запроса**.

ReplyToQMgr
Выключите переключатель **Переопределить ответ в очередь сообщения-запроса**.

Начиная с версии 7.0 поля контекста можно переопределить в пользовательском свойстве определения целевого объекта JNDI. Присвойте пользовательскому свойству MDCTX значение `SET_IDENTITY_CONTEXT` для целевого расположения отправки, чтобы передавать следующие поля в исходящее сообщение MQ:

- UserIdentifier
- AppIdentityData

Присвойте пользовательскому свойству MDCTX значение `SET_ALL_CONTEXT` для целевого расположения отправки, чтобы передавать следующие свойства в исходящее сообщение MQ:

- UserIdentifier
- AppIdentityData
- PutApplType
- PutApplName
- ApplOriginData

Некоторые поля не передаются в исходящее сообщение MQ. Следующие поля переопределяются в ходе отправки сообщения:

- BackoutCount
- AccountingToken

- PutDate
- PutTime
- Offset
- OriginalLength

Статическое добавление MQCIH в привязку WebSphere MQ:

IBM Business Process Manager поддерживает статическое добавление информации заголовка MQCIH без использования модуля передачи.

Существуют разные способы добавления информации заголовка MQCIH в сообщение (например, с использованием примитива передачи Header Setter). Может быть удобнее добавить эту информацию заголовка без использования дополнительного модуля передачи. Статическая информация заголовка, включающая имя программы CICS, ИД транзакции и другие сведения заголовка формата данных, может быть определена или добавлена как часть привязки WebSphere MQ.

WebSphere MQ, MQ CICS Bridge и CICS должны быть настроены на статическое добавление информации заголовка MQCIH.

Integration Designer можно использовать для настройки импорта WebSphere MQ со статическими значениями, которые требуются для информации заголовка MQCIH.

Когда сообщение приходит и обрабатывается импортом WebSphere MQ, выполняется проверка на присутствие информации заголовка MQCIH в сообщении. Если MQCIH присутствует, то статические значения, определяемые в импорте WebSphere MQ, используются для переопределения соответствующих динамических значений в сообщении. Если MQCIH отсутствует, то он создается в сообщении, и добавляются статические значения, определенные в импорте WebSphere MQ.

Статические значения, определенные в импорте WebSphere MQ, зависят от метода. Можно задать другие статические значения MQCIH для разных методов в пределах одного импорта WebSphere MQ.

Это средство не используется для предоставления значений по умолчанию, если MQCIH не содержит конкретной информации заголовка, поскольку статическое значение, определенное в импорте WebSphere MQ, будет переопределено соответствующим значением, предусмотренным во входящем сообщении.

Внешние клиенты:

IBM Business Process Manager может отправлять сообщения или принимать сообщения от внешних клиентов с помощью привязок WebSphere MQ.

Внешний клиент (например, веб-портал или информационная система организации) может отправить сообщение компоненту SCA в приложении при помощи экспорта или он может быть вызван компонентом SCA в приложении при помощи импорта.

Привязка экспорта WebSphere MQ развертывает управляемые сообщениями EJB (MDB) для прослушивания запросов, приходящих на целевой объект получения, указанный в привязке экспорта. Целевой объект указывается в поле send и используется для отправки ответа на входящий запрос, если вызванное приложение предоставляет ответ. Таким образом, внешний клиент может вызывать приложения через привязку экспорта.

Импорты WebSphere MQ привязываются к внешним клиентам и могут доставлять им сообщения. Эти сообщения могут требовать или могут не требовать ответа от внешнего клиента.

Дополнительную информацию о взаимодействии с внешними клиентами с помощью WebSphere MQ можно найти в справочной системе WebSphere MQ Information Center.

Устранение неполадок привязок WebSphere MQ:

Сбои и условия сбоев привязок WebSphere MQ JMS можно диагностировать и устранять.

Основные условия сбоя

Основные условия сбоя привязок WebSphere MQ определяются семантикой транзакций, конфигурацией WebSphere MQ или ссылкой на фактическое поведение других компонентов. В число основных условий сбоя входят следующие.

- Сбой установки соединения с администратором очередей или очередью WebSphere MQ.

  Сбой установки соединения с WebSphere MQ для приема сообщений приведет к сбою запуска порта получения запросов MDB. Это событие будет записано в протокол WebSphere Application Server. Постоянные сообщения останутся в очереди WebSphere MQ до момента их успешного получения (или пока WebSphere MQ не определит, что их срок истек).

  Сбой установки соединения с WebSphere MQ для отправки исходящих сообщений приведет к откату транзакции, которая управляет отправкой.
- Сбой анализа входящего сообщения или создания исходящего сообщения.

  Сбой связывания данных приводит к откату транзакции, которая управляет работой.
- Сбой отправки исходящего сообщения.

  Сбой отправки исходящего сообщения приводит к откату соответствующей транзакции.
- Получение нескольких или непредвиденных ответных сообщений.

  Импорт ожидает только одно ответное сообщение для каждого сообщения запроса. Если получено несколько ответов или ответ запоздал (время истечения срока ожидания ответа SCA истекло), то выбрасывается исключительная ситуация рабочей среды. Выполняется откат транзакции, и ответное сообщение удаляется из очереди или обрабатывается диспетчером недоставленных событий.

Сценарии неправильного употребления: сравнение с привязками WebSphere MQ JMS

Импорт и экспорт WebSphere MQ изначально предназначены для взаимодействия со стандартными приложениями WebSphere MQ и предоставляют передачам доступ ко всему содержимому тела сообщения WebSphere MQ. Однако привязка WebSphere MQ JMS предназначена для взаимодействия с приложениями JMS, развернутых на WebSphere MQ, которая предоставляет сообщения в соответствии с моделью сообщений JMS.

Следующие сценарии следует компоновать с использованием привязки WebSphere MQ JMS, а не WebSphere MQ.

- Вызов объекта EJB, управляемого сообщениями JMS (MDB) из модуля SCA, если MDB развернут на поставщике WebSphere MQ JMS. Используйте импорт WebSphere MQ JMS.
- Разрешение вызова модуля SCA из сервлета компонента Java EE или EJB с использованием JMS. Используйте экспорт WebSphere MQ JMS.
- Транзитная передача содержимого JMS MapMessage через WebSphere MQ. Используйте экспорт и импорт WebSphere MQ в комбинации с соответствующей привязкой данных.

Существуют ситуации, в которых ожидается взаимодействие привязки WebSphere MQ с привязкой WebSphere MQ JMS. В частности, в случае стыковки приложений WebSphere MQ, использующих Java EE, с приложениями, не использующими Java EE, используйте экспорт WebSphere MQ и импорт WebSphere MQ JMS (или наоборот) в комбинации с соответствующими привязками данных или модулями передачи как по отдельности, так и совместно.

Недоставленные сообщения

Если WebSphere MQ не удается доставить сообщение в требуемое целевое расположение (например, по причине ошибок в конфигурации), то она отправляет сообщения в указанную очередь недоставленных сообщений.

В ходе этого процесса она добавляет заголовок недоставленнооого сообщения в начало тела сообщения. Этот заголовок содержит причины сбоя, исходное целевое расположение и другие сведения.

Сообщения SCA на основе MQ, которые не выводятся в диспетчере событий с ошибками

В случае сбоя сообщений SCA, полученных по причине сбоя взаимодействия с WebSphere MQ, ожидается, что они будут показаны в диспетчере недоставленных сообщений. Если эти сообщения не выводятся в диспетчере недоставленных событий, убедитесь, что значение свойства максимального числа сбоя доставок для целевого расположения WebSphere MQ больше 1. Если присвоить этому свойству значение 2 или большее значение, то то взаимодействие с диспетчером недоставленных событий во время вызовов SCA для привязок WebSphere MQ будет возможным.

Недоставленные сообщения MQ повторно отправляются в ошибочный администратор очередей

Когда для исходящих соединений требуется использовать предварительно определенную фабрику соединений, свойства соединения должны совпадать со свойствами, заданными в спецификации активации, используемой для входящих сообщений.

Предварительно определенная фабрика соединений используется для создания соединения во время повторной отправки недоставленного события и поэтому должна быть настроена для использования того же самого администратора очередей, из которого изначально было получено сообщение.

Обработка исключительных ситуаций:

В конфигурации привязки настраивается способ обработки исключительных ситуаций, сгенерированных обработчиками данных или привязками данных. На действия системы при возникновении такой исключительной ситуации также влияет тип потока передачи.

В обработчике данных или привязке данных, вызванных из привязки, может возникнуть неполадка. Например, обработчик данных может получить сообщение с поврежденными данными или попытаться прочитать сообщение в неверном формате.

Способ обработки таких исключительных ситуаций в привязке зависит от того, каким образом реализован обработчик данных или привязка данных. Рекомендуется, чтобы привязка данных генерировала исключительную ситуацию **DataBindingException**.

В случае обработчика данных ситуация полностью аналогична. Поскольку обработчик данных вызывается из привязки данных, любая исключительная ситуация обработчика данных инкапсулируется в исключительную ситуацию привязки данных. По этой причине об исключительной ситуации **DataHandlerException** сообщается как о **DataBindingException**.

При возникновении любой исключительной ситуации времени выполнения, включая **DataBindingException**, выполняются следующие действия:

- Если поток передачи настроен для выполнения транзакций, то по умолчанию в администраторе событий со сбоем сохраняется сообщение JMS, которое можно обработать или удалить вручную.

  Примечание: В привязке можно настроить такой режим исправления, чтобы сообщение не сохранялось в администраторе событий со сбоем, а вместо этого выполнялся откат.
- Если поток передачи не поддерживает транзакции, то исключительная ситуация регистрируется в протоколе, а сообщение удаляется.

В случае обработчика данных ситуация полностью аналогична. Поскольку обработчик данных вызывается из привязки данных, любая исключительная ситуация обработчика данных инкапсулируется в исключительную ситуацию привязки данных. По этой причине об исключительной ситуации **DataHandlerException** сообщается как о **DataBindingException**.

Ограничения привязок:

При использовании привязок действуют определенные ограничения, которые описаны в этом разделе.

Ограничения привязки MQ:

Использование привязки MQ имеет несколько ограничений, которые рассматриваются в этом разделе.

Нет рассылки сообщений публикация-подписка

Метод публикация-подписка рассылки сообщений в настоящее время не поддерживается привязкой MQ, хотя сама WMQ поддерживает сообщения публикация-подписка. Однако привязка MQ JMS поддерживает этот метод рассылки.

Общие очереди получения

Многие привязки экспортов и импортов WebSphere MQ предполагают, что все сообщения, присутствующие в настроенных для них очередях получения, предназначены для этого экспорта или импорта. Привязки импортов и экспортов должны настраиваться с учетом следующего:

- Каждый импорт MQ должна иметь свою очередь получения, поскольку привязка импорта MQ предполагает, что все сообщения в очереди получения являются ответами на отправленные ею запросы. Если очередь получения совместно используется более чем одним импортом, то ответы могут быть получены другим импортом и не смогут быть связаны с исходным сообщением-запросом.
- Каждый экспорт MQ должен иметь свою очередь получения, поскольку в противном случае нельзя будет предсказать, какой экспорт получит конкретное сообщение-запрос.
- Импорты и экспорты MQ могут указывать на одну очередь отправления.

Ограничения привязок JMS, MQ JMS и базовых привязок JMS:

Привязки JMS и MQ JMS имеют некоторые ограничения.

Замечания о создании привязок по умолчанию

Ограничения использования привязок JMS, MQ JMS и базовых привязок JMS рассматриваются в следующих разделах:

- Замечания о создании привязок по умолчанию
- Схема зависимостей ответов
- Поддержка двунаправленного текста

При создании привязки несколько полей заполняются по умолчанию, но можно вводить и другие значения. Например, создается имя фабрики соединений. Если известно, что ваше приложение будет размещено на сервере с удаленным доступом к нему с помощью клиента, то во время создания привязки необходимо ввести имена JNDI, а не использовать значения по умолчанию, поскольку, скорее всего, потребуется управлять этими значениями через административную консоль во время выполнения.

Однако, если после принятия значений по умолчанию оказалось, что у вас нет доступа к приложению из удаленного клиента, то можно воспользоваться административной консолью для явного задания значения фабрики соединений. Найдите поле конечных точек провайдера в параметрах фабрики соединений и добавьте значение, например <имя-хоста-сервера>:7276 (если используется номер порта по умолчанию).

Схема зависимостей ответов

При использовании схемы зависимостей ответов ИД зависимости - ИД зависимости для связывания сообщений в операции запрос-ответ необходимо иметь динамический ИД зависимости в сообщении.

Для создания динамического ИД зависимости в модуле передачи в редакторе потока передачи перед импортом с помощью привязки JMS добавьте примитив передачи Mapping. Откройте редактор преобразований. Известные заголовки архитектуры компонентов служб будут доступны в целевом сообщении. Перенесите поле, содержащее уникальный ИД, в исходное сообщение на ИД зависимости в заголовке JMS целевого сообщения.

Поддержка двунаправленного текста

Во время выполнения для имен Java Naming and Directory Interface (JNDI) поддерживаются только символы ASCII.

Общие очереди получения

Многие привязки экспортов и импортов предполагают, что все сообщения, присутствующие в настроенных для них очередях получения, предназначены для этого экспорта или импорта. Привязки импортов и экспортов должны настраиваться с учетом следующего:

- Каждая привязка импорта должна иметь свою очередь получения, поскольку привязка импорта предполагает, что все сообщения в очереди получения являются ответами на отправленные ею запросы. Если очередь получения совместно используется более чем одним импортом, то ответы могут быть получены другим импортом и не смогут быть связаны с исходным сообщением-запросом.
- Каждый экспорт должен иметь свою очередь получения, поскольку в противном случае нельзя будет предсказать, какой экспорт получит конкретное сообщение-запрос.
- Импорты и экспорты могут указывать на одну очередь отправления.

Бизнес-объекты

В отрасли разработки программного обеспечения для компьютеров разработано несколько моделей и сред программирования, в которых *бизнес-объекты* предоставляют обычное представление бизнес-данных для обработки приложения.

Обычно эти бизнес-объекты имеют следующие характеристики:

- определяются с использованием отраслевых стандартов;
- прозрачно связывают данные с таблицами баз данных или с информационными системами предприятия;
- поддерживают протоколы удаленного вызова;
- предоставляют основу модели программирования приложений.

Process Designer и Integration Designer предоставляют разработчикам одну из таких моделей бизнес-объектов для представления различных видов бизнес-сущностей из различных доменов.

В Process Designer бизнес-объекты концентрируются на представлении типа данных. В наборах инструментов доступны базовые бизнес-объекты (типы переменных). Кроме того, можно создавать пользовательские типы переменных (пользовательские бизнес-объекты). См. раздел Бизнес-объекты и переменные.

В Integration Designer, который доступен только в IBM Business Process Manager Advanced, бизнес-объекты могут представлять более сложные конструкции XSD. В Integration Designer бизнес-объекты имеют тесную привязку со схемами XML. Дополнительная информация об интеграции бизнес-объектов Integration Designer с бизнес-объектами Process Designer приведена в разделах Зеркальное копирование библиотеки и Конструктивные элементы XML не поддерживаются.

Во время разработки в Integration Designer модель бизнес-объектов позволяет разработчикам определять бизнес-объекты в виде определений схемы XML. Во время выполнения бизнес-данные, определенные в определениях схемы XML, представлены в виде бизнес-объектов Java. В этой модели бизнес-объекты основаны на ранних проектах спецификации объектов данных служб (SDO) и предоставляют полный набор интерфейсов приложений программной модели, требуемых для управления бизнес-данными. В следующих подразделах приведена дополнительная информация о применении бизнес-объектов в Integration Designer.

Определение бизнес-объектов

Определения бизнес-объектов создаются с помощью редактора бизнес-объектов в Integration Designer. Редактор бизнес-объектов хранит бизнес-объекты в виде определений схем XML.

Использование схемы XML для определения бизнес-объектов имеет некоторые преимущества.

- Схема XML предоставляет модель определения данных, основанную на стандартах, и базу для взаимодействия между несовместимыми разнородными системами и приложениями. Схемы XML используются совместно с языком описания веб-служб (WSDL) для предоставления стандартных контрактов интерфейсов между компонентами, приложениями и системами.
- Схема XML предоставляет расширенную модель определения данных для представления бизнес-данных. Эта модель, в частности, включает в себя сложные типы, простые типы, пользовательские типы, наследование типов и количество значений.
- Бизнес-объекты могут быть определены в организациях отраслевых стандартов или в других системах и приложениях с помощью бизнес-интерфейсов и данных, определенных на языке описания веб-служб, а также с помощью схемы XML. Integration Designer может непосредственно импортировать эти бизнес-объекты.

Кроме того, Integration Designer обеспечивает поддержку обнаружения бизнес-данных в базах данных и информационных системах предприятия и последующего создания определения бизнес-объекта в виде стандартизированной схемы XML для этих бизнес-данных. Бизнес-объекты, созданные таким образом, часто называются *бизнес-объектами приложений*, поскольку их структура копирует структуру бизнес-данных, определенных в информационной системе предприятия.

Когда процесс манипулирует данными из многих различных информационных систем, очень полезным может оказаться преобразование несовместимого представления бизнес-данных (например, CustomerEIS1 и CustomerEIS2 или OrderEIS1 и OrderEIS2) в единое каноническое представление (например, Customer или Order). Каноническое представление часто называют *обобщенным бизнес-объектом*.

Определения бизнес-объектов, в частности обобщенных, часто используются несколькими приложениями. Для поддержки повторного использования Integration Designer позволяет создавать бизнес-объекты в библиотеках, которые впоследствии можно связывать с несколькими модулями приложения.

Язык описания веб-служб (WSDL) определяет контракты для служб, предоставляемые и принимаемые модулем приложения SCA, а также контракты, используемые для создания компонентов в модуле приложения. На WSDL могут быть описаны как операции, так и бизнес-объекты контракта (определенные схемой XML для представления бизнес-данных).

Работа с бизнес-объектами

Архитектура служебных компонентов (SCA) предоставляет среду определения модуля приложения, служб, которые оно предоставляет, служб, которыми оно пользуется, и состав компонентов, которые обеспечивают бизнес-логику модуля приложения. Бизнес-объекты играют важную роль в приложении, определяя бизнес-данные, которые используются для описания контрактов служб и компонентов и бизнес-данных, которыми манипулируют компоненты.

На следующей диаграмме показан модуль приложения SCA и проиллюстрированы многие точки, в которых разработчик работает с бизнес-объектами.

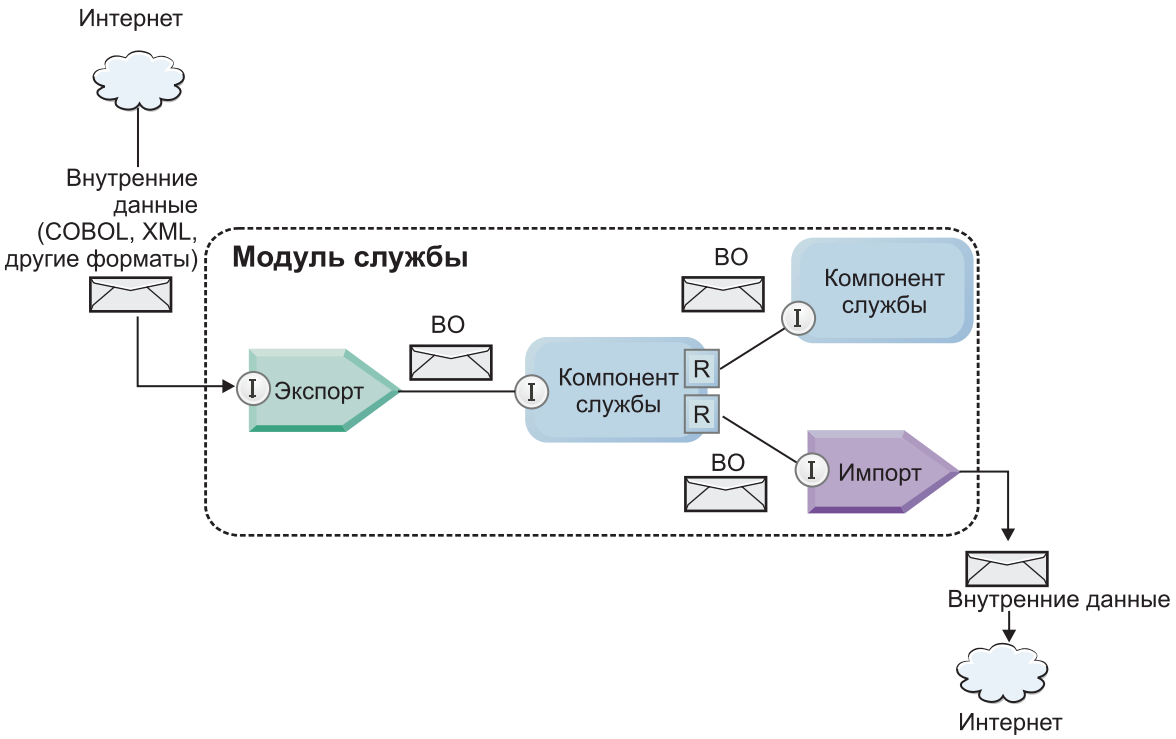


Рисунок 41. Бизнес-объекты представляют собой данные, которыми обмениваются собой службы приложения

Примечание: В этом разделе описано использования бизнес-объектов модулями приложения SCA. Если используются интерфейсы Java, то модули приложения SCA также могут обрабатывать объекты Java.

Программная модель бизнес-объектов

Программная модель бизнес-объектов состоит из набора интерфейсов Java, которые представляют собой:

- определение бизнес-объекта и данные экземпляра;
- набор служб, которые поддерживают операции над бизнес-объектами.

Определения типов бизнес-объектов представлены интерфейсами commonj.sdo.Type и commonj.sdo.Property. Программная модель бизнес-объектов предоставляет набор правил преобразования информации сложных типов схемы XML в интерфейс Type, а также каждого элемента определения сложного типа в интерфейс Property.

Экземпляры бизнес-объектов представлены интерфейсом commonj.sdo.DataObject. Программная модель бизнес-объектов является нетипизированной. Это означает, что один и тот же интерфейс commonj.sdo.DataObject можно использовать для представления различных определений бизнес-объектов, например Customer и Order. Определение свойств, значения которых можно задавать для бизнес-объекта и получать из него, зависят от типа информации, указанной в схеме XML, которая связана с каждым бизнес-объектом.

Поведение программной модели бизнес-объектов основано на спецификации объектов данных служб 2.1. Дополнительные сведения приведены в спецификации SDO 2.1 для Java, учебниках и javadocs в Интернете: http://www.oasis-opencsa.org/.

Службы бизнес-объектов поддерживают различные операции жизненного цикла (например, создание, равенство, анализ и сериализацию) бизнес-объектов.

Специфика программной модели бизнес-объектов описана в разделах Программирование с использованием служб бизнес-объектов и Генерируемая документация по API и SPI для бизнес-объектов.

Привязки, привязки данных и обработчики данных

Как показано в рис. 41 на стр. 153, бизнес-данные, используемые для вызова служб, предоставляемых модулями приложения SCA, преобразуются в бизнес-объекты таким образом, чтобы компоненты SCA могли манипулировать бизнес-данными. Подобным образом бизнес-объекты, которыми манипулируют компоненты SCA, преобразуются в формат данных, требуемый для внешних служб.

В некоторых случаях, например, в случае привязки веб-службы, привязка, используемая для экспорта и импорта служб, автоматически преобразует данные в соответствующий формат. В других случаях, например, в случае привязки JMS, разработчики могут предоставить привязку данных или обработчик данных, который преобразует нестандартные форматы в бизнес-объекты, представленные интерфейсом DataObject.

Дополнительные сведения о разработке привязок данных и обработчиков данных приведены в разделах “Обработчики данных” на стр. 59 и “Привязки данных” на стр. 61.

Компоненты

Компоненты SCA определяют свои контракты служб для передачи и приема с помощью комбинации языка описания веб-служб и схемы XML. Бизнес-данные, передаваемые SCA между компонентами, представлены в виде бизнес-объектов, использующих интерфейс DataObject. SCA проверяет совместимость типов этих бизнес-объектов с контрактом интерфейса, определенного в компоненте, который требуется вызвать.

Обобщения программной модели для манипулирования бизнес-объектами отличаются в различных компонентах. Компонент POJO и примитив Custom компонента потока передачи обеспечивают непосредственное управление бизнес-объектами, включая непосредственное программирование на Java с помощью интерфейсов и служб программирования бизнес-объектов. Большинство компонентов предоставляют высокоуровневые абстракции для манипулирования бизнес-объектами, а также фрагменты кода Java для определения пользовательского поведения в интерфейсах и службах бизнес-объектов.

Бизнес-объекты можно преобразовать с помощью любой комбинации компонентов Interface Flow Mediation и Business Object Map или компонента потока передачи и его примитива карты связей XML. Эти возможности преобразования бизнес-объектов полезны для преобразования бизнес-объектов, специфических для приложения, в обобщенные бизнес-объекты и обратно.

Специальные бизнес-объекты

Объекты служебных сообщений и бизнес-графиков - два специализированных типа бизнес-объектов, которые используются для особых целей приложения.

Объект служебного сообщения

Объект служебного сообщения (SMO) - это специализированный бизнес-объект, который используется компонентами потока передачи для представления набора данных, связанных с вызовом службы.

SMO имеет фиксированную структуру верхнего уровня, состоящую из заголовков, контекста, тела и вложений (если таковые имеются).

- Заголовки содержат информацию, связанную с вызовом службы с использованием отдельного протокола или привязки. Примерами являются заголовки SOAP и заголовки JMS.
- Данные контекста содержат дополнительную логическую информацию, связанную с вызовом во время его обработки компонентом потока передачи. Эта информация обычно не является частью данных приложения, отправляемых или получаемых клиентами.
- Тело SMO содержит бизнес-данные полезной нагрузки, которые представляют базовое сообщение приложения или данных вызова в форме стандартного бизнес-объекта.

Кроме того, SMO может содержать данные вложения для вызовов веб-службы с использованием SOAP со вложениями.

Потоки передачи выполняют такие задачи как маршрутизация запросов и преобразование данных, а SMO предоставляет объединенное представление заголовка и содержимого полезной нагрузки в единой унифицированной структуре.

Бизнес-график

Бизнес-график является специальным бизнес-объектом, который используется для обеспечения поддержки синхронизации данных в сценариях интеграции.

Рассмотрим пример, в котором две информационные системы предприятия имеют представление определенного заказа. При изменении заказа в одной системе можно отправить другой системе сообщение относительно синхронизации данных. Бизнес-графики поддерживают отправку части измененного заказа другой системе с комментарием, содержащем краткие сведения об изменении, определяющие тип изменения.

В данном примере бизнес-график Order сообщит другой системе, что одна из позиций в заказе удалена и что свойство предполагаемой даты отправки заказа обновлено.

Бизнес-графики можно без затруднения добавлять к существующим бизнес-объектам в Integration Designer. Чаще всего они находятся в сценариях, в которых используются адаптеры WebSphere, и для поддержки приложений WebSphere InterChange Server.

Режим анализа бизнес-объектов

В Integration Designer существует свойство модулей и библиотек, которое можно использовать для переключения активного и отложенного режимов анализа XML для бизнес-объектов.

- Если этой опции присвоено значение *активный*, то потоки байтов XML во время создания бизнес-объекта анализируются активно.
- Если опции присвоено значение *отложенный*, то бизнес-объект создается обычным способом, а фактическая обработка потока байтов XML откладывается и частичный анализ проводится только во время обращения к свойствам бизнес-объекта.

В любом режиме анализа XML при создании бизнес-объекта данные, имеющие иной формат, чем XML, всегда анализируются в активном режиме.

Замечания при выборе режима анализа бизнес-объекта:

Режим анализа бизнес-объекта определяет, как анализируются данные XML во время выполнения. Режим анализа бизнес-объекта определяется на модуле или библиотеке во время их создания. Можно изменить режим анализа модуля или библиотеки, однако необходимо знать о последствиях.

Режим анализа бизнес-объектов задается на уровне модулей и библиотек. Модули, которые созданы в IBM Integration Designer до версии 7, будут выполняться в режиме активного анализа без необходимости каких-либо изменений. По умолчанию модули и библиотеки, создаваемые в IBM Integration Designer версии 7 и выше пользуются наиболее подходящим режимом анализа в зависимости от числа факторов, в частности от режима анализа существующих проектов в рабочей области или от режима анализа зависимых проектов или других проектов в том же решении и так далее. Режим анализа бизнес-объектов модуля или библиотеки можно изменить в соответствии с требованиями существующей реализации, однако необходимо знать о следующем.

Замечания

- В режиме отложенного анализа бизнес-объектов данные XML обрабатываются быстрее. Однако существуют различия в совместимости между активным режимом и отложенным режимом, о которых необходимо знать перед изменением конфигурации модуля или библиотеки. Эти различия повлияют на

поведение модулей во время выполнения. Дополнительные сведения о том, какой режим анализа является оптимальным для приложения, приведены в разделе "Преимущества использования режима отложенного анализа по сравнению с режимом активного анализа" по связанным ссылкам.

- Модуль можно настроить для выполнения только в одном из режимов анализа. Библиотеки можно настроить для поддержки одного из режимов анализа или обоих режимов анализа. На библиотеку, настроенную для поддержки обоих режимов анализа, может ссылаться как модуль, использующий режим активного анализа, так и модуль, использующий режим отложенного анализа. Во время выполнения режим анализа библиотеки будет определяться модулями, которые на нее ссылаются. Во время выполнения модуль объявляет свой режим анализа, и этот режим анализа используется модулем и всеми библиотеками, которые он использует.
- Модули и библиотеки, настроенные для различных режимов анализа, совместимы в следующих случаях:
  - модули и библиотеки, настроенные для режима отложенного анализа, совместимы с библиотеками, которые используют режим отложенного анализа или как активного, так и отложенного анализа;
  - модули и библиотеки, настроенные для режима активного анализа, совместимы с библиотеками, которые используют режим активного анализа или как активного, так и отложенного анализа;
  - библиотеки, настроенные для режимов отложенного и активного анализа, совместимы только с библиотеками, использующими режим как отложенного, так и активного анализа;
- используйте один и тот же режим анализа для взаимодействия модулей, которые обмениваются сообщениями с использованием привязки SCA. Если модули обмениваются сообщениями с помощью других режимов анализа, то это может привести к недостаточной производительности.

Понятия, связанные с данным:

“Преимущества использования режимов отложенного и активного анализа”
В некоторых приложениях выгодно использовать режим отложенного анализа XML, а в некоторых производительность повышается при использовании режима активного анализа. Рекомендуется протестировать приложение в обоих режимах анализа для того чтобы определить, какой из них больше подходит для конкретных характеристик приложения.

Преимущества использования режимов отложенного и активного анализа:

В некоторых приложениях выгодно использовать режим отложенного анализа XML, а в некоторых производительность повышается при использовании режима активного анализа. Рекомендуется протестировать приложение в обоих режимах анализа для того чтобы определить, какой из них больше подходит для конкретных характеристик приложения.

Производительность приложений, анализирующих большие потоки данных XML, повышается в случае использования режима отложенного анализа XML. Производительность повышается с ростом потока байтов XML и уменьшения объема данных в потоке байтов, к которому обращается приложение.

Производительность следующих приложений может повышаться в режиме активного анализа:

- приложения, анализирующие потоки данных не в формате XML;
- приложения с сообщениями, созданные с помощью службы BOFactory;
- приложения, анализирующие очень короткие сообщения в формате XML.

Ссылки, связанные с данной:

“Замечания при выборе режима анализа бизнес-объекта” на стр. 155
Режим анализа бизнес-объекта определяет, как анализируются данные XML во время выполнения. Режим анализа бизнес-объекта определяется на модуле или библиотеке во время их создания. Можно изменить режим анализа модуля или библиотеки, однако необходимо знать о последствиях.

Замечания по миграции приложений и разработке:

Если требуется перенастроить приложение, которое изначально разработано для использования режима активного анализа, на использование режима отложенного анализа или если планируется предусмотреть

возможность переключения приложения между режимами отложенного и активного анализа, необходимо знать о различиях между режимами и об их переключении.

Обработка ошибок

Если в анализируемом потоке байтов XML присутствуют ошибки формата, то выбрасываются исключительные ситуации анализа.

- В режиме активного анализа XML эти исключительные ситуации происходят в момент выделения бизнес-объекта из входного потока XML.
- Если включен режим отложенного анализа XML, то исключительные ситуации анализа происходят скрыто при доступе к свойствам бизнес-объектов и анализе фрагмента XML с ошибкой формата.

Для работы с XML, в котором присутствует ошибка формата, выберите одну из следующих опций:

- развернуть корпоративную сервисную шину на ребрах для проверки входного XML;
- создать код отложенного распознавания ошибок в точке, в которой происходит обращение к свойствам бизнес-объектов.

Стеки и сообщения об исключительных ситуациях

Поскольку в основе режимов активного и отложенного анализа XML лежат разные реализации, трассировка стека, выбрасываемая интерфейсами программирования бизнес-объектов и службами, содержит одно и то же имя исключительной ситуации, но может содержать иное сообщение об исключительной ситуации или классы исключительных ситуаций, специфических для реализации, в оболочке.

Формат сериализации XML

Режим отложенного анализа XML обеспечивает оптимизацию производительности, при которой во время сериализации производится попытка копирования кода XML из входного потока байтов в выходной без изменений. В результате производительность растет, но формат сериализации выходного потока байтов XML может отличаться в случае обновления всего бизнес-объекта в режиме отложенного анализа и в случае его работы в режиме активного анализа XML.

Несмотря на то, что формат сериализации XML может быть не полностью эквивалентным синтаксически, семантическое значение, представляемое бизнес-объектом, будет эквивалентным независимо от режима анализа, и благодаря семантической эквивалентности приложения могут беспрепятственно обмениваться кодом XML, работая в различных режимах анализа.

Агент проверки экземпляра бизнес-объекта

Агент проверки экземпляра режима отложенного анализа XML бизнес-объекта обеспечивает высококачественную проверку бизнес-объектов, в частности проверку фасетов значений свойств. Благодаря этим усовершенствованиям, агент проверки экземпляра режима отложенного анализа перехватывает дополнительные неполадки, которые не перехватываются в режиме активного анализа, и предоставляет более подробные сообщения об ошибках.

Карты связей XML версии 602

Потоки передачи, изначально разработанные перед WebSphere Integration Developer версии 6.1, могли содержать примитивы Mapping, использующие карту связей таблицы стилей, которая не может непосредственно выполняться в режиме отложенного анализа XML. Во время миграции приложения для использования в режиме отложенного анализа XML файлы карты связей, связанные с примитивами Mapping, могут автоматически обновляться мастером миграции для выполнения в новом режиме. Однако если примитив Mapping относится непосредственно к таблице стилей, которую невозможно отредактировать вручную, то миграция таблицы стилей не происходит, и таблица стилей не может выполняться в режиме отложенного анализа XML.

Частные непубликуемые API

Если в приложении используются непубликуемые частные и специфические для реализации интерфейсы программирования бизнес-объектов, то может произойти сбой компиляции приложения при смене режима анализа. В режиме активного анализа эти частные интерфейсы обычно являются классами реализации бизнес-объекта, определенными в Eclipse Modeling Framework (EMF).

Во всех случаях настоятельно рекомендуется удалять частные API из приложения.

API EMF для объектов служебного сообщения

Компонент отображения в IBM Integration Designer предоставляет возможность манипулирования содержимым сообщения с помощью классов и интерфейсов Java из пакета com.ibm.websphere.sibx.smobo. В режиме отложенного анализа XML интерфейсы Java из пакета com.ibm.websphere.sibx.smobo также могут использоваться, но методы, которые относятся непосредственно к классам и интерфейсам Eclipse Modeling Framework (EMF) или наследуются от интерфейсов EMF, могут дать сбой.

ServiceMessageObject и его содержимое невозможно преобразовать в объекты EMF в режиме отложенного анализа XML.

Служба BOMode

Служба BOMode используется для того чтобы определить, является ли текущий режим анализа активным или отложенным.

Миграция

Все приложения до версии 7.0.0.0 работают в режиме активного анализа XML. При их миграции во время выполнения с помощью инструментов BPM для миграции во времени выполнения они продолжают работать в режиме активного анализа XML.

Для того чтобы приложение до версии 7.0.0.0 можно было использовать в режиме отложенного анализа XML, сначала требуется использовать Integration Designer для миграции артефактов приложения. После миграции можно настроить приложение для использования режима отложенного анализа XML.

Сведения о миграции артефактов в Integration Designer приведены в разделе Миграция исходных артефактов, а сведения о назначении режима анализа - в разделе Настройка режима анализа бизнес-объектов в модулях и библиотеках.

Взаимосвязи

Взаимосвязь определяет связь между двумя или более элементами данных, как правило - бизнес-объектами. В IBM Business Process Manager Advanced взаимосвязь может применяться для преобразования данных, которые одинаковы во всех бизнес-объектах или имеют различное представление, либо для определения взаимосвязей между различными объектами, расположенными в разных приложениях. Взаимосвязи могут совместно использоваться внутри приложения, в нескольких решениях и даже в нескольких продуктах.

Служба взаимосвязей в IBM Business Process Manager Advanced предоставляет инфраструктуру и операции для управления взаимосвязями. Поскольку она позволяет работать с любыми бизнес-объектами вне зависимости от их расположения, с ее помощью можно получить единую картину, описывающую все приложения предприятия. Кроме того, ее можно использовать как конструктивный блок в решениях BPM. Расширяемые и управляемые взаимосвязи могут использоваться в сложных решениях интеграции.

Что такое взаимосвязи?

Взаимосвязь определяет связь между бизнес-объектами. Каждый из таких бизнес-объектов называется *участником* взаимосвязи. Участники взаимосвязи отличаются друг от друга по выполняемой ими функции, или *роли*. Взаимосвязь содержит список возможных ролей.

Определение взаимосвязи содержит описание каждой роли и их связей между собой. Кроме того, оно описывает общее представление взаимосвязи. Например, одна роль может иметь только одного участника, а другая - произвольное число участников. Например, можно определить взаимосвязь *автомобиль-владелец*, в которой одному владельцу может принадлежать несколько автомобилей. Один из экземпляров этой взаимосвязи может иметь следующих участников в каждой из ролей:

- Автомобиль (Ferrari)
- Владелец (John)

Определение взаимосвязи задает шаблон *экземпляра* взаимосвязи. Экземпляр - это динамическое представление взаимосвязи. В примере взаимосвязи *автомобиль-владелец* экземпляр может описывать любое из следующих отношений:

- John владеет Ferrari
- Sara владеет Mazda
- Bob владеет Ferrari

Использование взаимосвязей освобождает от необходимости создания пользовательского кода для определения и отслеживания взаимосвязей в бизнес-логике. В ряде случаев служба взаимосвязей может делать все автоматически. Ознакомьтесь с примером, описанным в разделе Взаимосвязи идентификаторов.

Сценарии

Здесь описана типичная ситуация, в которой решению интеграции могут потребоваться взаимосвязи. Крупная компания покупает несколько компаний, или бизнес-подразделений. Каждое бизнес-подразделение использует свое программное обеспечение для отслеживания сотрудников и ноутбуков. Компании требуется средство для отслеживания всех сотрудников и их ноутбуков. Это решение должно предоставлять возможности для выполнения следующих задач:

- Просмотр всех сотрудников из разных бизнес-подразделений так, как если бы информация о них хранилась в одной базе данных
- Единое представление всех портативных ноутбуков
- Возможность для сотрудника войти в систему и купить ноутбук
- Объединение различных прикладных систем предприятия из разных бизнес-подразделений

Для решения этих задач компании, в частности, необходимо средство для того, чтобы пользователь John Smith из одного приложения и пользователь John A. Smith из другого приложения был показан как один сотрудник. То есть необходим способ получения консолидированного представления на основе данных из нескольких приложений.

Более сложные сценарии использования взаимосвязей включают в себя создание процессов BPEL, которые строят взаимосвязи между объектами из разных приложений. В еще более сложном сценарии бизнес-объекты расположены в решении интеграции, а не в приложениях. Служба взаимосвязей предоставляет платформу для постоянного управления взаимосвязями. При отсутствии такой службы вам бы потребовалось создать собственную службу хранения объектов. Ниже приведены два примера сложных сценариев использования взаимосвязей:

- В приложении SAP есть бизнес-объект **car** (автомобиль) с номером VIN, и необходимо узнать, не принадлежит ли данный автомобиль кому-то еще. Однако взаимосвязь владения установлена с пользователем, определенным в приложении PeopleSoft. В данном случае есть два решения, между которыми необходимо создать мост.

- Крупная компания, занимающаяся розничной торговлей, хочет получить возможность отслеживать возвращенные товары, деньги за которые были возвращены наличными или на кредитную карту. В этом должны участвовать два разных приложения: система управления заказами (OMS) с информацией о покупках и система управления возвратами (RMS) с информацией о возврате товара. Бизнес-объекты расположены в нескольких приложениях, и необходимо каким-то образом показать существующие между ними взаимосвязи.

Стандартные шаблоны

Чаще всего используются такие шаблоны взаимосвязей, как шаблоны *эквивалентности*. Они основаны на перекрестных ссылках или соотношении. Существует два типа взаимосвязей с таким шаблоном: *неидентифицирующие* и *идентифицирующие*.

- **Неидентифицирующие взаимосвязи** определяют связь между бизнес-объектами и другими данными по схеме один-ко-многим или многие-ко-многим. В каждом экземпляре взаимосвязи может быть один или несколько экземпляров каждого участника. Один из примеров неидентифицирующей взаимосвязи - это статическая взаимосвязь поиска. Например, она может связывать **CA** в приложении SAP со значением **California** в приложении Siebel.
- **Идентифицирующая взаимосвязь** устанавливает связь между бизнес-объектами и другими данными по схеме один-к-одному. В каждом экземпляре взаимосвязи может быть только один экземпляр каждого участника. Идентифицирующие взаимосвязи отражают перекрестные ссылки между семантически эквивалентными бизнес-объектами, которые по-разному идентифицируются в разных приложениях. Каждый участник взаимосвязи представляет бизнес-объект, у которого есть одно или несколько значений, которые однозначно идентифицируют этот объект. Идентифицирующие взаимосвязи обычно служат для преобразования ключевых атрибутов бизнес-объектов, таких как ИД или код продукта.

  Например, если в приложениях SAP, PeopleSoft и Siebel есть бизнес-объект **car**, то для их синхронизации в обычной ситуации вам потребовалось бы вручную создать логику синхронизации взаимосвязей, состоящую из шести связей:

  SAP -> универсальный объект

  универсальный объект -> SAP

  PeopleSoft-> универсальный объект

  универсальный объект-> PeopleSoft

  Siebel-> универсальный объект

  универсальный объект-> Siebel

  Однако служба взаимосвязей предоставляет готовые реализации шаблонов, содержащие все указанные экземпляры взаимосвязей.

Инструменты для работы с взаимосвязями

Редактор взаимосвязей из Integration Designer позволяет моделировать роли и взаимосвязи бизнес-интеграции. Подробная информация и инструкции по созданию взаимосвязей и использованию редактора взаимосвязей приведены в разделе Создание взаимосвязей.

Служба взаимосвязей - это инфраструктурная служба из состава IBM Business Process Manager, сохраняющая взаимосвязи и роли в системе и предоставляющая операции для управления ролями и взаимосвязями.

Администратор взаимосвязей - это административный интерфейс для управления взаимосвязями. С ним можно работать на страницах Администратор взаимосвязей в административной консоли.

Взаимосвязи можно вызывать внутри программ с помощью API службы взаимосвязей.

Служба взаимосвязей

Служба взаимосвязей хранит данные взаимосвязей в таблицах взаимосвязей, в которых содержатся значения, заданные в различных приложениях и решениях. Служба взаимосвязей предоставляет операции для управления ролями и взаимосвязями.

Принципы работы взаимосвязей

Взаимосвязи и роли можно определить в графическом интерфейсе редактора взаимосвязей, который доступен в Integration Designer. Служба взаимосвязей хранит данные о соответствии в таблицах базы данных взаимосвязей, расположенной в источнике данных, заданном при настройке службы взаимосвязей. Отдельная таблица (иногда именуемая таблицей участников) содержит информацию о каждом участнике взаимосвязи. Служба взаимосвязей использует таблицы взаимосвязей для отслеживания значений, заданных в разных приложениях, и передачи обновленной информации во все решения.

Взаимосвязи, представляющие собой бизнес-артефакты, можно развернуть в проекте или в общей библиотеке. При первом развертывании служба взаимосвязей добавляет данные.

Когда во время выполнения связям или другим компонентам IBM Business Process Manager требуется экземпляр взаимосвязи, экземпляры взаимосвязи обновляются или извлекаются, в зависимости от сценария.

С данными экземпляров взаимосвязи и роли можно работать тремя способами:

- Через вызовы API службы взаимосвязи во фрагменте кода на Java в компоненте IBM Business Process Manager
- С помощью преобразования взаимосвязей в службе преобразования бизнес-объектов IBM Business Process Manager
- Используя администратор взаимосвязей

Подробную информацию и пошаговые инструкции по созданию взаимосвязей, определению типов взаимосвязей и использованию редактора взаимосвязей можно найти в разделе Создание взаимосвязей.

Администратор взаимосвязей

Администратор взаимосвязей - это административный интерфейс для управления взаимосвязями. С ним можно работать на страницах Администратор взаимосвязей в административной консоли.

Администратор взаимосвязей предоставляет графический пользовательский интерфейс для создания и изменения данных о взаимосвязях и ролях во время выполнения. Элементами взаимосвязей можно управлять на всех уровнях: на уровне экземпляра взаимосвязи, экземпляра роли, данных атрибутов и данных свойств. С помощью администратора взаимосвязей можно выполнить следующие действия:

- Просмотреть список взаимосвязей в системе и подробную информацию о каждой взаимосвязи
- Управлять экземплярами взаимосвязей:
  - Запросить данные взаимосвязи для просмотра подмножеств данных экземпляра
  - Запросить данные взаимосвязи для просмотра подмножеств данных экземпляра с помощью представлений базы данных
  - Просмотреть список экземпляров взаимосвязей, соответствующих запросу, а также подробную информацию о каждом экземпляре
  - Изменить значения свойств экземпляра взаимосвязи
  - Создать или удалить экземпляр взаимосвязи
- Управлять ролями и экземплярами ролей:
  - Просмотреть сведения о роли или ее экземпляре
  - Изменить свойства экземпляра роли
  - Создать или удалить экземпляр роли для взаимосвязи

  - Выполнить откат данных экземпляра взаимосвязи до того момента времени, когда данные еще были правильными
- Импортировать данные из статической взаимосвязи в свою систему или экспортировать данные статической взаимосвязи в файл RI ли CSV
- Удалить схему и данные взаимосвязи из хранилища при удалении приложения, использовавшего эту взаимосвязь

Взаимосвязи в среде сетевого развертывания

Для использования взаимосвязей в среде сетевого развертывания (ND) не требуется выполнять никакую дополнительную настройку.

В средах сетевого развертывания (ND) взаимосвязи устанавливаются в кластере приложений. Эти взаимосвязи доступны во всем кластере, то есть любой сервер кластера может использовать данные экземпляра, хранящиеся в базе данных взаимосвязей. Возможность запуска службы взаимосвязей в среде ND повышает ее масштабируемость и готовность.

Администратор взаимосвязей позволяет управлять взаимосвязями из разных кластеров, используя единый административный интерфейс. Для подключения администратора взаимосвязей к серверу кластера необходимо выбрать его MBean взаимосвязи.

API службы взаимосвязей

Взаимосвязи можно вызывать внутри программ с помощью API службы взаимосвязей, используя их внутри связей бизнес-объектов или за их пределами.

Доступны три типа API:

- API для работы с экземплярами взаимосвязей (в том числе создания, обновления и удаления данных экземпляра напрямую)
- API для поддержки шаблонов взаимосвязей (в том числе correlate(), correlateforeignKeyLookup)
- Шаблоны поиска взаимосвязей (API поиска)

Шина служб предприятия в IBM Business Process Manager

IBM Business Process Manager поддерживает интеграцию служб приложений, в том числе весь набор функций продукта WebSphere Enterprise Service Bus.

Подключение служб через Enterprise Service Bus

С помощью Enterprise Service Bus (ESB) можно значительно повысить гибкость SOA. Участники взаимодействия служб соединяются с ESB, а не непосредственно друг с другом.

Когда клиент соединяется с ESB, ESB принимает на себя ответственность за доставку этих запросов, с помощью сообщений, провайдеру служб, предоставляющему требуемые функцию и Quality of Service. ESB обеспечивает взаимодействия клиента и провайдера и адресует несоответствующие протоколы, шаблоны взаимодействия или возможности служб. ESB также обеспечивает или расширяет мониторинг и управление. ESB предоставляет средства виртуализации и управления, которые реализуют и расширяют базовые возможности SOA.

ESB абстрагирует следующее:

Расположение и идентификация
: Участникам не нужно знать расположение или идентификатор других участников. Например, клиентам не требуется знать, каким из нескольких провайдеров обслуживается запрос; провайдеры служб могут добавляться и удаляться без прерывания.

Протокол взаимодействия
: Участникам не требуется совместно использовать один и тот же протокол связи или стиль

взаимодействия. Например, запрос, выраженный как SOAP через HTTP, может обслуживаться провайдером, который использует только SOAP через Java Message Service (JMS).

Интерфейс
Клиентам и провайдерам не нужно согласовывать общий интерфейс. ESB согласовывает различия преобразованием запроса и ответных сообщений в форму, ожидаемую провайдером.

Качества службы (взаимодействия)
Участники или системные администраторы объявляют свои требования к Quality of Service, включая идентификацию запросов, шифрование и дешифрование содержания сообщений, автоматический контроль взаимодействия служб и то, как их запросы должны маршрутизироваться (например, с оптимизацией скорости или стоимости).

Вставка ESB между участниками позволяет модулировать их взаимодействие через логическую конструкцию, называемую *передачей*. Передачи оперируют сообщениями на их пути между клиентами и провайдерами. Например, передачи могут использоваться для поиска служб с конкретными характеристиками, которые запрашивает клиент, или для разрешения различий в интерфейсах между клиентами и провайдерами. Для сложных взаимодействий передачи могут соединяться последовательно.

С помощью передач Enterprise Service Bus выполняет следующие действия между клиентом и службой:

- *Маршрутизация* сообщений между службами. Enterprise Service Bus предоставляет общую инфраструктуру связей, которую можно использовать для соединений служб, и тем самым бизнес-функций, которые они представляют, без необходимости написания и обслуживания сложной логики связей программистами.
- *Преобразование* протоколов передачи между клиентом и службой. Enterprise Service Bus обеспечивает основанный на стандартах совместимый способ интеграции бизнес-функций, которые используют разные стандарты ИТ. Он предусматривает интеграцию бизнес-функций, которые обычно несовместимы, например, соединение приложений в подразделенческих отделениях или предоставление приложений другим компаниям для участия в служебных взаимодействиях.
- *Преобразование* форматов сообщений между клиентом и службой. Enterprise Service Bus позволяет бизнес-функциям изменять информацию в разных форматах с помощью шины, обеспечивающей, чтобы информация, доставляемая бизнес-функции, была в формате, требуемом этим приложением.
- *Обработка* бизнес-событий из разных источников. Enterprise Service Bus поддерживает основанные на событиях взаимодействия, кроме обмена сообщениями для обработки запросов служб.

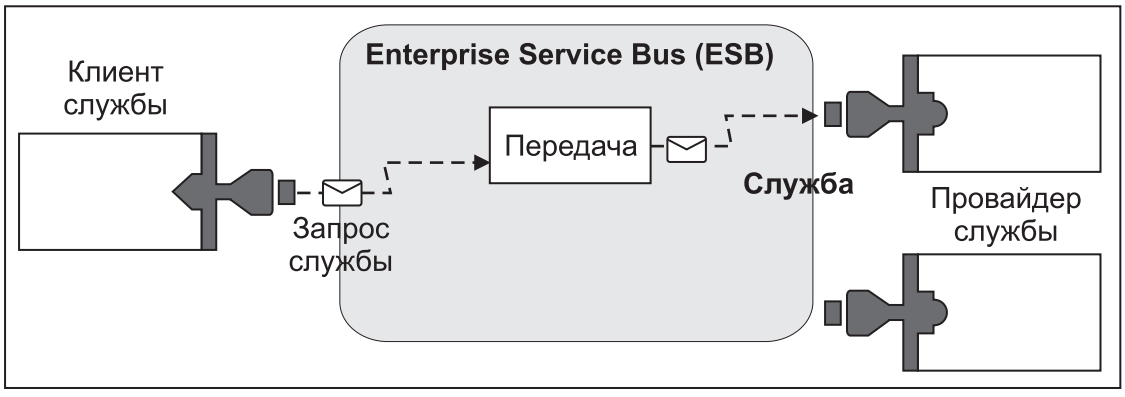


Рисунок 42. Enterprise Service Bus. Enterprise Service Bus направляет сообщения между приложениями, которыми являются клиенты или провайдеры служб. Эта шина преобразует протоколы передачи и форматы сообщений между клиентами и провайдерами. На следующем рисунке каждое приложение использует свой протокол (показанный разными геометрическими формами их соединителей) и свой формат сообщений.

Использование Enterprise Service Bus позволяет сосредоточиться на основной деятельности и не отвлекаться на свои компьютерные системы. При необходимости можно изменять и добавлять службы. Например, в ответ на изменения бизнес-требования можно предоставить дополнительный ресурс службы или добавить

новые возможности. Необходимые изменения можно вносить перенастройкой шины без влияния или с небольшим влиянием на существующие службы и приложения, использующие шину.

Инфраструктура обмена сообщениями шины служб предприятия

IBM Business Process Manager предоставляет функции шины служб предприятия. IBM Business Process Manager поддерживает интеграцию технологий, основанных на службах, сообщениях, а также технологий, управляемых событиями, для предоставления инфраструктуры обмена сообщениями, основанной на стандартах, в интегрированной шине служб предприятия.

Функции служб предприятия, которые можно использовать в своем приложении предприятия, предоставляют не только транспортный уровень, но и поддержку средств передачи данных для обеспечения взаимодействия служб. Шина служб предприятия основана на открытых стандартах и архитектуре на основе служб (SOA). Она использует надежную инфраструктуру Java EE и связанные с ней службы платформы, предоставляемые IBM WebSphere Application Server Network Deployment.

IBM Business Process Manager включает в себя аналогичную технологию, предоставляемую IBM WebSphere Enterprise Service Bus. Эта функция входит в состав IBM Business Process Manager, поэтому для ее применения не требуется отдельная лицензия на WebSphere Enterprise Service Bus.

Однако в среде предприятия можно установить дополнительные автономные лицензии на продукт WebSphere Enterprise Service Bus для расширения набора средств связи в решениях интеграции процессов, основанных на IBM Business Process Manager. Например, WebSphere Enterprise Service Bus можно установить рядом с приложением SAP для обслуживания IBM WebSphere Adapter for SAP и преобразования сообщений SAP перед их отправкой по сети в бизнес-процессы, созданные с помощью IBM Business Process Manager.

Целевые хосты сообщений и очередей:

Целевые хосты сообщений или очередей обеспечивают функцию обмена сообщениями на сервере. Для того чтобы сервер стал целевым хостом сообщений, настройте его в качестве адресата сообщений.

На сервере работает модуль обмена сообщениями. Модуль обмена сообщениями предоставляет функции обмена сообщениями и точку подключения для приложений, которые подключаются к шине. Асинхронная связь SCA, импорт и экспорт JMS, а также асинхронная внутренняя обработка используют очереди модуля обмена сообщениями.

Среда развертывания подключает адресанта к адресату сообщений через шину при развертывании модулей приложений. Если известен адресант и адресат сообщений, можно определить необходимый тип среды развертывания.

Приложения могут хранить постоянные данные в хранилище данных, которое представляет собой набор таблиц в базе данных/схеме или в хранилище файлов. Модуль обмена сообщениями использует экземпляр источника данных JDBC для взаимодействия с этой базой данных.

Настройте целевой хост сообщений при определении среды развертывания, выбрав пункт **Сервер** в административной консоли или назначив сервер целевым хостом при установке ПО.

Хранилища данных:

Каждый модуль обмена сообщениями может использовать хранилище данных, представляющее собой набор таблиц базы данных или схему, хранящую постоянные данные.

Все таблицы хранилища данных расположены в одной схеме таблиц базы данных. Каждое хранилище данных можно создать в отдельной базе данных. Вместо этого можно создать несколько хранилищ данных в одной базе данных так, чтобы каждое хранилище использовало свою схему.

Модуль обмена сообщениями использует экземпляр источника данных JDBC для взаимодействия с базой данных, содержащей хранилище данных этого модуля.

Комплексы связи JDBC:

Комплексы связи JDBC можно использовать для взаимодействия приложений с реляционными базами данных.

Приложения используют комплексы связи JDBC для взаимодействия с реляционными базами данных. Комплекс связи JDBC сообщает определенный класс реализации драйвера JDBC для доступа к определенному типу баз данных. Для того чтобы создать пул подключений к этой базе данных, свяжите источник данных с комплексом связи JDBC. Комплекс связи JDBC и объекты источника данных функционируют вместе подобно фабрике соединений Java EE Connector Architecture (JCA), обеспечивающей соединение с нереляционными базами данных.

Примеры установки типовой автономной среды и типовой среды развертывания приведены в предыдущем разделе.

Дополнительная информация о комплексах связи JDBC приведена в разделе “Комплексы связи JDBC” справочной системы Information center WebSphere Application Server.

Шина интеграции служб для IBM Business Process Manager:

Шина интеграции служб - управляемый механизм связи, поддерживающий интеграцию служб путем синхронного и асинхронного обмена сообщениями. Шина состоит из взаимосвязанных модулей обмена сообщениями, которые управляют ресурсами шины. Это одна из технологий WebSphere Application Server, на которой основывается IBM Business Process Manager.

Одиночная шина интеграции служб, как и одиночный модуль обмена сообщениями, использует ту же схему базы данных, которая применяется в базе данных продукта по умолчанию. У каждой среды развертывания имеется собственная шина. Одиночная шина имеет имя **BPM.*имя-среды-развертывания*.Bus**.

Целевой объект шины - это логический адрес, к которому приложение может обратиться как источник и/или приемник данных. Целевой объект очереди - целевое объект шины, используемый для двухточечного обмена сообщениями.

Каждая шина может включать один или несколько элементов, каждый из которых является сервером или кластером.

Топология шины - физическое размещение серверов приложений, служб обмена сообщениями и администраторов очередей WebSphere MQ, а также шаблонов соединений шины и ссылок между ними, образующих шину служб предприятия.

Служебные приложения и служебные модули

Служебный модуль - это модуль SCA, предоставляющий службы в среде выполнения. При развертывании служебного модуля в IBM Business Process Manager выполняется компоновка связанного служебного приложения, которое упаковывается в файл EAR.

Служебные модули являются базовыми блоками развертывания и могут содержать компоненты, библиотеки и промежуточные модули, используемые связанным служебным приложением. Служебные модули имеют экспорты и могут иметь импорты для определения взаимосвязей между модулями и клиентами и провайдерами служб. WebSphere Process Server поддерживает модули для бизнес-служб и модули передачи. Как модули, так и модули передачи являются типами модулей SCA. Модуль передачи обеспечивает связь между приложениями путем преобразования вызова службы в формат, понятный целевому объекту, передачи запроса целевому объекту и возврата результата отправителю. Модуль для

бизнес-службы реализует логику бизнес-процесса. Однако модуль может также включать ту же логику передачи, которая может быть упакована в модуле передачи.

Развертывание служебного приложения

Процесс развертывания файла EAR, содержащего служебное приложение, ничем не отличается от процесса развертывания любого файла EAR. Во время развертывания можно изменить значения параметров передачи. После развертывания файла EAR, содержащего модуль SCA, можно просмотреть сведения о служебном приложении и связанном с ним модулем. Можно посмотреть, как служебный модуль соединяется с клиентами служб (через экспорты) и провайдерами служб (через импорты).

Просмотр сведений о модуле SCA

Сведения о служебном модуле, доступные для просмотра, зависят от модуля SCA. В них предусмотрены следующие атрибуты.

- Имя модуля SCA
- Описание модуля SCA
- Имя связанного приложения
- Информация о версии модуля SCA, если модуль имеет разные версии
- Импорты модуля SCA:
  - Интерфейсы импорта являются абстрактными определениями, которые описывают, как модуль SCA обращается к службе.
  - Привязки импортов являются конкретными определениями, которые описывают физический механизм, с помощью которого модуль SCA обращается к службе. Например, с помощью SOAP/HTTP.
- Экспорты модуля SCA:
  - Интерфейсы экспортов являются абстрактными определениями, которые описывают, как клиенты служб обращаются к модулю SCA.
  - Привязки экспортов являются конкретными определениями, которые описывают физический механизм, с помощью которого клиент службы обращается к модулю SCA и, косвенно, к службе.
- Свойства модуля SCA

Импорт и привязка импорта:

Импорт определяет взаимодействие между модулями архитектуры компонентов служб (SCA) и провайдерами служб. Модули SCA используют импорты для обеспечения доступа компонентов к внешним службам (за пределами модуля SCA) с использованием локального представления. Привязки импортов определяют конкретный способ обращения к внешней службе.

Если модулям SCA не требуется обращение к внешним службам, то им не нужны и импорты. Модули передачи обычно имеют один или несколько импортов, которые используются для передачи сообщений или запросов назначенным целевым объектам.

Интерфейсы и привязки

Импорту модуля SCA требуется по крайней мере один интерфейс, и импорт модуля SCA имеет одну привязку.

- Интерфейсы импортов являются абстрактными определениями, обозначающими некоторый набор операций, использующих язык описания веб-служб (WSDL), язык XML для описания веб-служб. Модуль SCA может иметь много интерфейсов импортов.
- Привязки импортов являются конкретными определениями, которые обозначают физический механизм, который модули SCA используют для доступа к внешней службе.

Поддерживаемые привязи импортов

IBM Business Process Manager поддерживает следующие привязки импортов:

- Привязки SCA связывают модули SCA с другими модулями SCA. Привязки SCA также указываются как привязки по умолчанию.
- Привязки веб-служб позволяют компонентам вызывать веб-службы. Поддерживаются протоколы SOAP1.1/HTTP, SOAP1.2/HTTP и SOAP1.1/JMS.

  Можно использовать привязку SOAP1.1/HTTP или SOAP1.2/HTTP, основанную на Java API for XML Web Services (JAX-WS), которая предусматривает взаимодействие со службами с использованием документа или литеральных привязок RPC и которая использует обработчики JAX-WS для настройки вызовов. Отдельная привязка SOAP1.1/HTTP предусматривается для обеспечения взаимодействия со службами, которые используют привязку в коде RPC или где существует требование использования обработчиков JAX-RPC для настройки вызовов.
- Привязки HTTP допускают обращение к приложениям с помощью протокола HTTP.
- Привязки импортов Enterprise JavaBeans (EJB) позволяют компонентам SCA вызывать службы, предоставляемые бизнес-логикой Java EE, выполняемой на сервере Java EE.
- Привязки информационной системы предприятия (EIS) обеспечивают связь между компонентами SCA и внешней EIS. Эта связь обеспечивается использованием адаптеров ресурсов.
- Привязки Java Message Service (JMS) 1.1 допускают взаимодействие с провайдером обмена сообщениями WebSphere Application Server по умолчанию. JMS может использовать разные типы передачи, включая TCP/IP и HTTP или HTTPS. Класс сообщений JMS и его пять подтипов (Текст, Байты, Объект, Поток и Преобразование) автоматически поддерживаются.
- Generic JMS bindings permit interoperability with third-party JMS providers that integrate with the WebSphere Application Server using the JMS Application Server Facility (ASF).
- Привязки WebSphere MQ JMS допускают взаимодействие с провайдерами JMS на основе WebSphere MQ. Класс сообщений JMS и его пять подтипов (Текст, Байты, Объект, Поток и Преобразование) автоматически поддерживаются. Для использования WebSphere MQ в качестве провайдера JMS используйте привязки WebSphere MQ JMS.
- Привязки WebSphere MQ допускают взаимодействие с WebSphere MQ. Привязки WebSphere MQ можно использовать только с удаленными администраторами очередей через соединение с клиентом WebSphere MQ; они не могут использоваться с локальными администраторами очередей. Для соединения с внутренними приложениями WebSphere MQ используйте привязки WebSphere MQ.

Динамический вызов служб

Службы могут через любую поддерживаемую привязку импорта. Служба обычно находится в конечной точке, указанной в импорте. Эта конечная точка вызывается статической конечной точкой. Другую службу можно вызвать переопределением статической конечной точки. Динамическое переопределение статических конечных точек позволяет вызвать службу в другой конечной точке через любую поддерживаемую привязку импорта. Динамический вызов служб позволяет вызвать службу, где поддерживаемая привязка импорта не имеет статической конечной точки.

Импорт со связанной привязкой используется для задания протокола и его конфигурации для динамического вызова. Импорт, используемый для динамического вызова, может быть соединен с вызывающим компонентом или динамически выбран во время выполнения.

Для веб-службы и вызовов SCA также возможно сделать динамический вызов без импорта с помощью протокола и конфигурации, извлеченных из URL конечной точки. Тип целевого объекта вызова определяется из URL конечной точки. Если используется импорт, то URL должен быть совместим с протоколом привязки импорта.

- URL SCA указывает вызов другого модуля SCA.

- URL HTTP или JMS по умолчанию указывает вызов веб-службы; для этих URL можно предоставить дополнительное значение типа привязки, которое указывает, что URL представляет вызов через привязку HTTP или JMS.
- Для URL HTTP веб-службы по умолчанию используется SOAP 1.1 и может быть задано значение типа привязки, которое указывает использование SOAP 1.2.

Экспорт и привязка экспорта:

Экспорт определяет взаимодействие между модулями архитектуры компонентов служб (SCA) и клиентами служб. Модули SCA используют экспорты для предоставления служб другим. Привязки экспортов определяют конкретный способ обращения клиентов служб к модулю SCA.

Интерфейсы и привязки

Экспорту модуля SCA требуется по крайней мере один интерфейс.

- Интерфейсы экспортов являются абстрактными определениями, обозначающими некоторый набор операций, использующих язык описания веб-служб (WSDL), язык XML для описания веб-служб. Модуль SCA может иметь много интерфейсов экспортов.
- Привязки экспортов являются конкретными определениями, которые обозначают физический механизм, который клиенты служб используют для доступа к службе. Обычно экспорт модуля SCA имеет одну заданную привязку. Экспорт без заданной привязки интерпретируется средой выполнения как экспорт с привязкой SCA.

Поддерживаемые привязи экспортов

IBM Business Process Manager поддерживает следующие привязки экспортов:

- Привязки SCA связывают модули SCA с другими модулями SCA. Привязки SCA также указываются как привязки по умолчанию.
- Привязки веб-служб допускают вызов экспортов как веб-служб. Поддерживаются протоколы SOAP1.1/HTTP, SOAP1.2/HTTP и SOAP1.1/JMS.

  Можно использовать привязку SOAP1.1/HTTP или SOAP1.2/HTTP, основанную на Java API for XML Web Services (JAX-WS), которая предусматривает взаимодействие со службами с использованием документа или литеральных привязок RPC и которая использует обработчики JAX-WS для настройки вызовов. Отдельная привязка SOAP1.1/HTTP предусматривается для обеспечения взаимодействия со службами, которые используют привязку в коде RPC или где существует требование использования обработчиков JAX-RPC для настройки вызовов.
- Привязки HTTP допускают обращение к экспортам с помощью протокола HTTP.
- Привязки экспортов Enterprise JavaBeans (EJB) допускают представление компонентов SCA как EJB, чтобы бизнес-логика Java EE могла вызывать компоненты SCA, которые в противном случае недоступны ей.
- Привязки информационной системы предприятия (EIS) обеспечивают связь между компонентами SCA и внешней EIS. Эта связь обеспечивается использованием адаптеров ресурсов.
- Привязки Java Message Service (JMS) 1.1 допускают взаимодействие с провайдером обмена сообщениями WebSphere Application Server по умолчанию. JMS может использовать разные типы передачи, включая TCP/IP и HTTP или HTTPS. Класс сообщений JMS и его пять подтипов (Текст, Байты, Объект, Поток и Преобразование) автоматически поддерживаются.
- Базовые привязки JMS допускают взаимодействие с внешними провайдерами, которые интегрируются с WebSphere Application Server с помощью JMS Application Server Facility (ASF).
- Привязки WebSphere MQ JMS допускают взаимодействие с провайдерами JMS на основе WebSphere MQ. Класс сообщений JMS и его пять подтипов (Текст, Байты, Объект, Поток и Преобразование) автоматически поддерживаются. Для использования WebSphere MQ в качестве провайдера JMS используйте привязки WebSphere MQ JMS.
- Привязки WebSphere MQ допускают взаимодействие с WebSphere MQ. Для соединения с администратором очередей MQ в удаленной системе используется удаленное соединение (или клиент). Локальное соединение

(или привязки) являются прямым соединением с WebSphere MQ. Оно может использоваться для соединения с администратором очередей MQ только в той же системе. WebSphere MQ предусматривает оба типа соединений, но привязки MQ поддерживают только "удаленное" (или "клиентское") соединение.

Модули передачи:

Модули передачи являются модулями архитектуры компонентов служб (SCA), которые могут изменять формат, содержимое или целевой объект запросов служб.

Модули передачи работают с сообщениями, которые находятся на пути между клиентами служб и провайдерами служб. Можно направлять сообщения другим провайдерам служб и изменять содержимое и формат сообщений. Модули передачи могут предоставлять такие функции, как ведение протокола и обработка ошибок, которые отвечают вашим требования.

С помощью административной консоли можно изменять определенные аспекты модулей передачи без необходимости повторного развертывания модуля.

Компоненты модулей передачи

Модули передачи содержат следующие элементы:

- Импорты, которые определяют взаимодействия между модулями SCA и провайдерами служб. Они позволяют модулям SCA вызывать внешние службы так, как если бы они были локальными. Можно просматривать импорты модуля передачи и изменять привязку.
- Экспорты, которые определяют взаимодействия между модулями SCA и клиентами служб. Они позволяют модулю SCA предлагать службу и определять внешние интерфейсы (точки доступа) модуля SCA. Экспорты модуля передачи можно просматривать.
- Компоненты SCA, которые создают блоки для модулей SCA, таких как модули передачи. Модули и компоненты SCA можно создавать и настраивать графически с помощью Integration Designer. После развертывания с помощью административной консоли можно настраивать определенные аспекты модуля передачи без необходимости повторного развертывания модуля.

  Обычно модули передачи содержат специальный тип компонента SCA, называемого *компонентом потока передачи*. Компоненты потока передачи определяют потоки передачи.

  Компонент потока передачи может не содержать или содержать один или несколько примитивов передачи. IBM Business Process Manager поддерживает поставляемый набор примитивов передачи, который предоставляет функциональные возможности для маршрутизации и преобразования сообщений. Для получения дополнительных функциональных возможностей примитивов передачи можно применить пользовательский примитив передачи для вызова пользовательской логики.

  Назначением модуля передачи, который не содержит компонента потока передачи, является преобразование запросов служб из одного протокола в другой. Например, запрос службы может быть сделан с помощью SOAP/JMS, но может требовать преобразования в SOAP/HTTP перед отправкой.

  Примечание: Модули передачи можно просматривать и вносить определенные изменения в них с помощью IBM Business Process Manager. Однако IBM Business Process Manager не позволяет просматривать или изменять компоненты SCA в пределах модуля. Для настройки компонентов SCA используйте Integration Designer.

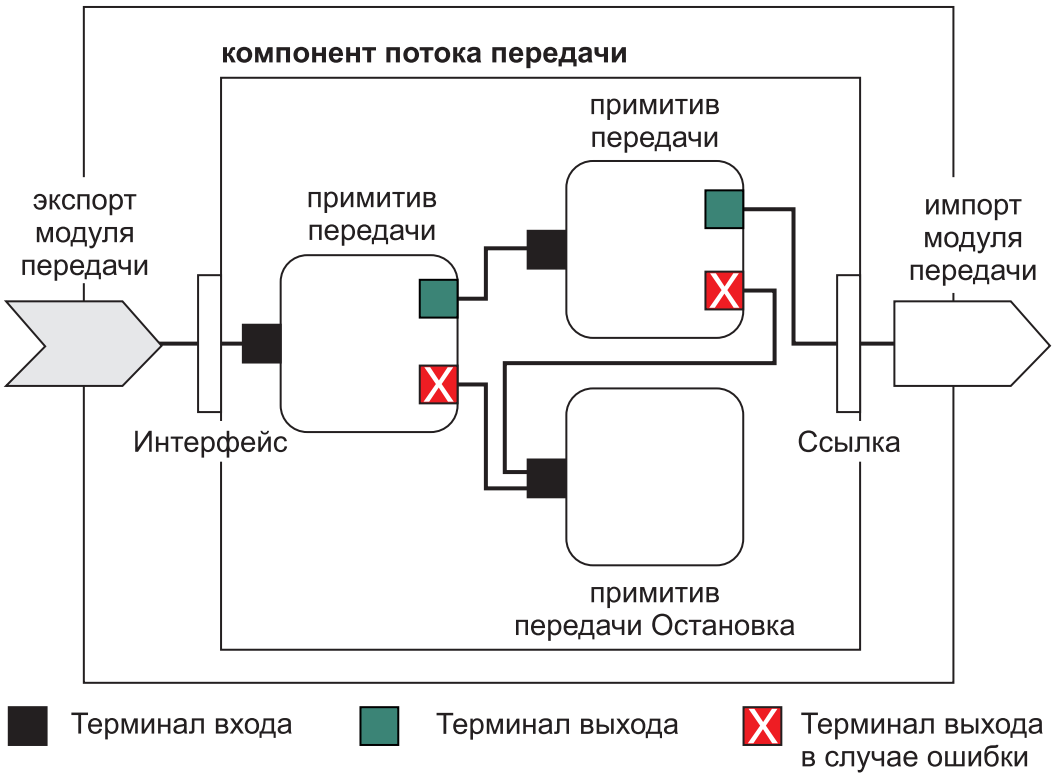


Рисунок 43. Упрощенный пример модуля передачи. Модуль передачи содержит один компонент потока передачи, который содержит примитивы передачи.

- Свойства

  Примитивы передачи имеют свойства, некоторые из которых могут отображаться в административной консоли как дополнительные свойства модуля SCA.

  В случае свойств примитива передачи в административной консоли IBM Business Process Manager разработчик интеграции должен открыть эти свойства. Некоторые свойства сами предоставляют себя для административной настройки, и Integration Designer описывает эти открываемые свойства, поскольку они могут быть открыты из цикла интеграции в административном цикле. Другие свойства не готовы к административной настройке, поскольку их изменение может повлиять на поток передачи таким образом, что модуль передачи потребует повторного развертывания. Integration Designer перечисляет свойства, которые можно выбрать для открытия в открытых свойствах примитива передачи.

  Административную консоль IBM Business Process Manager можно использовать для изменения значения открытых свойств без необходимости повторного развертывания модуля передачи или перезапуска сервера или модуля.

  Обычно потоки передачи используют изменения свойств сразу же. Однако если изменения свойств делаются в ячейке администратора развертывания, то они могут влиять на каждый узел. поскольку данный узел синхронизируется. Кроме того, потоки передачи, которые находятся в процессе, продолжают использовать предыдущие значения.

  Примечание: Из административной консоли можно изменять только значения свойств, но не группы, имена или типы свойств. При необходимости изменения групп, имен или типов свойств используйте Integration Designer.
- Модуль передачи или зависимая библиотека может также определять подпотоки. Подпоток представляет собой повторно используемую единицу логики интеграции и состоит из соединенных друг с другом примитивов передачи. Примитив можно добавлять в поток передачи для вызова подпотока.

Развертывание модулей передачи

Модули передачи создаются с помощью Integration Designer и обычно развертываются в IBM Business Process Manager внутри файла EAR.

Значения открытых свойств можно изменить во время развертывания.

Можно экспортировать модуль из Integration Designer и вызвать Integration Designer для упаковки модуля передачи в файл архива Java (JAR) и файл JAR внутри файла EAR. Затем можно развернуть файл EAR, установив новое приложение из административной консоли.

Модули передачи можно рассматривать как одну сущность. Однако модули SCA определяются несколькими файлами XML, хранящимися в файле JAR.

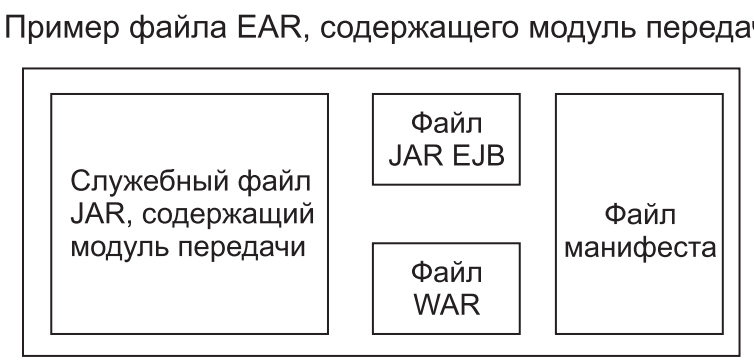


Рисунок 44. Упрощенный пример файла EAR, содержащего модуль передачи. Файл EAR содержит файлы JAR. Файл JAR утилит содержит модуль передачи.

Примитивы передачи:

Компоненты потока передачи работают на потоках сообщений между компонентами служб. Возможности компонента передачи реализуются *примитивами передачи*, которые реализуют стандартные типы реализаций служб.

Компонент потока передачи имеет один или несколько потоков. Например, один для запроса и один для ответа.

IBM Business Process Manager поддерживает поставляемый набор примитивов передачи, которые реализуют стандартные возможности передачи для модулей передачи или модулей, развернутых в IBM Business Process Manager. Когда требуются специальные возможности передачи, можно развернуть свои пользовательские примитивы передачи.

Примитив передачи определяет “входящую” операцию, обрабатывающую сообщения, которые представляются объектами сообщений служб (SMO). Примитив передачи также может определять “исходящую” операцию, которая отправляет сообщения другому компоненту или модулю.

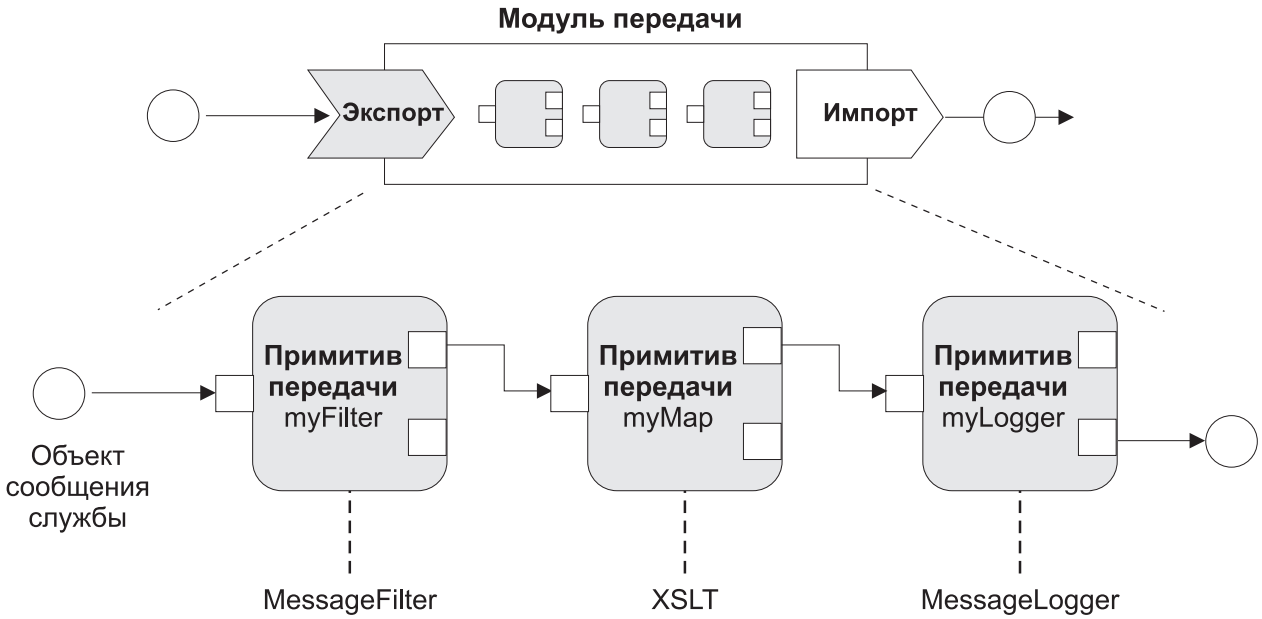


Рисунок 45. Модуль передачи, содержащий три примитива передачи

Integration Designer можно использовать для настройки примитивов передачи и задания их свойств. Некоторые из этих свойств можно сделать видимыми администратору среды выполнения открытием их. Любое свойство примитива передачи, которое может быть открыто, также может быть динамическим свойством. Динамическое свойство можно переопределять во время выполнения с помощью файла стратегий.

Integration Designer также позволяет графически моделировать и производить сборку компонентов потока передачи из примитивов передачи, а также собирать модули передачи или модули из компонентов потока передачи. Административная консоль ссылается на модули передачи и модули как на модули SCA.

Integration Designer также предусматривает определение подпотоков в модулях или их зависимых библиотеках. Подпоток может содержать любой примитив передачи за исключением примитива передачи обработки стратегий. Подпоток вызывается из потока запроса или ответа или из другого подпотока с помощью примитива передачи подпотока. Свойства, открытые из примитивов передачи в подпотоке, представляются как свойства на примитивах передачи подпотока. Они затем могут открываться снова, пока они не достигнут уровня модуля, на котором они затем могут быть изменены администратором среды выполнения.

Поддерживаемые примитивы передачи

IBM Business Process Manager поддерживает следующий набор примитивов передачи:

Карта бизнес-объектов

Преобразует сообщения.

- Определяет преобразования сообщений с помощью карты бизнес-объектов, которая может использоваться многократно.
- Позволяет определять преобразования сообщений графически с помощью редактора карты бизнес-объекта.
- Может изменять содержимое сообщения.
- Может преобразовать тип входящего сообщения в другой тип исходящего сообщения.

Пользовательская передача

Позволяет реализовать пользовательскую логику передачи в коде Java. Пользовательский примитив передачи объединяет гибкость определенного пользователем примитива передачи с простотой готового примитива передачи. Можно создавать сложные преобразования и шаблоны маршрутизации путем:

- Создания кода Java.
- Создания собственных свойств.
- Добавления новых терминалов.

Службу можно вызвать из пользовательского примитива передачи, а примитив передачи вызовов служб предназначен для вызова служб и предоставляет дополнительную возможность, такую как повторение.

Обработчик данных

Позволяет преобразовать часть сообщения. Он используется для преобразования некоторого элемента сообщения из физического формата в логическую структуру или из логической структуры в физический формат. Основным применением примитива является преобразование физического формата, такого как текстовая строка, в пределах объекта текстового сообщения JMS в логическую структуру бизнес-объекта и обратно. Эта передача обычно используется для следующего:

- Преобразование некоторого раздела входящего сообщения из определенной структуры в другую. Например, когда SMO включает строковое значение, отделенное запятой, и требуется проанализировать его в конкретном бизнес-объекте.
- Изменение типа сообщения. Для примера можно привести ситуацию, когда экспорт JMS настроен на использование привязки данных базового типа JMS, но в пределах модуля передачи разработчик интеграции решает, что это содержимое должно быть включено в конкретную структуру BO.

Поиск в базе данных

Изменяет сообщения, используя информацию из предоставленной пользователем базы данных.

- Необходимо настроить базу данных, источник данных и все параметры идентификации сервера для используемого примитива передачи Поиск в базе данных. В этом может помочь административная консоль.
- Примитив передачи Поиск в базе данных может выполнить чтение только одной таблицы.
- Указанный ключевой столбец должен содержать уникальное значение.
- Данные в столбцах значений должны быть либо простого типа схемы XML, либо типа схемы XML, расширенного простым типом схемы XML.

Поиск конечных точек

Предусматривает динамическую маршрутизацию запросов поиском конечных точек служб в хранилище.

- Информация о конечных точках служб извлекается из WebSphere Service Registry and Repository (WSRR). Реестр WSRR может быть локальным или удаленным.
- Изменения реестра делаются через административную консоль WSRR.
- IBM Business Process Manager необходимо знать, какой реестр используется, поэтому необходимо создать определения доступа WSRR с помощью административной консоли IBM Business Process Manager.

Отправитель событий

Расширяет мониторинг, позволяя отправлять события изнутри компонента потока передачи.

- Можно остановить действие передачи, выключив переключатель.
- Можно просмотреть события обработчика событий с помощью браузера событий общего формата IBM Business Process Manager.
- События необходимо отправлять только в значимую точку потока передачи из соображений производительности.

- Можно указать части сообщения, которые содержит событие.
- События отправляются в формате событий общего формата и отправляются на сервер инфраструктуры событий общего формата.
- Для полного использования информации отправителя событий получателям событий необходимо понимать структуру событий общего формата. События общего формата имеют общую схему, но это не модель конкретных данных приложения, которая содержится в расширенных элементах данных. Для моделирования расширенных элементов данных инструменты Integration Designer создают файл определений каталога событий инфраструктуры событий общего формата для каждого из настроенных примитивов передачи Отправитель событий. Файлы определений каталога событий являются артефактами экспортов, которые предоставляются в помощь; они не используются Integration Designer или средой выполнения IBM Business Process Manager. Файлы определений каталога событий должны указываться при создании приложений для использования событий отправителя событий.
- Можно указать другой мониторинг из IBM Business Process Manager. Например, можно отслеживать события, отправляемые из импортов и экспортов.

Ошибка

Останавливает некоторый конкретный путь в потоке и создает исключительную ситуацию.

Объединение

Позволяет объединять (комбинировать) сообщения.

- Может использоваться только с примитивом передачи Разъединение.
- Вместе примитивы передачи Разъединение и Объединение позволяют объединять данные в одном исходящем сообщении.
- Примитив передачи Объединение получает сообщения до точки принятия решения, затем отправляется одно сообщение.
- Для удержания объединенных данных должен использоваться общий контекст.

Разъединение

Позволяет разделять и объединять (комбинировать) сообщения.

- Вместе примитивы передачи Разъединение и Объединение позволяют объединять данные в одном исходящем сообщении.
- В режиме повторения примитив передачи Разъединение позволяет повторять одно входящее сообщение, которое содержит повторяющийся элемент. При каждом появлении повторяющегося элемента отправляется сообщение.
- Для удержания объединенных данных должен использоваться общий контекст.

Метод set заголовков HTTP

Предоставляет механизм управления заголовками в сообщениях HTTP.

- Может создавать, задавать, копировать или удалять заголовки сообщений HTTP.
- Может задавать несколько действий для изменения нескольких заголовков HTTP.

Преобразование

Преобразует сообщения.

- Позволяет выполнять преобразования расширяемого языка таблицы стилей (XSL) или преобразования по карте связей бизнес-объектов.
- Сообщения преобразуются с использованием преобразования XSLT 1.0, XSLT 2.0 или преобразования по карте связей бизнес-объектов. Преобразование XSL применяется к сообщению, сериализованному в формат XML; преобразование по карте связей бизнес-объектов применяется к объектам данных службы (SDO).

Метод set элементов сообщения

Предоставляет простой механизм задания содержимого сообщений.

- Может изменять, добавлять или удалять элементы сообщения.
- Не изменяет тип сообщения.

- Данные в столбцах значений должны быть либо простого типа схемы XML, либо типа схемы XML, расширенного простым типом схемы XML.

Фильтр сообщений

Направляет сообщения на различные пути на основании содержания сообщения.

- Можно остановить действие передачи, выключив переключатель.

Регистратор сообщений

Регистрирует сообщения в реляционной базе данных или через пользовательский регистратор. Сообщения сохраняются как XML, поэтому данные впоследствии могут обрабатываться приложениями XML.

- Можно остановить действие передачи, выключив переключатель.
- Схема реляционной базы данных (структура таблиц) определяется IBM.
- По умолчанию примитив передачи Регистратор сообщений использует общую базу данных. Среда выполнения преобразует источник данных в **jdbc/mediation/messageLog** в общую базу данных.
- Для настройки работы пользовательского обработчика можно настроить классы реализации Обработчика. Также для настройки работы пользовательского регистратора можно предоставить классы реализации программы форматирования, классы реализации фильтра или того и другого.

Метод set заголовков MQ

Предоставляет механизм управления заголовками в сообщениях MQ.

- Может создавать, задавать, копировать или удалять заголовки сообщений MQ.
- Может задавать несколько действий для изменения нескольких заголовков MQ.

Обработка стратегий

Обеспечивает динамическую настройку запросов путем поиска конечных точек служб и связанных файлов стратегий в хранилище.

- Файл стратегий можно использовать для динамического переопределения открытых свойств других примитивов передачи.
- Информация о конечных точках служб и информация о стратегиях извлекается из WebSphere Service Registry and Repository (WSRR). Реестр WSRR может быть локальным или удаленным.
- Изменения реестра делаются через административную консоль WSRR.
- IBM Business Process Manager необходимо знать, какой реестр используется, поэтому необходимо создать определения доступа WSRR с помощью административной консоли IBM Business Process Manager.

Вызов службы

Вызывает службу изнутри потока передачи вместо ожидания конца потока передачи и использования механизма вызова.

- Если служба возвращает ошибку, можно повторить вызов той же службы или вызвать другую службу.
- Примитив передачи Вызов службы является мощным примитивом передачи, который может использоваться сам по себе для простых вызовов служб или в комбинации с другими примитивами передачи для сложных передач.

Задание типа сообщения

Во время разработки интеграции позволяет рассматривать слаботипизированные поля сообщения как строготипизированные. Поле является слаботипизированным, если оно может содержать более одного типа данных. Поле является строготипизированным, если его тип и внутренняя структура известны.

- Во время выполнения примитив передачи Задание типа сообщения позволяет проверить, соответствует ли содержимое сообщения ожидаемым типам данных.

Метод set заголовков SOAP

Предоставляет механизм управления заголовками в сообщениях SOAP.

- Может создавать, задавать, копировать или удалять заголовки сообщений SOAP.
- Может задавать несколько действий для изменения нескольких заголовков SOAP.

Остановка
Останавливает некоторый конкретный путь в потоке без создания исключительной ситуации.

Фильтр типов
Позволяет направить сообщения по другому пути потока на основании их типа.

WebSphere eXtreme Scale - Получить
Можно получить информацию из среды кэша сервера eXtreme Scale.

- Можно найти значения в кэше и хранить их как элементы в сообщении с использованием ключа.
- Сочетая примитивы передачи Хранение и Получение eXtreme Scale, можно кэшировать ответы из системы базового сервера каталогов. Для будущих запросов не потребуется получение прав доступа к этой системе базового сервера каталогов.
- Необходимо создать определения eXtreme Scale с помощью административной консоли WebSphere ESB, чтобы можно было задать сервер eXtreme Scale для использования.

WebSphere eXtreme Scale - Сохранить
Можно сохранить информацию в среде кэша сервера eXtreme Scale.

- Можно сохранить информацию в кэше eXtreme Scale с помощью ключа и объекта.
- Сочетая примитивы передачи Сохранить и Получить eXtreme Scale, можно использовать примитив передачи Сохранить для сохранения данных в кэше, а примитив передачи Получить применять для получения данных, сохраненных ранее в кэше.
- Необходимо создать определения eXtreme Scale с помощью административной консоли WebSphere ESB, чтобы можно было задать сервер eXtreme Scale для использования.

Динамическая маршрутизация:

Можно направлять сообщения различными путями, используя конечные точки, определенные во время интеграции, или конечные точки определенные динамически во время выполнения.

К динамической маршрутизации относятся два случая маршрутизации сообщений:

- Динамическая маршрутизация, в которой поток динамический, но все возможные конечные точки предопределены в модуле архитектуры компонентов служб (SCA).
- Динамическая маршрутизация, в которой поток динамический и выбор конечных точек также динамический. Конечные точки служб выбираются из внешнего источника во время выполнения

Динамический выбор конечной точки

Среда выполнения имеет возможность направлять сообщение-запрос и ответное сообщение в адрес конечной точки, определенный элементом заголовка сообщения. Этот элемент заголовка сообщения может обновляться примитивами передачи в потоке передачи. Адрес конечной точки может обновляться информацией из реестра, базы данных или информацией из самого сообщения. Маршрутизация ответных сообщений применяется, только когда ответ отправляется экспортом JAX-WS веб-службы.

Для того чтобы среда выполнения реализовывала динамическую маршрутизацию запроса или ответа, модуль SCA должен иметь заданное свойство `Использовать динамическую конечную точку при задании в заголовке сообщения`. Разработчики интеграции могут задать свойство `Использовать динамическую конечную точку при задании в заголовке сообщения` или могут открыть его (сделать видимым в среде выполнения), чтобы администратор мог задать его. Свойства модуля можно просмотреть в окне Свойства модуля. Для вызова окна выберите **Приложения > Модули SCA > Свойства модулей**. Разработчик интеграции дает имена-псевдонимы открытым свойствам, и эти имена показываются в административной консоли.

Реестр

IBM WebSphere Service Registry and Repository (WSRR) можно использовать для хранения информации о конечных точках служб и затем создавать модули SCA для извлечения конечных точек из реестра WSRR.

При разработке модулей SCA используйте примитив передачи Поиск конечных точек для предоставления возможности потоку передачи запроса реестра WSRR для поиска конечной точки службы или набора конечных точек служб. Если модуль SCA извлекает набор конечных точек, то он должен использовать другой примитив передачи для выбора предпочитаемой конечной точки.

Управление стратегией передачи запросов служб:

С помощью стратегий передачи можно управлять потоками передачи между клиентами служб и поставщиками служб.

Для управления потоками передачи можно использовать стратегии передачи, хранящиеся в IBM WebSphere Service Registry and Repository (WSRR). Реализация управления стратегией служб в WSRR основана на Web Services Policy Framework (WS-Policy).

Для управления запросами служб с помощью стратегий передачи в реестре WSRR должны содержаться необходимые модули SCA и документы стратегии передачи.

Как прикрепить стратегию передачи к запросу службы

При разработке модуля SCA, который будет использовать стратегию передачи, в поток передачи необходимо включить примитив передачи Policy Resolution. Во время выполнения примитив передачи Policy Resolution получает информацию о стратегии передачи из реестра. Следовательно, для управления стратегией передачи запросов служб модуль SCA должен содержать компонент потока передачи.

В реестре стратегии передачи можно прикрепить к модулю SCA или целевой службе, применяемой модулем SCA. Прикрепленные стратегии передачи можно использовать для всех сообщений службы, обрабатываемых модулем SCA. К стратегии передачи можно прикрепить условия. Условия стратегий передачи определяют, в каком контексте следует применять ту или иную стратегию. Кроме того, с помощью классификации стратегий передачи можно задать состояние управления.

WebSphere Service Registry and Repository:

Продукт WebSphere Service Registry and Repository (WSRR) позволяет хранить, использовать и управлять информацией о конечных точках служб и стратегиях передачи. WSRR может помочь сделать служебные приложения более динамичными и адаптируемыми к изменяющимся бизнес-условиям.

Введение

Потоки передачи могут использовать WSRR как динамичный механизм поиска, предоставляющий информацию о конечных точках служб или стратегиях передачи.

Для настройки доступа к WSRR необходимо создать документы определения WSRR с помощью административной консоли. Кроме этого, можно использовать административные команды WSRR из клиента сценариев wsadmin. Определения WSRR и его свойства соединения представляют собой механизм, предназначенный для подключения к экземпляру реестра и получения конечной точки службы или стратегии передачи.

Конечные точки служб

Можно использовать WSRR для хранения информации о службах, с которыми вы уже работаете или планируете работать, либо о которых должны быть осведомлены. Эти службы могут находиться в ваших

или в других системах. Например, с помощью WSRR приложение может выполнять поиск наиболее подходящей службы для удовлетворения требований к функциональности и производительности.

При разработке модуля SCA, которому требуется доступ к конечным точкам служб из WSRR, необходимо включить в поток передачи примитив передачи Поиск конечных точек. В среде выполнения примитив-посредник Поиска конечной точки получает конечные точки служб из реестра.

Стратегии передачи

WSRR также можно использовать для хранения информации и стратегиях передачи. Стратегии передачи могут помочь управлять запросами служб путем динамического переопределения свойств модуля. Если WSRR содержит стратегии передачи, прикрепленные к некоторому объекту, представляющему ваш модуль SCA или вашу целевую службу, то стратегии передачи могут переопределять свойства модуля. Если требуется применять стратегии передачи в разных контекстах, то можно создать условия стратегий передачи.

Примечание: Стратегии передачи отвечают за управление потоками отображения и не имеют отношения к защите.

При разработке модуля SCA, который будет использовать стратегию передачи, необходимо включить в поток передачи примитив передачи Обработка стратегий. В среде выполнения примитив передачи Обработка стратегий получает информацию о стратегиях передачи из реестра.

WebSphere eXtreme Scale:

С помощью продукта WebSphere eXtreme Scale (eXtreme Scale) можно получить систему кэширования, интегрируемую с приложением IBM Business Process Manager. С помощью eXtreme Scale с IBM Business Process Manager может улучшить значения времени ответа службы и ее надежность, а также предоставить дополнительные функции интеграции.

eXtreme Scale действует как эластичная масштабируемая сетка данных в памяти. Таблица данных динамически кэширует, делит на разделы, копирует данные приложений и бизнес-логику, а также управляет ими на нескольких серверах. С помощью eXtreme Scale можно также получить показатели QoS, такие как целостность транзакции, высокая готовность и предсказуемые значения времени ответа.

С помощью потоков передачи можно обратиться к функции кэширования eXtreme Scale, путем включения примитивов передачи WebSphere eXtreme Scale в свой поток. При разработке модуля архитектуры компонентов служб (SCA), которому требуется сохранять информацию в кэше eXtreme Scale, необходимо включить в поток передачи примитив передачи Сохранить WebSphere eXtreme Scale. Если требуется получить информацию из кэша eXtreme Scale, необходимо включить примитив передачи Получить WebSphere eXtreme Scale. Применяя сочетание этих двух примитивов передачи в потоке передачи, можно кэшировать ответ из системы базового сервера каталогов, с тем чтобы дальнейшие запросы могли получать ответ из кэша.

Для настройки доступа к eXtreme Scale необходимо создать определение WebSphere eXtreme Scale с помощью административной консоли. Вместо этого можно также использовать административные команды WebSphere eXtreme Scale из клиента сценариев wsadmin. Определение eXtreme Scale является тем механизмом, который примитивы передачи Получить и Сохранить WebSphere eXtreme Scale используют для соединения с сервером eXtreme Scale.

Клиенты службы сообщений

Клиенты службы сообщений доступны C/C++ и .NET для разрешения приложениям не Java соединения с Enterprise Service Bus.

Message Service Clients for C/C++ and .NET предоставляет API с именем XMS, который имеет тот же набор интерфейсов, что и API Java Message Service (JMS). Message Service Client for C/C++ содержит две реализации

XMS: одну для использования приложениями C и другую для использования приложениями C++. Message Service Client for .NET содержит полностью управляемую реализацию XMS, которая может использоваться любым языком, совместимым с .NET.

Message Service Clients for .NET можно получить на http://www-01.ibm.com/support/docview.wss?rs=0&q1=IA9H&uid=swg24011756&loc=en_US&cs=utf-8&cc=us&lang=en

Message Service Clients for C/C++ можно получить на http://www-01.ibm.com/support/docview.wss?rs=0&q1=ia94&uid=swg24007092&loc=en_US&cs=utf-8&cc=us&lang=en.

Также можно установить и использовать поддержку клиента Java EE на WebSphere Application Server Network Deployment, включая клиент веб-служб, клиент EJB и клиент JMS.

Глава 2. Дополнительные сведения об основных концепциях

Этот раздел может служить введением для знакомства с технологиями, применяемыми в IBM Business Process Manager.

Создание сценариев

Сценарии предназначены для работы с компонентами и продуктами из семейства средств, предназначенных для управления бизнес-процессами.

Поддержка версий

Жизненный цикл приложения процесса начинается с создания приложения процесса. Далее оно проходит этапы обновления, развертывания, сопряженного развертывания, отмены развертывания и архивирования. *Поддержка версий* - это способ управления жизненным циклом приложения процесса, в котором уникальным образом идентифицируются отдельные версии приложения процесса.

Механизм работы поддержки версий в IBM Business Process Manager зависит от того, что именно развертывается: приложение процесса, развертываемое из хранилища в IBM Process Center, или приложение J2EE, развертываемое непосредственно из IBM Integration Designer.

По умолчанию поддержка версий включена для приложений процессов и комплектов инструментов, развертываемых в среде выполнения из Process Center. Для приложений J2EE можно включить поддержку версией в IBM Integration Designer.

Кроме того, можно создавать версии неавтоматизированных задач или конечных автоматов, чтобы в среде выполнения могли сосуществовать несколько версий этих сущностей.

Поддержка версий приложений процессов

Поддержка версий обеспечивает идентификацию моментальный копий средой выполнения в жизненном цикле приложения процесса и возможность одновременного выполнения нескольких моментальных копий на сервере процессов.

Для того чтобы понять принципы поддержки версий приложений процессов, важно помнить, что приложение процесса является контейнером, который хранит различные артефакты, используемые в приложении процесса (например, модели процессов или BPD, ссылки на комплекты инструментов, службыили дорожки). Поддержка версий осуществляется на уровне этого контейнера, а не на уровне отдельных артефактов. Для приложений процессов это означает, что изменение версии происходит во время создания моментальной копии.

Можно сравнить моментальные копии и определить различия между версиями. Например, если один разработчик исправил неполадку со службой и сделал моментальную копию ее, содержащую приложение процесса или набор инструментов в этой точке, а затем другой разработчик внес несколько дополнительных изменений в ту же службу и сделал новую моментальную копию, то руководитель проекта может сравнить две моментальные копии и узнать, какие изменения были сделаны и кем. Если руководитель проекта решит, что дополнительные изменения в службе нецелесообразны, то у него будет возможность вернуть моментальную копию с первоначальным исправлением.

Можно выполнять различные версии (моментальные копии) приложения процесса параллельно на сервер. При установки новой моментальной копии или удалите первоначальную, или оставьте ее выполняющейся.

Контекст версии

Каждая моментальная копия содержит уникальные метаданные для идентификации версии (называемые контекстом версии). Можно присвоить этот идентификатор, но IBM рекомендует использовать трехразрядную цифровую систему обозначения версий в формате <основная>.<дополнительная>.<служебная>. Более подробное описание этой схемы обозначения версий содержится в разделах о соглашении об именах.

IBM Business Process Manager назначает глобальное пространство имен для каждого приложения процесса. Конкретно глобальным пространством имен является либо концевая точка приложения процесса, либо конкретная моментальная копия приложения процесса. Имя версии, используемое сервером, не должно быть длиннее семи символов, поэтому назначенным именем является акроним, состоящий из символов имени, присвоенного моментальной копии. Акронимы моментальных копий идентичны именам их моментальных копий, если имя моментальной копии соответствует рекомендованному стилю IBM VRM и не длиннее семи символов. Например, имя моментальной копии 1.0.0 имеет акроним 1.0.0, а имя моментальной копии 10.3.0 – акроним 10.3.0. Акроним моментальной копии должен быть уникальным в пределах контекста приложения процесса в области сервера Process Center. Поэтому нельзя изменять акроним моментальной копии.

Замечания о поддержке версий приложений процессов в нескольких кластерах

Одну и ту же версию приложения процесса можно развернуть в нескольких кластерах в пределах одной ячейки. Для различения этих нескольких развертываний одной версии приложения процессов создайте моментальную копию для каждого развертывания и включите уникальный для ячейки ИД в имя моментальной копии, (например, v1.0_cell1_1 и v1.0_cell1_2). Каждая моментальная копия является новой версией приложения процесса (только с точки зрения управления жизненным циклом), но содержимое и функции одни и те же.

При развертывании приложения процесса в кластере осуществляется автоматическая синхронизация узлов.

Замечания о поддержке версий для комплектов инструментов Process Designer

Помните, что моментальные копии приложения процесса обычно создаются перед тестированием или установкой. Однако моментальные копии комплекта инструментов делаются, когда он готов к использованию приложениями процессов. Впоследствии если потребуется обновить набор инструментов, необходимо создать другую моментальную копию "концевой точки", после чего владельцы приложений процессов могут решить, переходить ли им на новую моментальную копию.

Поддержка версий для модулей и библиотек

Если модуль или библиотека входит в приложение процесса или комплект инструментов, то он следует соответствующему жизненному циклу (версии, моментальные копии, дорожки и т. д.). Имена модулей и библиотек должны быть уникальными в области действия приложения процесса или комплекта инструментов.

В этом разделе описана поддержка версий модулей и библиотек, используемых с приложениями процессов. Однако если модули развертываются на Process Server прямо из IBM Integration Designer, то можно присваивать номера версий модулям в ходе развертывания, как описано в разделе “Создание модулей и библиотек с поддержкой версий”.

Связанные с IBM Process Center модуль или библиотека должны иметь доступ к своим зависимым библиотекам в том же приложении процесса или в зависимом комплекте инструментов.

В следующей таблице перечислены варианты выбора, доступные в редакторе зависимостей в IBM Integration Designer, когда библиотека связана с приложением процесса или комплектом инструментов:

Таблица 36. Зависимости для модулей, приложений процессов или комплектов инструментов и глобальные библиотеки

Область библиотеки	Описание	Может зависеть от . . .
Модуль	Для каждого модуля, использующего библиотеку, на сервере существует копия библиотеки.	Библиотека с областью модуля может зависеть от всех типов библиотек.
Приложение процесса или комплект инструментов	Библиотека, совместно используемая всеми модулями в области приложения процесса или комплекта инструментов. Этот параметр имеет силу, если развертывание осуществляется посредством IBM Process Center. Если развертывание выполняется без IBM Process Center, то библиотека копируется в каждый модуль. **Примечание:** Для библиотек, созданных в IBM Integration Designer версии 8, по умолчанию применяется общий доступ на уровне **Приложение процесса или комплект инструментов**.	Библиотека этого типа может зависеть только от глобальных библиотек.
Глобальная	Библиотека, совместно используемая всеми запущенными модулями.	Глобальная библиотека может зависеть только от других глобальных библиотек. **Примечание:** Для развертывания глобальной библиотеки необходимо настроить общую библиотеку WebSphere. Дополнительная информация приведена в разделе “Модули и зависимости библиотек”.

Модули и библиотеки, связанные с инструментариями и приложениями процессов

Вам не обязательно поддерживать версии модулей и библиотек, связанных с инструментариями и приложениями процессов.

Модули и библиотеки, связанные с инструментарием или приложением процесса, не требуют поддержки версий. Фактически невозможно создать версию модуля или библиотеки, связанной с инструментарием или приложением процесса в редакторе зависимостей. Модули и библиотеки, связанные с инструментарием или приложением процесса, используют моментальные копии (функцию Process Center) для достижения того же результата, который обеспечивается поддержкой версий.

Библиотеки, связанные с инструментарием или приложением процесса, не имеют требуемого номера версии в разделе Библиотеки редактора Зависимостей, потому что версии не нужны.

Соглашения о присвоении имен

Соглашение о присвоении имен позволяет различить версии приложения процесса по мере того как оно проходит этапы обновления, развертывания, сопряженного развертывания, отмены развертывания и архивирования.

В этом разделе описан соглашения, применяемые для уникальной идентификации версий приложения процесса.

Контекст версии - это аббревиатура, уникальным образом описывающая приложение процесса или комплект инструментов. Для каждого типа аббревиатуры применяется соглашение об именах. Длина аббревиатуры не может превышать семь символов из набора [A-Z0-9_], за исключением аббревиатуры моментальной копии, которая также может содержать точку.

- Аббревиатура приложения процесса создается вместе с приложением процесса. Ее длина не должна превышать семь символов.
- Аббревиатура моментальной копии создается автоматически вместе моментальной копией. Ее длина не должна превышать семь символов.

  Если имя моментальной копии также является допустимой аббревиатурой моментальной копии, то аббревиатура и имя моментальной копии будут совпадать.

  Примечание: При использовании функций маршрутизации, поддерживающих компоненты потока передачи с учетом версии, присваивайте имя моментальной копии согласно схеме <*версия*>.<*выпуск*>.<*модификация*>, например, **1.0.0**. Поскольку аббревиатура моментальной копии ограничена семью символами, то можно использовать не более пяти цифр (плюс две точки). Поэтому увеличивать значение в полях следует с осторожностью, так как все символы после первых семи усекаются.

  Например, имя моментальной копии **11.22.33** усекается в аббревиатуре моментальной копии до **11.22.3**.
- Аббревиатура дорожки автоматически составляется из первых букв каждого слова в имени дорожки. Например, для новой дорожки с именем **My New Track** будет использоваться аббревиатура **MNT**.

  Имя и аббревиатура по умолчанию для дорожки - **Main**. При развертывании на сервере IBM Process Center в контексте версии аббревиатура дорожки используется в том случае, если она отличается от **Main**.

Определение бизнес-процесса в приложении процесса обычно задается аббревиатурой имени приложения процесса, аббревиатурой моментальной копии и именем определения бизнес-процесса. При возможности используйте уникальные имена в определениях бизнес-процесса. Повторяющиеся имена могут привести в следующим ошибкам:

- Для экспорта определений бизнес-процесса в виде веб-служб может потребоваться посредник.
- Определение бизнес-процесса, созданное в IBM Process Designer, может быть недоступным в процессе BPEL, созданном в IBM Integration Designer.

Контекст версии может меняться в зависимости от способа развертывания приложения процесса.

Соглашения об именах для развертывания на сервере Process Center

На IBM Process Center можно развернуть моментальную копию приложения процесса, а также моментальную копию комплекта инструментов. Кроме того, можно развернуть головную версию приложения процесса или комплекта инструментов. Термин *головная версия* означает текущую рабочую версию приложения процесса или комплекта инструментов. Контекст версии может меняться в зависимости от типа развертывания.

Для приложений процессов головная версия приложения процесса или какая-либо его моментальная копия позволяет уникальным образом идентифицировать версию.

Комплекты инструментов могут быть развернуты с несколькими приложениями процессов, но жизненный цикл комплекта инструментов связан с жизненным циклом приложения процесса. Каждое приложение процесса имеет собственную копию зависимых комплектов инструментов, развернутых на сервере. Совместно использование развернутого комплекта инструментов несколькими приложениями процессов невозможно.

Если с приложением процесса связана дорожка, имя которой отличается от применяемого по умолчанию (**Main**), то аббревиатура дорожки также составляет часть контекста версии.

Дополнительная информация приведена в разделе “Примеры” на стр. 34 ниже в этой статье.

Моментальные копии приложения процесса

Для развертывания моментальной копии приложения процесса контекст версии включает в себя следующие элементы:

- Аббревиатура приложения процесса
- Аббревиатура дорожки приложения процесса (для дорожек с именем, отличным от **Main**)
- Аббревиатура моментальной копии приложения процесса

Автономные комплекты инструментов

Для развертывания моментальной копии комплекта инструментов контекст версии включает в себя следующие элементы:

- Аббревиатура комплекта инструментов
- Аббревиатура дорожки комплекта инструментов (для дорожек с именем, отличным от **Main**)
- Аббревиатура моментальной копии инструментов

Головные версии

Головные версии приложений процессов используются в итерационном тестировании в Process Designer. Они могут быть развернуты только на серверах Process Center.

Для развертывания головной версии приложения процесса контекст версии включает в себя следующие элементы:

- Аббревиатура приложения процесса
- Аббревиатура дорожки приложения процесса (для дорожек с именем, отличным от **Main**)
- "Головная версия"

Головные версии комплектов инструментов также используются в итерационном тестировании в Process Designer. Они не могут быть развернуты на рабочем сервере.

Для развертывания моментальной копии головной версии контекст версии включает в себя следующие элементы:

- Аббревиатура комплекта инструментов
- Аббревиатура дорожки комплекта инструментов (для дорожек с именем, отличным от **Main**)
- "Головная версия"

Примеры

Ресурсы должны иметь уникальные имена и определяться во внешней области с помощью контекста версии.

- В следующей таблице показаны примеры уникальных имен. В этом примере головная версия приложения процесса использует имя дорожки по умолчанию (**Main**):

Таблица 37. Головная версия приложения процесса с именем дорожки по умолчанию

Тип имени	**Пример**
Имя приложения процесса	**Process Application 1**
Аббревиатура приложения процесса	**PA1**
Дорожка приложения процесса	**Main**
Аббревиатура дорожки приложения процесса	"" (если дорожка - **Main**)
Моментальная копия приложения процесса	
Аббревиатура моментальной копии приложения процесса	**Tip**

Все модули SCA, связанные с этой головной версией приложения процесса, включают контекст версии, как показано в следующей таблице:

Таблица 38. Модули SCA и файлы EAR с учетом версий

Название модуля SCA	Имя с учетом версии	Имя EAR/приложения с учетом версии
M1	PA1-Tip-M1	PA1-Tip-M1.ear
M2	PA1-Tip-M2	PA1-Tip-M2.ear

- В следующей таблице показан пример головной версии приложения процесса, где имя дорожки отличается от значения по умолчанию:

Таблица 39. Головная версия приложения процесса с именем дорожки, отличающимся от принятого по умолчанию

Тип имени	Пример
Имя приложения процесса	**Process Application 1**
Аббревиатура приложения процесса	**PA1**
Дорожка приложения процесса	**Track1**
Аббревиатура дорожки приложения процесса	**T1**
Моментальная копия приложения процесса	
Аббревиатура моментальной копии приложения процесса	**Tip**

Все модули SCA, связанные с этой головной версией приложения процесса, включают контекст версии, как показано в следующей таблице:

Таблица 40. Модули SCA и файлы EAR с учетом версий

Название модуля SCA	Имя с учетом версии	Имя EAR/приложения с учетом версии
M1	PA1-T1-Tip-M1	PA1-T1-Tip-M1.ear
M2	PA1-T1-Tip-M2	PA1-T1-Tip-M2.ear

Аналогичные соглашения об именах применяются к расширенным головным версиям комплектов и развертываниям моментальных копий. Они применимы также к дополнительным моментальным копиям, установленным в Process Server.

- В следующей таблице показаны примеры уникальных имен. В этом примере моментальная копия приложения процесса использует имя дорожки по умолчанию (**Main**):

Таблица 41. Моментальная копия приложения процесса с именем дорожки по умолчанию

Тип имени	Пример
Имя приложения процесса	**Process Application 1**
Аббревиатура приложения процесса	**PA1**
Дорожка приложения процесса	**Main**
Аббревиатура дорожки приложения процесса	"" (если дорожка - **Main**)
Моментальная копия приложения процесса	**Process Shapshot V1**
Аббревиатура моментальной копии приложения процесса	**PSV1**

Все модули SCA, связанные с этой моментальной копией приложения процесса, включают контекст версии, как показано в следующей таблице:

Таблица 42. Модули SCA и файлы EAR с учетом версий

Название модуля SCA	Имя с учетом версии	Имя EAR/приложения с учетом версии
M1	PA1-PSV1-M1	PA1-PSV1-M1.ear
M2	PA1-PSV1-M2	PA1-PSV1-M2.ear

- В следующей таблице показан пример моментальной копии приложения процесса, где имя дорожки отличается от значения по умолчанию:

Таблица 43. Моментальная копия приложения процесса с именем дорожки, отличающимся от принятого по умолчанию

Тип имени	Пример
Имя приложения процесса	**Process Application 1**
Аббревиатура приложения процесса	**PA1**
Дорожка приложения процесса	**Track1**
Аббревиатура дорожки приложения процесса	**T1**
Моментальная копия приложения процесса	**Process Snapshot V1**
Аббревиатура моментальной копии приложения процесса	**PSV1**

Все модули SCA, связанные с этой моментальной копией приложения процесса, включают контекст версии, как показано в следующей таблице:

Таблица 44. Модули SCA и файлы EAR с учетом версий

Название модуля SCA	Имя с учетом версии	Имя EAR/приложения с учетом версии
M1	PA1-T1-PSV1-M1	PA1-T1-PSV1-M1.ear
M2	PA1-T1-PSV1-M2	PA1-T1-PSV1-M2.ear

Соглашения об именах для развертывания на Process Server

На Process Server можно развернуть моментальную копию приложения процесса. Аббревиатура моментальной копии приложения процесса уникальным образом идентифицирует версию.

Для развертывания моментальной копии приложения процесса контекст версии включает в себя следующие элементы:

- Аббревиатура приложения процесса
- Аббревиатура моментальной копии приложения процесса

Ресурсы должны иметь уникальные имена и определяться во внешней области с помощью контекста версии. В следующей таблице показаны примеры уникальных имен:

Таблица 45. Пример имен и аббревиатур

Тип имени	Пример
Имя приложения процесса	**Process Application 1**
Аббревиатура приложения процесса	**PA1**
Моментальная копия приложения процесса	**1.0.0**
Аббревиатура моментальной копии приложения процесса	**1.0.0**

Контекст версии является часть идентификации ресурса, такого как модуль или библиотека.

В следующей таблице показаны примеры контекста версии для двух модулей и связанных файлов EAR:

Таблица 46. Модули SCA и файлы EAR с учетом версий

Название модуля SCA	Имя с учетом версии	Имя EAR/приложения с учетом версии
M1	PA1-1.0.0-M1	PA1-1.0.0-M1.ear
M2	PA1-1.0.0-M2	PA1-1.0.0-M2.ear

В следующей таблице показаны примеры контекста версии для двух библиотек в области приложения процесса и связанных файлов JAR:

Таблица 47. Библиотеки в области приложения процесса и файлы JAR с учетом версии

Имя библиотеки SCA с областью действия приложения процесса	Имя с учетом версии	Имя JAR с учетом версии
Lib1	PA1-1.0.0-Lib1	PA1-1.0.0-Lib1.jar
Lib2	PA1-1.0.0-Lib2	PA1-1.0.0-Lib2.jar

Связывание с учетом версий

Приложения процессов могут содержать модули SCA, включающие связывание импорта и экспорта. При сопряженном развертывании приложений связывание для каждой версии приложения должно быть уникальным. В некоторых случаях это достигается при развертывании приложений, поскольку часть описаний связывания обновляется автоматически. В других случаях уникальность связывания требуется обеспечить вручную, обновив его после развертывания.

Связывание *с учетом версии* применяется для заданной версии приложения процесса. Это гарантирует уникальность связывания для приложений процессов. В следующих разделах перечислены описания связывания, которые обновляются автоматически для обеспечения учета версий, а также указаны действия, которые необходимы в среде выполнения для описаний связывания, не учитывающих версии. Дополнительная информация об операциях при создании модулей приведена в разделе “Замечания по использованию связывания”.

SCA

Если связывание импорта и экспорта определено в одной и той же области приложения процесса, то целевой объект связывания SCA переименовывается автоматически с учетом версий во время развертывания.

Если эти связывания не определены в одной и той же области приложения процесса, то в протокол заносится сообщение. Для изменения целевого адреса конечной точки необходимо изменить связывание импорта после развертывания. Это можно сделать из административной консоли.

Веб-служба (JAX-WS или JAX-RPC)

Целевой адрес конечной точки связывания веб-службы автоматически переименовывается с учетом версий, если выполнены все следующие условия:

- В адресе используется соглашение об именах по умолчанию:

  http://*IP-адрес*:*порт*/*имя-модуля*Web/sca/*имя-экспорта*
- Адрес конечной точки имеет формат SOAP/HTTP.
- Связывание импорта и экспорта определено в одной и той же области приложения процесса.

Если эти условия не выполнены, в протокол заносится сообщение. Дальнейшие действия зависят от способа развертывания приложения процесса:

- В случае сопряженного развертывания приложения процесса необходимо вручную переименовать URL конечной точки SOAP/HTTP или URL целевой очереди SOAP/JMS, чтобы он был уникальным для разных версий приложения процесса. Это можно сделать из административной консоли.

- Если развертывается только одна версия приложения процесса, то это сообщение можно проигнорировать

Для сопряженного развертывания моментальной копии связывания веб-службы SOAP/JMS дальнейшие действия зависят от способа развертывания приложения процесса:

- Если импорт и целевой объект экспорта относятся к одному и тому же приложению процесса, то перед публикацией приложения процесса в Process Center и созданием моментальной копии выполните следующее:
  1. Измените URL конечной точки экспорта. Целевой объект и фабрика соединений должны быть уникальными.
  2. Измените URL конечной точки импорта, чтобы он совпадал с указанным для экспорта на предыдущем шаге.
- Если импорт и целевой объект экспорта относятся к разным приложениям процессов, то выполните следующее:
  1. Измените URL конечной точки экспорта. Целевой объект и фабрика соединений должны быть уникальными.
  2. Опубликуйте приложение процесса в Process Center.
  3. Создайте моментальную копию.
  4. Разверните приложение процесса на Process Server.
  5. В административной консоли WebSphere измените URL соответствующей конечной точки импорта, чтобы он совпадал с указанным для экспорта.

HTTP

URL конечной точки связывания HTTP автоматически переименовывается с учетом версий, если выполнены все следующие условия:

- В адресе используется соглашение об именах по умолчанию:

  http(s)://*IP-адрес*:*порт*/*имя-модуля*Web/*путь-контекста-в-экспорте*
- Связывание импорта и экспорта определено в одной и той же области приложения процесса.

Если эти условия не выполнены, в протокол заносится сообщение. Дальнейшие действия зависят от способа развертывания приложения процесса:

- В случае сопряженного развертывания приложения процесса необходимо вручную переименовать URL конечной точки, чтобы он был уникальным для разных версий приложения процесса. Это можно сделать из административной консоли.
- Если развертывается только одна версия приложения процесса, то это сообщение можно проигнорировать

JMS и базовый JMS

Системные и базовые связывания JMS автоматически учитывают версии.

Примечание: Для пользовательских и базовых связываний JMS во время развертывания автоматическое переименование не применяется, чтобы связывание могло учитывать версии. Если связывание определено пользователем, то необходимо переименовать следующие атрибуты для обеспечения уникальности различных версий приложения процесса:

- Конфигурация конечной точки
- Целевая очередь приема
- Имя порта получателя запросов (если он определен)

При изменении конечной точки модуля настройте соответствующий целевой адрес отправки.

MQ/JMS и MQ

Для связываний MQ/JMS или MQ во время развертывания автоматическое переименование не применяется, чтобы связывание могло учитывать версии.

Необходимо переименовать следующие атрибуты для обеспечения уникальности различных версий приложения процесса:

- Конфигурация конечной точки
- Целевая очередь приема

При изменении конечной точки модуля настройте соответствующий целевой адрес отправки.

EJB

Для связывания EJB во время развертывания автоматическое переименование не применяется, чтобы связывание могло учитывать версии.

Необходимо переименовать атрибуты имен JNDI для обеспечения уникальности различных версий приложения процесса.

Обратите внимание, что эти новые имена JNDI должны также обновляться в клиентских приложениях.

EIS

Если имя по умолчанию адаптера ресурса (***имя-модуля*приложение:описание-адаптера**) не было изменено во время развертывания, то он переименовывается автоматически с учетом версий.

Если имя по умолчанию адаптера ресурса было изменено, то необходимо использовать уникальные имена адаптеров ресурсов для различных версий приложения.

Если имена адаптеров ресурса не являются уникальными, то во время развертывания в протокол заносится предупреждение. После развертывания можно вручную изменить имена адаптеров ресурсов из административной консоли.

Динамический вызов с учетом версии

В компонентах потока передачи можно настроить динамическое определение маршрутов сообщений, отправляемых в конечные точки. При создании модуля посредника можно настроить поиск маршрута к конечной точке с учетом версии.

Если для моментальной копии используется стиль IBM_VRM (<*версия*>.<*выпуск*>.<*модификация*), то можно экспортировать файл EAR приложения процесса в WebSphere Service Registry and Repository (WSRR). Тогда при создании модуля посредника можно настроить поиск маршрута к конечной точке с учетом версии. Например, в поле **Стратегия соответствия** можно указать **Найти конечную точку с последней совместимой версией служб SCA на основе модулей** и в поле **Тип связывания** выбрать **SCA**.

В этом случае будущие версии приложения процесса, развернутые на сервере и опубликованные в WSRR, будут использовать последние совместимые версии службы конечной точки, которые динамически будет находить модуль посредника.

Обратите внимание, что вместо этого можно настроить целевой объект в SMOHeader, и это значение будет передаваться в сообщении-запросе.

Развертывание приложений процессов с модулями и проектами Java

Приложения процессов могут содержать пользовательские модули Java EE и проекты Java. При сопряженном развертывании приложений пользовательский модуль Java EE для каждой версии приложения должен быть уникальным.

Обратите внимание, что пользовательские модули Java EE и проекты Java развертываются на сервере, если они развернуты с модулем SCA, в котором объявлена зависимость от них. Если параметр **Развернуть с модулем** (значение по умолчанию) не выбран при объявлении зависимости, то необходимо развернуть модуль или проект вручную.

Развертывание приложений процессов с бизнес-правилами и селекторами

При развертывании нескольких версий приложения процесса с бизнес-правилами и селекторами необходимо учитывать способ использования соответствующих мета-данных версиями.

Динамические мета-данные для бизнес-правила или компонента селектора определяются во время выполнения на основе имени компонента, целевого пространства имен компонента и типа компонента. Если в одной среде выполнения развернуто две или более версии приложения процесса, содержащего бизнес-правило или селектор, то эти версии будут совместно использовать одну и ту же логику правила (бизнес-правило) или мета-данные маршрутизации (селектор).

Для того чтобы все версии приложения процесса, содержащего бизнес-правило или селектор, применяли собственные динамические мета-данные (логика правила или маршрутизация), сделайте целевое пространство имен уникальным для каждой версии приложения процесса.

Объекты конфигурации

С помощью команд инструмента администрирования из командной строки WebSphere (wsadmin) AdminConfig можно обращаться к свойствам базы данных и к свойствам защиты в IBM Business Process Manager и изменять их.

Термин *объект конфигурации* обозначает объект, к которому можно обращаться с помощью команд wsadmin AdminConfig. Дополнительная информация приведена в разделе Свойства конфигурации защиты.

Архитектура развертывания

Архитектура развертывания IBM Business Process Manager состоит из программных процессов, именуемых серверами, топологических элементов, именуемых узлами и ячейками, а также хранилища конфигурации, применяемого для хранения информации о конфигурации.

Ячейки

В IBM Business Process Manager *ячейками* называются логические группы, включающие один или несколько узлов в распределенной сети.

Ячейка представляет собой концепцию конфигурации, т.е. способ, который используют администраторы для логической связи узлов друг с другом. Администраторы определяют узлы ячейки в соответствии с особыми критериями, имеющими смысл в среде организации.

Административные данные конфигурации хранятся в файлах XML. Ячейка хранит основные файлы конфигурации каждого сервера из каждого узла ячейки. У каждого узла и сервера есть собственные логические файлы конфигурации. Изменения, вносимые в логические файлы конфигурации сервера или логического узла, носят временный характер, если сервер входит в ячейку. Пока логические изменения действуют, они переопределяют конфигурацию из ячейки. Изменения основных файлов конфигурации узла и

сервера, внесенные на уровне ячейки, заменяют любые временные изменения на узле после синхронизации документов конфигурации из ячейки с узлами. Синхронизация выполняется при возникновении определенных событий, например при запуске сервера.

Серверы

Серверы обеспечивают основные функции IBM Business Process Manager. Серверы процессов расширяют или дополняют функции сервера приложений, связанные с обработкой модулей архитектуры компонентов служб (SCA). Другие серверы (администраторы развертывания и агенты узлов) используются для управления серверами процессов.

Сервер процессов может быть *автономным* или *управляемым*. Управляемый сервер может быть элементом *кластера*. Набор управляемых серверов, кластеров серверов и другого промежуточного программного обеспечения называется *средой развертывания*. В среде развертывания каждый управляемый сервер или кластер настроен для выполнения определенной функции (например, целевой хост, хост модуля приложения или сервер инфраструктуры событий общего формата). Автономный сервер настроен для выполнения всех необходимых функций.

Серверы обеспечивают среду выполнения для модулей SCA, для ресурсов, используемых этими модулями (источниками данных, спецификациями активации и целевыми объектами JMS), а также для ресурсов IBM (адресаты сообщений , контейнеры Business Process Choreographer, и серверы инфраструктуры событий общего формата).

Агент узла - административный агент, представляющий узел системы и управляющий серверами данного узла. Агенты узлов выполняют мониторинг серверов на хостах и направляют административные запросы на серверы. Агент узла создается при объединении узла с администратором развертывания.

Администратор развертывания - административный агент, предоставляющий централизованную панель управления для нескольких серверов и кластеров.

Автономный сервер определяется автономным профайлом, администратор развертывания - профайлом администратора развертывания, управляемые серверы создаются на *управляемом узле*, определяемом профайлом управляемого узла.

Автономные серверы

Автономный сервер предоставляет среду для развертывания модулей SCA в одному процессе сервера. К этому процессу сервера относятся административная консоль, целевая среда развертывания, поддержка обмена сообщениями, администратор правил бизнес-процессов и сервер инфраструктуры событий общего формата.

Автономный сервер отличается простотой установки и имеет консоль Быстрое начало работы, из которой можно запускать и останавливать сервер, открывать галерею примеров и административную консоль. При открытии галереи примеров после установки примеров IBM Business Process Manager в автономном сервере развертывается пример системы. Ресурсы, используемые в данном примере, можно найти в административной консоли.

На автономном сервере можно развернуть собственное решение, однако он не имеет производительности, масштабируемости или устойчивости, необходимой для рабочей среды. В качестве рабочей среды рекомендуется использовать среду сетевого развертывания.

Можно начать с автономного сервера, а потом включить его в среду сетевого развертывания, объединив его с ячейкой администратора развертывания, *если другие узлы не были объединены с этой ячейкой*. Невозможно объединить несколько автономных серверов в одной ячейке. Для того чтобы добавить автономный сервер, используйте административную консоль администратора развертывания или команду **addNode**. Если для объединения используется команда **addNode**, автономный сервер не должен работать.

Автономный сервер определяется профайлом автономного сервера.

Кластеры

Кластеры - группы совместно управляемых серверов, участвующих в управлении рабочей нагрузкой.

Кластер может включать узлы или отдельные серверы приложений. Как правило, узел - это физическая компьютерная система со своим IP-адресом, на которой работает один или несколько серверов приложений. Кластеры можно объединять в конфигурацию ячейки, которая логически связывает многие серверы и кластеры, имеющие разные конфигурации и приложения, учитывая стоящие перед администратором задачи и соображения целесообразности для организации.

Кластеры применяются для распределения нагрузки между серверами. Серверы, входящие в состав кластера, называются элементами кластера. При установке приложения в кластере оно устанавливается на всех элементах кластера.

Поскольку каждый элемент кластера содержит одни и те же приложения, задачи клиентов можно распределять по системам в зависимости от имеющихся у них ресурсов, присвоив им весовые коэффициенты.

Присваивание весовых коэффициентов серверам в кластере позволяет повысить производительность и упростить аварийное восстановление. Задачи назначаются тем серверам, у которых есть ресурсы для их выполнения. Если один из серверов недоступен для выполнения задачи, то она назначается другому элементу кластера. Среда с возможностью переназначения задач имеет очевидные преимущества перед использованием одного сервера приложений, который может быть перегружен при наличии слишком большого числа запросов.

Профайлы

Профайл определяет уникальную рабочую среду с отдельными файлами команд, файлами конфигурации и файлами протокола. Профайлы определяют три разных типа сред в системах IBM Business Process Manager: автономный сервер, администратор развертывания и управляемый узел.

С помощью профайлов можно включить в систему несколько рабочих сред без необходимости установки нескольких копий исполняемых файлов IBM Business Process Manager.

С помощью Утилита командной строки BPMConfig можно создать профайлы IBM BPM. В качестве альтернативных способов создания профайлов администратора развертывания или управляемого узла можно использовать утилиту командной строки **manageprofiles** или ее графический интерфейс Инструмент управления профайлами (PMT). PMT больше не поддерживается при создании автономного профайла.

Примечание: В распределенных платформах каждый профайл имеет уникальное имя. В z/OS все профайлы имеют имя “default”; переименовывать, изменять, копировать или удалять профайлы в z/OS нельзя.

Типы профайлов

В IBM Business Process Manager V8.5 доступны следующие типы профайлов IBM BPM:

Автономный профайл IBM BPM
: Автономный профайл определяет автономные серверы с функциями и возможностями для конкретных конфигураций IBM BPM Express. Для создания автономного профайла требуется шаблон профайла `BPM/BpmServer`, который поддерживается только в IBM BPM Express.

Профайл администратора развертывания IBM BPM
: Профайл администратора развертывания определяет администратор развертывания, предоставляющий один административный интерфейс для логической группы серверов на одной или нескольких рабочих станциях. Для создания профайла администратора развертывания требуется шаблон профайла `BPM/BpmDmgr`, который поддерживается только в IBM BPM Standard и IBM BPM Advanced.

Профайл управляемого узла IBM BPM
: Профайл управляемого узла определяет управляемый узел, если узел объединен с администратором

развертывания. Для создания профайла управляемого узла требуется шаблон профайла `BPM/BpmNode`, который поддерживается только в IBM BPM Standard и IBM BPM Advanced.

Указанные типы профайлов доступны для всех конфигураций IBM BPM.

Таблица 48. Доступные типы профайлов IBM BPM

	Типы профайлов		
Конфигурация IBM BPM	**Автономные**	**Администратор развертывания**	**Управляемый узел**
IBM BPMExpress	Да	Нет	Нет
IBM BPM Standard	Нет	Да	Да
IBM BPMРасширенный	Нет	Да	Да
Среда полнофункционального тестирования (UTE) для IBM Integration Designer	Да	Необязательный	Необязательный

Каталог профайла

Каждый профайл в системе находится в собственном каталоге, который содержит все файлы. Задайте каталог расположения профайла при его создании. По умолчанию это каталог `profiles` в каталоге установки IBM Business Process Manager. Например, профайл Dmgr находится в каталоге `C:\Program Files\IBM\WebSphere\ProcServer\profiles\Dmgr`.

Стандартный профайл

Первый профайл, созданный в одном экземпляре продукта IBM Business Process Manager, считается *стандартным*. Стандартный профайл является стандартным целевым объектом для команд, запущенных из каталога `bin` в каталоге установки IBM Business Process Manager. Если в системе существует всего один профайл, к нему обращается каждая команда. Если создается другой профайл, его можно сделать стандартным.

Примечание: Стандартный профайл необязательно имеет имя “default”.

Дополнение профайлов

Если профайл администратора развертывания, профайл управляемого узла или профайл автономного сервера уже создан для WebSphere Application Server Network Deployment, его можно *дополнить* для поддержки IBM Business Process Manager в дополнение к существующей функции. Для того чтобы расширить профайл, сначала необходимо установить IBM Business Process Manager. Затем используйте утилиту командной строки **`manageprofiles`** для дополнения автономного профайла либо воспользуйтесь инструментом управления профайлами или утилитой командной строки **`manageprofiles`** для дополнения профайла администратора развертывания или управляемого узла.

Важное замечание: В среде сетевого развертывания необходимо сначала дополнить профайл администратора развертывания, а затем дополнить профайлы управляемого узла.

Ограничение: Невозможно дополнить автономный профайл или профайл администратора развертывания, в котором был изменен реестр пользователей WebSphere VMM по умолчанию, например для использования LDAP.

Администраторы развертывания

Администратор развертывания представляет собой сервер, управляющий операциями для логических групп, ячеек или других серверов. Администратор развертывания является центральным элементом администрирования серверов и кластеров.

При создании среды развертывания первым создается профайл администратора развертывания. Каждая создаваемая среда развертывания имеет консоль Быстрое начало работы, с помощью которой можно запускать или останавливать администратор развертывания и запустить его административную консоль. Административная консоль администратора развертывания применяется для управления серверами и кластерами в ячейке. В частности, она применяется для настройки серверов и кластеров, добавления серверов в кластеры запуска и остановки серверов и кластеров, а также развертывания модулей SCA.

Несмотря на то, что администратор развертывания представляет собой один из типов серверов, модули в самом администраторе развертывания развертывать нельзя.

Узлы

Узел - это логическая группа управляемых серверов.

Узел, как правило, соответствует логической или физической компьютерной системе с отдельным IP-адресом хоста. Узлы не могут охватывать несколько компьютеров. Имена узлов идентичны имени хоста компьютера.

Узлы в топологии сетевого развертывания могут быть как управляемыми, так и неуправляемыми. Агент управляемого узла управляет его конфигурацией и серверами. Неуправляемые узлы агента не имеют.

Управляемые узлы

Управляемый узел - это узел, объединенный с администратором развертывания, который содержит агент узла и может включать управляемые серверы. В управляемом узле можно настраивать и запускать управляемые серверы.

Серверы, настроенные на управляемом узле, составляют ресурсы среды развертывания. Эти серверы создаются, настраиваются, запускаются, останавливаются, управляются и удаляются с помощью административной консоли администратора развертывания.

Агент управляемого узла управляет всеми серверами этого узла.

Если узел объединяется, процесс агента узла создается автоматически. Этот агент узла должен работать для управления конфигурацией профайла. Например, при выполнении следующих задач:

- Запуск и остановка процессов сервера.
- Синхронизация данных конфигурации в администраторе развертывания с копией на узле.

Однако агент узла не обязательно должен работать для того, чтобы приложения могли запускать или настраивать ресурсы узла.

Управляемый узел может содержать один или несколько серверов, которые управляются администратором развертывания. Решения можно развертывать на серверах управляемого узла, но он не содержит галереи примеров приложений. Управляемый узел определяется профайлом управляемого узла и имеет консоль Быстрое начало работы.

Неуправляемые узлы

У неуправляемого узла нет агента узла, управляющего его серверами.

У неуправляемых узлов в топологии сетевого развертывания могут быть определения серверов, например веб-серверов, но не определения серверов приложений. Неуправляемые узлы нельзя встроить в ячейку. Это означает, что на неуправляемый узел нельзя добавить агент узла. Особым типом неуправляемого узла

является автономный сервер. Администратор развертывания не поддерживает управление автономными серверами, так как информация о них отсутствует в ячейке. Автономный сервер можно встроить в ячейку. В результате этого автоматически создается агент узла. Узел становится управляемым узлом ячейки.

Агенты узлов

Агенты узла представляют собой административных агентов, которые передают административные запросы на серверы.

Агент узла - это сервер, который работает в компьютерной системе каждого хоста, включенного в конфигурацию сетевого развертывания. Этот агент применяется только для администрирования и не участвует в работе функций обслуживания приложений. На агенте узла выполняется ряд важных административных функций, в том числе службы передачи файлов, синхронизации конфигурации и мониторинга производительности.

Особенности присвоения имен профайлам, узлам, серверам, хостам и ячейкам

В этом подразделе описываются зарезервированные термины и неполадки, связанные с именами профайлов, узлов, серверов, хостов и ячеек (в соответствующих случаях). Информация из этого раздела относится к распределенным платформам.

Особенности имен профайлов

В качестве имени профайла можно использовать любое уникальное имя со следующими ограничениями. Не используйте следующие символы в именах профайлов:

- Пробелы
- Специальные символы, недопустимые в имени каталога для данной ОС, такие как *, & или ?.
- Символы косой черты (/) и обратной косой черты (\)

Двухбайтовые символы допустимы.

Windows **Особенности указания пути к каталогу:** длина пути к каталогу установки не должна превышать 60 символов. Число символов в имени каталога *profiles_directory_path\profile_name* не должно превышать 80 символов.

Примечание: При создании профайла в среде Windows рекомендуется использовать соглашение о кратких именах, поскольку в системах Windows длина пути не должна превышать 255 символов.

Особенности присвоения имен для узлов, серверов, хостов и ячеек

Зарезервированные имена: Избегайте использования зарезервированных имен в качестве значений полей. Использование зарезервированных имен может привести к непредсказуемым результатам. Зарезервированы следующие слова:

- cells
- nodes
- servers
- clusters
- applications
- deployments

Описание полей имен узлов и хостов, а также страниц имен узлов, хостов и ячеек: При создании профайлов используйте соответствующие рекомендации по именованию.

- Профайлы автономных серверов

- Профайлы администратора развертывания
- Профайлы управляемых узлов

Таблица 49. Рекомендации по именованию профайлов автономного сервера

Имя поля	Стандартное значение	Ограничения	Описание
Имя узла	Linux UNIX Windows *shortHostName* Node *NodeNumber*, где: • *shortHost Name* - краткое имя хоста. • *NodeNumber* - последовательный номер, начинающийся с 01.	Избегайте использования зарезервированных имен.	Выберите любое имя. В целях улучшения организации установки экземпляра используйте уникальное имя, если планируется создание нескольких серверов в системе.
Имя сервера	Linux UNIX Windows server1	Используйте уникальное имя сервера.	Логическое имя сервера.
Имя хоста	Linux UNIX Windows Длинная форма имени сервера имен доменов (DNS).	Используйте полное имя хоста, допускающее обращение к нему через вашу сеть.	Используйте фактическое имя DNS или IP-адрес рабочей станции, чтобы включить связь с ним. Дополнительная информация об имени хоста приведена после следующей таблицы.

Таблица 50. Рекомендации по именованию профайлов администратора развертывания

Имя поля	Стандартное значение	Ограничения	Описание
Имя узла	Linux UNIX Windows *shortHostName* Cell Manager*Node Number*, где: • *shortHost Name* - краткое имя хоста. • *NodeNumber* - последовательный номер, начинающийся с 01.	Используйте уникальное имя администратора развертывания. Избегайте использования зарезервированных имен.	Имя используется для администрирования в ячейке администратора развертывания.
Имя хоста	Linux UNIX Windows Длинная форма имени сервера имен доменов (DNS).	Используйте полное имя хоста, допускающее обращение к нему через вашу сеть. Избегайте использования зарезервированных имен.	Используйте фактическое имя DNS или IP-адрес рабочей станции, чтобы включить связь с ним. Дополнительная информация об имени хоста приведена после следующей таблицы.

Таблица 50. Рекомендации по именованию профайлов администратора развертывания (продолжение)

Имя поля	Стандартное значение	Ограничения	Описание
Имя ячейки	Linux UNIX Windows *shortHostName* Cell *CellNumber*, где: • *shortHost Name* - краткое имя хоста. • *CellNumber* - последовательный номер, начинающийся с 01.	Используйте уникальное имя ячейки администратора развертывания. Имя ячейки должно быть уникальным в любых ситуациях, при которых продукт работает на одной физической рабочей станции или в кластере из нескольких рабочих станций, таком как Sysplex. Кроме того, имя ячейки должно быть уникальным в любых ситуациях, при которых необходимо сетевое подключение между объектами: между ячейками или клиентом и каждой ячейкой. Имена ячеек также должны быть уникальными, если их пространства имен будут объединяться. В противном случае произойдет исключительная ситуация javax.naming.Name NotFoundException, при возникновении которой необходимо создать ячейки с уникальными именами.	Все объединенные узлы становятся элементами ячейки администратора развертывания, имя которой используется на странице Имена узлов, хостов и ячеек в инструменте управления профайлами.

Таблица 51. Рекомендации по именованию профайлов для управляемых узлов

Имя поля	Стандартное значение	Ограничения	Описание
Имя узла	Linux UNIX Windows *shortHostName* Node *NodeNumber*, где: • *shortHost Name* - краткое имя хоста. • *NodeNumber* - последовательный номер, начинающийся с 01.	Избегайте использования зарезервированных имен. Используйте уникальное имя в ячейке администратора развертывания.	Имя используется для администрирования в ячейке администратора развертывания, в которую добавляется профайл управляемого узла. Используйте уникальное имя в ячейке администратора развертывания.
Имя хоста	Linux UNIX Windows Длинная форма имени сервера имен доменов (DNS).	Используйте полное имя хоста, допускающее обращение к нему через вашу сеть.	Используйте фактическое имя DNS или IP-адрес рабочей станции, чтобы включить связь с ним. Дополнительная информация об имени хоста приведена после следующей таблицы.

Особенности указания имени хоста:

Имя хоста - сетевое имя физической рабочей станции, на которой установлен узел. Имя хоста должно определять физический узел сети на сервере. Если на сервере есть несколько сетевых карт, имя хоста или IP-адрес должны указывать на одну из них. Удаленные узлы используют имя хоста для подключения к нему и связи с ним.

IBM Business Process Manager соответствует протоколу IP версии 4 (IPv4) и версии 6 (IPv6). Пр вводе IP-адресов в административной консоли или в других программах можно использовать любой формат. Обратите внимание на то, что если в системе реализован протокол IPv6, то IP-адреса необходимо вводить в формате IPv6, и наоборот, если IPv6 недоступен, вводите IP-адреса в формате IPv4. Дополнительная информация о IPv6 приведена в следующем описании: IPv6.

Инструкции по определению правильного имени хоста рабочей станции:

- Выберите имя хоста для доступа к данной рабочей станции в сети.
- Не используйте общий идентификатор localhost в качестве этого значения.
- Не устанавливайте продукты IBM Business Process Manager на сервер, имя хоста которого использует символы двухбайтового набора (DBCS). Символы DBCS не поддерживаются в именах хостов.
- Не используйте нижнее подчеркивание (_) в именах серверов. Интернет-стандарты предусматривают, что имена доменов должны соответствовать требованиям к именам хостов, описанным в официальных стандартах протокола IP RFC 952 и RFC 1123. Имена доменов должны состоять только из букв (в нижнем или верхнем регистре) и цифр. Имена доменов могут также включать дефис (-), если они не находятся в конце имени. Символы нижнего подчеркивания (_) в именах хостов не поддерживаются. Если продукт IBM Business Process Manager установлен на сервере, имя которого содержит нижнее подчеркивание, используйте для доступа к этому серверу его IP-адрес до тех пор, пока он не будет переименован.

Если в одной системе выявлены параллельные узлы с уникальными IP-адресами, определите каждый IP-адрес в таблице сервера имен доменов (DNS). Файлы конфигурации серверов не поддерживают преобразование имени домена в адрес для нескольких IP-адресов на рабочей станции с одним сетевым адресом.

Значение, заданное для имени хоста, используется в качестве значения свойства hostName в документации конфигурации. Задайте значение имени хоста в одном из следующих форматов:

- Строка полного имени хоста серверов имен доменов (DNS), например xmachine.manhattan.ibm.com
- Строка стандартного краткого имени хоста DNS, например xmachine
- Числовой IP-адрес, например 127.1.255.3

Полное имя хоста DNS связано с такими преимуществами, как полная однозначность и гибкость. Можно изменять фактический IP-адрес хоста системы, не изменяя при этом конфигурацию сервера. Это значение имени хоста особенно полезно, если планируется частое изменение IP-адреса при использовании протокола настройки динамического хоста (DHCP) для назначения IP-адресов. Недостаток этого формата заключается в том, что он зависит от DNS. Если DNS недоступен, соединение отсутствует.

Краткое имя хоста может также распознаваться динамически. Краткое имя хоста позволяет повторное определение в файле локального хоста, поэтому система может запустить сервер даже при отключении от сети. Задайте краткое имя 127.0.0.1 (локальный циклический адрес) в файле хоста, чтобы работать в режиме отключения. Недостатком краткого имени является то, что он зависит от DNS при удаленном доступе. Если DNS недоступен, соединение отсутствует.

Числовой IP-адрес дает такое преимущество, как отсутствие необходимости преобразования имен через DNS. Удаленный узел может подключиться к узлу, которому присвоен числовой IP-адрес, даже если DNS недоступен. Недостаток этого формата заключается в том, что числовой IP-адрес является фиксированным. Для того чтобы изменить IP-адрес рабочей станции, необходимо изменить параметр свойства hostName в документации конфигурации. Таким образом, не используйте числовой IP-адрес, если используется протокол DHCP или если IP-адрес изменяется регулярно. Другим недостатком этого формата является то, что если хост отключен от сети, узел становится недоступным.

BPMN 2.0

Определения бизнес-процессов IBM Business Process Manager поддерживают производный класс Common Executable класса моделирования процессов BPMN 2.0, предназначенный для работы с исполняемыми моделями.

BPMN (Business Process Model and Notation) - это основополагающий стандарт для процессов IBM Process Designer и IBM Process Center. Диаграммы определений бизнес-процессов (BPD) соответствуют требованиям спецификации BPMN. В этом разделе рассмотрены отдельные способы применения BPMN 2.0 в IBM Business Process Manager. Подробная информация о BPMN приведена на странице спецификации BPMN: http://www.bpmn.org/.

IBM Business Process Manager поддерживает следующие типы задач BPMN 2.0:

- Нет (абстрактная задача в спецификации BPMN 2.0)
- Системная задача (служебная задача в спецификации BPMN 2.0)
- Пользовательская задача
- Сценарий
- Задача решения (задача бизнес-правила в спецификации BPMN 2.0)

Вместо задач отправки и приема BPMN можно использовать промежуточные события сообщений IBM BPM.

Нотация BPMN 2.0

Начиная с версии 7.5.1, значки задач BPMN 2.0 Process Designer на диаграммах BPD собраны на упрощенной палитре и отображаются на диаграммах процессов. Значок позволяет определить тип операции: системная задача, пользовательская задача, задача решения, сценарий или связанный процесс. В моделях, созданных с помощью предыдущих версий, при просмотре в версии 7.5.1 и выше для операций отображаются типы и значки задач BPMN 2.0.

Операции и задачи

По сравнению с предыдущими версиями Process Designer был внесен ряд изменений в терминологию. В частности, были переименованы отдельные типы операций.

- Операции службы (автоматические) теперь называются системными задачами.
- Операции службы (задачи), расположенные на несистемной полосе, теперь называются пользовательскими задачами.
- Операции службы (задачи), расположенные на системной полосе, теперь называются задачами решения, если они связаны со службой решения.
- Операции службы (задачи), расположенные на системной полосе, теперь называются системными задачами, если они связаны со службами, отличными от службы решения.
- Операции Javascript теперь называются задачами сценариев.
- Операции вложенных процессов теперь называются связанными процессами.
- Внешние операции из предыдущих версий Process Designer доступны в качестве внешних реализаций пользовательских или системных задач.

Шлюзы

Изменения нотаций по сравнению со шлюзами предыдущих версий отсутствуют. Однако внесены изменения в терминологию. Шлюз решения теперь называется *шлюзом исключающего ИЛИ*, шлюз простого разбиения или соединения теперь называется *параллельным шлюзом*, а шлюз условного разбиения или соединения теперь называется *шлюзом И*.

Кроме того, предусмотрен новый тип шлюза - *шлюз событий*. Шлюз событий представляет точку разветвления, альтернативные пути в которой выбираются с учетом возникающих событий, таких как прием сообщения, а не вычисления выражений на основе данных процесса (как в случае включающих и исключающих шлюзов).

Непрерывающие события

В BPMN 2.0 добавлена нотация непрерывающих событий. По умолчанию граничное событие прерывает связанную операцию. В ответ на возникновение события операция останавливается с передачей маркера вниз по потоку последовательности события. Непрерывающее событие не останавливает связанную операцию, которая продолжает работу в параллельном режиме. При этом создается новый маркер, который передается вниз по потоку последовательности события. Граница непрерывающего события отображается в виде пунктирной линии.

Промежуточные события, связанные с операциями, называются прерывающими промежуточными событиями, если они закрывают связанные операции. В противном случае они называются непрерывающими промежуточными событиями.

Событие запуска

Спецификация BPMN допускает отсутствие событий запуска и завершения в модели процесса. В Process Designer наличие событий запуска и завершения в модели процесса обязательно.

В Process Designer доступны различные типы событий запуска:

процессы

- нет
- сообщение
- специальные

подпроцессы

- нет

подпроцессы событий

- ошибка
- сообщение
- таймер

Тип события запуска можно изменить в свойствах события. Процесс может содержать несколько событий запуска типа Сообщение, но только одно событие запуска типа Нет.

События завершения

Доступно четыре типа событий завершения: *сообщение*, *остановка*, *ошибка* и *нет*. При необходимости тип конечного события можно изменить.

Если дочерний процесс, вызванный родительским процессом, выполняет действие события остановки, то дочерний процесс останавливается, а родительский продолжает работу.

Подпроцессы

В спецификации BPMN определено два типа подпроцессов: встроенные и многоразовые. Оба типа доступны в Process Designer. Встроенные подпроцессы в Process Designer называются просто *подпроцессами*; это новое понятие в версии 7.5.1. Многоразовые подпроцессы BPMN в Process Designer называются *связанным процессом*.

Подпроцесс существует в родительском процессе и представляет собой способ группировки связанных элементов процесса. Подпроцессы группируют несколько шагов в одну операцию. Подпроцесс доступен только родительскому процессу. Подпроцесс существует в области вызывающего процесса и обладает доступом ко всем переменным в этой среде. Для встроенного подпроцесса не предусмотрены входные и выходные параметры.

Помимо подпроцесса и связанного процесса в Process Designer применяется подпроцесс событий, который представляет собой специальный подпроцесс для обработки событий. Он не связан с другими операциями в потоке последовательности и выполняется только при возникновении конкретного события запуска.

Связанные процессы

Многоразовый подпроцесс BPMN называется *связанным процессом* в Process Designer. Он существует за пределами текущего процесса и может быть вызван текущим процессом. Он является многоразовым, поскольку его могут вызывать определения других процессов. Связанный процесс задает входные и выходные параметры и не обладает доступом к области или среде вызывающего процесса. Связанный процесс аналогичен вложенным процессам, которые поддерживались в предыдущих версиях; поведение операции не изменилось. Операции вложенных процессов теперь называются связанными процессами. Связанный процесс выглядит аналогично подпроцессу с толстой границей. Кроме того, он выделен в окне Inspector.

Циклы

В спецификации BPMN описана концепция повторяемой операции. Может повторяться отдельная операция или подпроцесс, содержащий несколько операций. При развертывании повторяемой операции отображаются вложенные операции, которые будут выполнены несколько раз. Условие всегда проверяется в начале каждого повторения цикла. Возможность проверки условия в конце повторения не предусмотрена.

В IBM Business Process Manager доступен *цикл с несколькими экземплярами*, который выполняется ограниченное число раз - вложенные операции выполняются последовательно или параллельно.

Импорт процессов, отличных от BPMN

Можно импортировать модели, созданные в IBM WebSphere Business Modeler, для применения в Process Designer. За информацией об импорте BPMN 2.0 обратитесь к разделу Связывание элементов IBM WebSphere Business Modeler с конструктивными элементами IBM Business Process Manager. При необходимости можно импортировать модели BPMN 2.0, созданные с помощью IBM WebSphere Business Compass, Rational Software Architect или других сред моделирования.

Определения бизнес-процессов (BPD)

Для моделирования процесса в IBM Process Designer необходимо создать определение бизнес-процесса (BPD). Определение бизнес-процесса может быть основано на импортированной модели BPMN.

Определение бизнес-процесса представляет собой многократно используемую модель процесса, описывающую общую часть всех экземпляров этой модели процесса во время выполнения. Определение BPD должно содержать событие запуска, событие завершения, по крайней мере одну полосу и одну или несколько операций. Дополнительная информация об ограничениях на использование символов в BPD приведена в разделе "Соглашения о присвоении имен в IBM Process Designer", указанном в ссылках на связанную информацию.

Определение бизнес-процесса (BPD) должно содержать полосу для каждой системы или группы пользователей, участвующих в процессе. Полоса может являться полосой участника или полосой системы. Однако при необходимости можно создать такое определение BPD, в котором операции группы и системы объединены в одной полосе. За информацией о создании BPD обратитесь к разделу "Создание определения бизнес-процесса (BPD)", указанном в ссылках на связанную информацию.

Любого пользователя или группу можно назначить ответственным за операции в полосе участника. По умолчанию любая созданная полоса назначается группе Все пользователи. BPD можно выполнить и протестировать в компоненте Inspector, используя эту группу по умолчанию. Группа Все пользователи содержит всех пользователей, которые входят в состав группы защиты **tw_allusers** - специальной группы защиты, в которую автоматически добавляются все пользователи системы.

Полоса системы содержит операции, выполняемые определенной системой IBM Process Center. У любой операции должна быть реализация, определяющая операцию и свойства задачи. Во время реализации разработчик создает службы или пишет сценарии JavaScript, необходимые для выполнения операций в полосе системы. Обратитесь к разделу "Описание типов служб", указанному в ссылках на связанную информацию, за информацией о службах.

При создании любого BPD необходимо объявить переменные, которые будут содержать бизнес-данные, передаваемые из одной операции в другую внутри процесса. Для получения информации о реализации переменных обратитесь к разделу "Управление переменными и создание связей", указанному в ссылках на связанную информацию.

При необходимости в BPD можно добавить события. В программе IBM BPM события могут возникать при наступлении определенной даты, при возникновении исключительной ситуации или при получении сообщения. Триггер определяет тип события, выбранного для реализации. За дополнительной информацией о доступных типах событий и их триггерах обратитесь к разделу "Моделирование событий".

Привязки

В основе архитектуры на основе служб лежит понятие *службы* - функциональной единицы, в работе которой используется взаимодействие между компьютерными устройствами. *Экспорт* определяет внешний интерфейс (или точку доступа) модуля, благодаря чему компоненты SCA модуля могут предоставлять свои службы внешним клиентам. *Импорт* определяет интерфейс доступа к службам, расположенным вне модуля, что позволяет вызывать службы внутри модуля. Вместе с объектами импорта и экспорта используются *привязки* определенных протоколов, определяющие способ передачи данных в модуль и из него.

Экспорты

Внешние клиенты могут вызывать компоненты SCA модуля интеграции с помощью различных протоколов (таких как HTTP, JMS, MQ и RMI/IIOP) с данными в различных форматах (в том числе XML, CSV, COBOL и JavaBeans). Объекты экспорта - это компоненты, которые получают запросы из внешних источников и вызывают компоненты IBM Business Process Manager, использующие модель программирования SCA.

Например, на следующем рисунке объект экспорта получает запрос по протоколу HTTP от приложения-клиента. Данные преобразуются в бизнес-объект - формат, применяемый компонентами SCA. После этого вызывается компонент, которому передается этот объект данных.

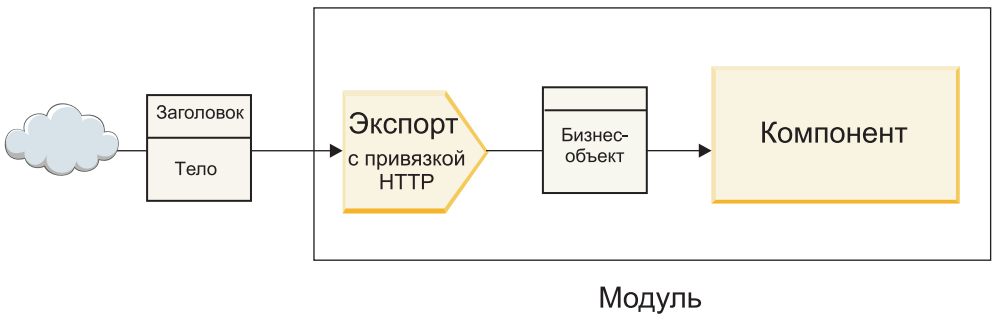


Рисунок 46. Экспорт с привязкой HTTP

Импорты

Компоненту SCA может потребоваться вызвать внешнюю службу, не поддерживающую SCA и принимающую данные в другом формате. Для вызова внешней службы в компоненте SCA используется объект импорта. Объект импорта вызывает целевую службу тем способом, который поддерживается этой службой.

Например, на следующем рисунке запрос компонента SCA отправляется во внешнюю службу через объект импорта. Бизнес-объект, то есть данные в формате, поддерживаемом в компоненте SCA, преобразуются в формат, ожидаемый службой, после чего вызывается служба.

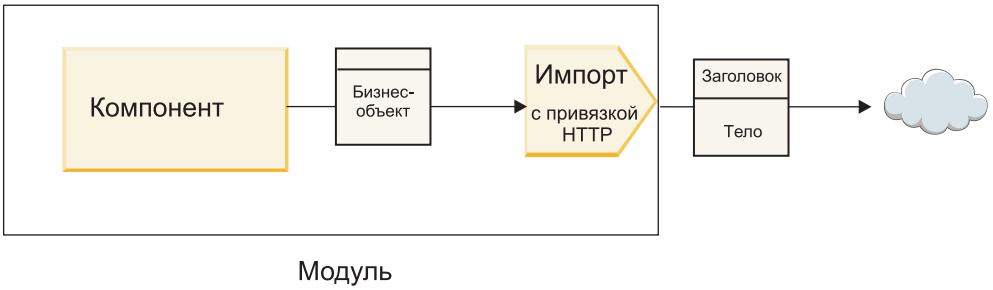


Рисунок 47. Импорт с привязкой HTTP

Список привязок

С помощью Integration Designer можно создать привязку для импорта или экспорта и настроить привязку. В следующем списке перечислены доступные типы привязок.

- SCA

  Привязка SCA, которая используется по умолчанию, позволяет службе взаимодействовать со службами из других модулей SCA. Используя импорт с привязкой SCA, можно получить доступ к службе из другого модуля SCA. Используя экспорт с привязкой SCA, можно предоставить службу для других модулей SCA.
- Веб-служба

  Привязка веб-службы позволяет получить доступ к внешней службе, используя универсальные сообщения SOAP и параметры качества обслуживания. Используя привязку веб-службы, можно добавлять прикрепления в сообщение SOAP.

  Привязка веб-службы может использовать транспортный протокол SOAP/HTTP (SOAP через HTTP) или SOAP/JMS (SOAP через JMS). Вне зависимости от типа транспортного протокола (HTTP или JMS), используемого для передачи сообщений SOAP, привязка веб-службы обеспечивает синхронное взаимодействие по типу запрос-ответ.
- HTTP

  Привязка HTTP позволяет получить доступ к внешней службе по протоколу HTTP, если используются сообщения, отличные от SOAP или необходим прямой доступ по HTTP.. Эта привязка используется при работе с веб-службами, основанными на модели HTTP (то есть службами, поддерживающими стандартные операции интерфейса HTTP, такие как GET, PUT, DELETE и т. п.).
- Enterprise JavaBean (EJB)

  Привязка EJB позволяет компонентам SCA взаимодействовать со службами, предоставляемыми бизнес-логикой Java EE, которая выполняется на сервере Java EE.
- EIS

  Привязка EIS (enterprise information system - информационная система предприятия), используемая совместно с адаптером ресурсов JCA, позволяет получать доступ к службам информационной системы предприятия или предоставлять свои службы для использования в EIS.
- Привязки JMS

Привязки Java Message Service (JMS), базового JMS и WebSphere MQ JMS (MQ JMS) используются для взаимодействия с системами доставки сообщений, если для обеспечения надежности необходимо асинхронное взаимодействие через очереди сообщений.

Объект экспорта с одной из привязок JMS отслеживает поступление сообщений в очередь и асинхронно отправляет ответ (если он есть) в очередь ответов. Объект импорта с одной из привязок JMS составляет и отправляет сообщение в очередь JMS и отслеживает поступление ответа в очередь.

- JMS

  Привязка JMS позволяет получить доступ к встроенному в WebSphere провайдеру JMS.

- Базовый JMS

  Привязка базового JMS позволяет получить доступ к системе доставки сообщений другого поставщика.

- MQ JMS

  Привязка MQ JMS позволяет получить доступ к подмножеству функций JMS из системы доставки сообщений WebSphere MQ. Эту привязку можно использовать в том случае, если для работы приложения достаточно этого подмножества функций JMS.

- MQ

  Привязка WebSphere MQ позволяет взаимодействовать с внутренними приложениями MQ, встраивая их в среду архитектуры на основе служб и предоставляя доступ к информации заголовка, специфичной для MQ. Эту привязку следует использовать в том случае, если планируется применять внутренние функции MQ.

Обзор привязок экспорта и импорта

Экспорт делает службы из модуля интеграции доступными для внешних клиентов, а импорт позволяет компонентам SCA в модуле интеграции вызывать внешние службы. Привязка, связанная с экспортом или импортом, определяет взаимосвязь между сообщениями протокола и бизнес-объектами. Кроме того, она определяет способ выбора операций и обработки ошибок.

Передача информации при экспорте

Объект экспорта получает запрос, адресованный компоненту, с которым экспорт связан проводником, через транспортный протокол, который определяется соответствующей привязкой (например, HTTP).

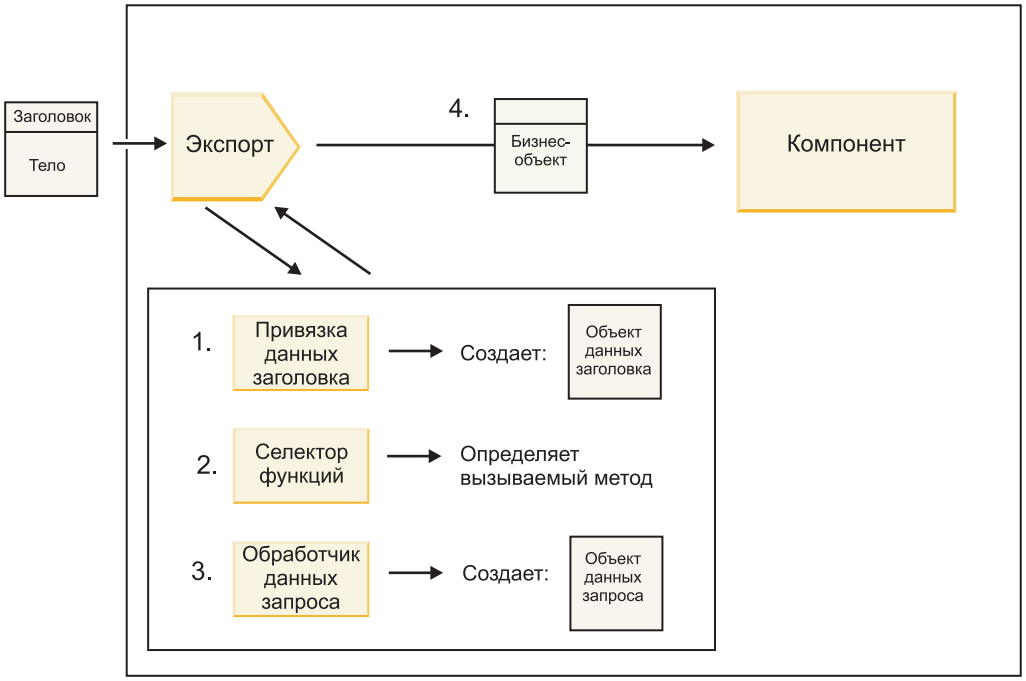


Рисунок 48. Передача запроса в компонент через объект экспорта

Когда объект экспорта получает запрос, выполняется следующая последовательность действий:

1. Для привязок WebSphere MQ: привязка данных заголовка преобразует заголовок протокола в объект данных заголовка.
2. Селектор функции определяет имя внутреннего метода, используя сообщение протокола. Имя внутреннего метода преобразуется конфигурацией экспорта в имя операции интерфейса экспорта.
3. Обработчик данных запроса или привязка данных в методе преобразует запрос в бизнес-объект запроса.
4. Объект экспорта вызывает метод компонента для бизнес-объекта запроса.
   - Привязка экспорта HTTP, привязка экспорта веб-службы и привязка экспорта EJB выполняют синхронный вызов компонента SCA.
   - Привязки экспорта JMS, базового JMS, MQ JMS и WebSphere MQ выполняют асинхронный вызов компонента SCA.

Обратите внимание, что при экспорте может передаваться информация о заголовках и пользовательских свойствах, полученных по протоколу, если включено распространение информации о контексте. В этом случае заголовки и пользовательские свойства доступны компонентам, связанным с экспортом. Более подробные сведения приведены в разделе “Распространение информации” справочной системы WebSphere Integration Developer Information Center.

В случае двунаправленной операции компонент возвращает ответ.

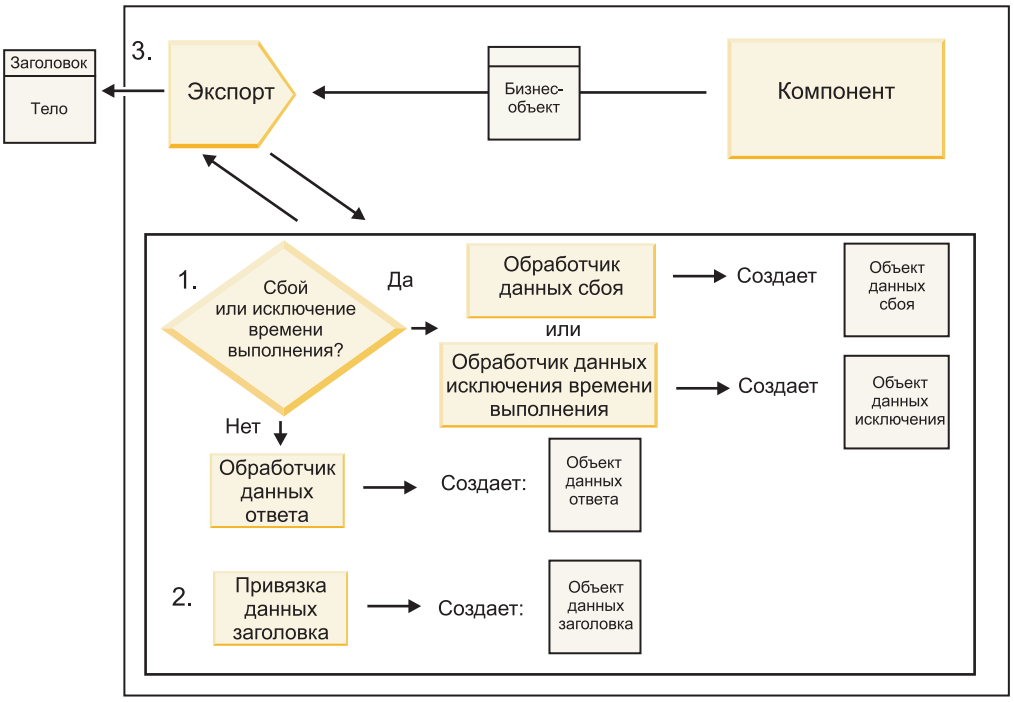


Рисунок 49. Возврат ответа через объект экспорта

Выполняется следующая последовательность действий:

1. Если привязка экспорта получает обычный ответ, то обработчик данных ответа или привязка данных в методе преобразуют бизнес-объект в ответ.

   Если в ответ получен сбой, то обработчик сбоев или привязка данных в методе преобразуют сбой в ответ о сбое.

   Только для привязок экспорта HTTP: если в ответ получена исключительная ситуация времени выполнения, то вызывается обработчик исключительных ситуаций, если он настроен.
2. Для привязок WebSphere MQ: привязка данных заголовка преобразует объекты данных заголовка в заголовки протокола.
3. Объект экспорта отправляет ответ службы по транспортному протоколу.

Передача информации при импорте

Компоненты отправляют запросы к службе, расположенной вне модуля, используя импорт. Запрос отправляется по транспортному протоколу, который определяется соответствующей привязкой.

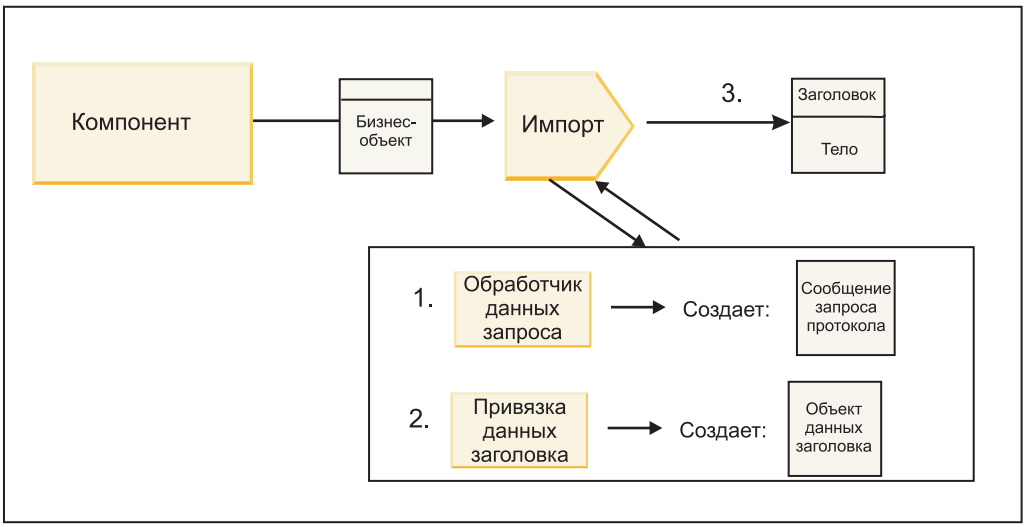


Рисунок 50. Передача данных из компонента в службу через объект импорта

Компонент вызывает функцию импорта, передавая в нее бизнес-объект запроса.

Примечание:

- Привязка импорта HTTP, привязка импорта веб-службы или привязка импорта EJB должна вызываться синхронно в компоненте.
- Привязка импорта JMS, базового JMS, MQ JMS или WebSphere MQ должна вызываться асинхронно.

После вызова функции импорта в компоненте выполняется следующая последовательность действий:

1. Обработчик данных запроса или привязка данных в методе преобразует бизнес-объект запроса в сообщение запроса протокола.
2. Для привязок WebSphere MQ: привязка данных заголовка из метода задает бизнес-объект заголовка в заголовке протокола.
3. Объект импорта вызывает службу с помощью запроса к службе, переданного через транспортный протокол.

Если выполняется двунаправленная операция, то служба возвращает ответ, и выполняется следующая последовательность действий:

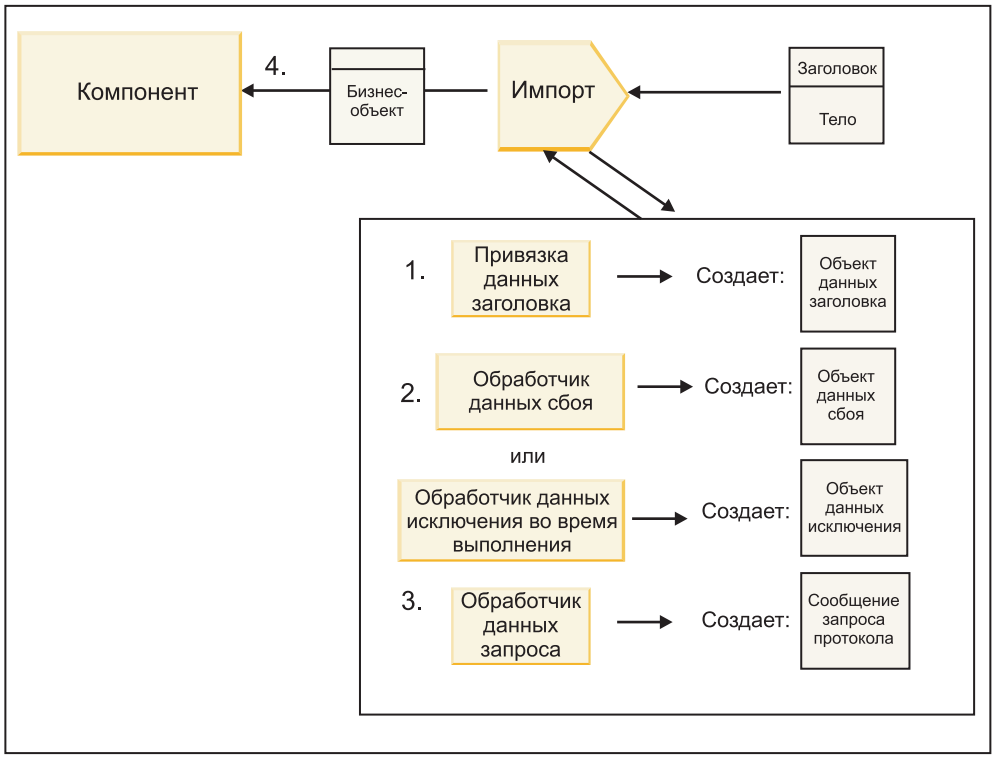


Рисунок 51. Возврат ответа через объект импорта

1. Для привязок WebSphere MQ: привязка данных заголовка преобразует заголовок протокола в объект данных заголовка.
2. Делается проверка того, является ли ответ сбоем.
   - Если ответ является сбоем, то селектор сбоев определяет соответствующий сбой WSDL. Обработчик сбоев из метода преобразует сбой в ответ о сбое.
   - Если ответ является исключительной ситуацией времени выполнения, то вызывается обработчик исключительных ситуаций, если он настроен.
3. Обработчик данных ответа или привязка из метода преобразуют ответ в бизнес-объект ответа.
4. Объект импорта возвращает бизнес-объект ответа в компонент.

Конфигурация привязки экспорта и импорта

Одной из основных особенностей привязок экспорта и импорта является преобразование (десериализация) данных из внутреннего формата проводника в бизнес-объект или преобразование (сериализация) бизнес-объекта во внутренний формат проводника. Для привязок экспорта можно дополнительно выбрать функцию, определяющую операцию над данными. И для привязок экспорта, и для привязок импорта можно указать способ обработки ошибок, возникающих во время обработки.

Кроме того, в привязках можно указать параметры, специфичные для типа транспорта. Например, для привязки HTTP можно указать URL конечной точки. Специфичная для транспорта информация, настраиваемая для привязки HTTP, описана в разделах “Создание привязки импорта HTTP” и “Создание привязки экспорта HTTP”. Информация о других привязках также приведена в Information Center.

Преобразование формата данных в объектах импорта и экспорта

При настройке привязки экспорта или импорта в IBM Integration Designer в числе прочих параметров указывается формат данных привязки.

- Для привязок экспорта, через которые приложение-клиент отправляет запросы и получает ответы от компонента SCA, требуется указывать формат внутренних данных. Система выбирает обработчик данных

или привязку данных с учетом формата для преобразования внутренних данных в бизнес-объект (который применяется в компоненте SCA) и для обратного преобразования бизнес-объекта во внутренние данные (то есть в ответ для приложения-клиента).

- Для привязок импорта, через которые компонент SCA отправляет запросы и получает ответы от внешней службы, необходимо указывать формат внутренних данных. Система выбирает обработчик данных или привязку данных с учетом формата для преобразования бизнес-объекта во внутренние данные и обратного преобразования.

IBM Business Process Manager предоставляет набор предопределенных форматов данных, а также обработчиков данных или привязок данных, поддерживающих эти форматы. При необходимости можно создать собственные обработчики данных и зарегистрировать для них формат данных. За дополнительной информацией обратитесь к разделу “Разработка обработчиков данных” в справочной системе IBM Integration Designer Information Center.

- *Обработчики данных* не зависят от протокола и выполняют преобразование данных в другой формат. В IBM Business Process Manager обработчики данных как правило преобразуют внутренние данные (такие как XML, CSV или COBOL) в бизнес-объект, а также бизнес-объект во внутренние данные. Поскольку обработчики не зависят от протокола, один обработчик можно использовать в различных привязках экспорта и импорта. Например, обработчик данных XML можно использовать как с привязками экспорта и импорта HTTP, так и с привязками экспорта и импорта JMS.
- *Привязки данных* также выполняют преобразование внутренних данных в бизнес-объект (и наоборот), однако они зависят от протокола. Например, привязку данных HTTP можно использовать только с привязками экспорта и импорта HTTP. В отличии от обработчиков данных, привязку данных HTTP нельзя использовать в привязке импорта или экспорта MQ.

  Примечание: Три привязки данных HTTP (HTTPStreamDataBindingSOAP, HTTPStreamDataBindingXML и HTTPServiceGatewayDataBinding) устарели и больше не используются в IBM Business Process Manager версии 7.0. При любой возможности вместо них следует использовать обработчики данных.

Как уже было отмечено ранее, при необходимости можно создать собственные обработчики данных. Кроме того, можно создать свои привязки данных, однако рекомендуется создавать именно обработчики данных, поскольку их можно использовать в любых привязках.

Обработчики данных:

Обработчики данных настраиваются для привязок импорта и экспорта и предназначены для преобразования данных в другой формат, не связанный с определенным протоколом. Несколько обработчиков данных входят в состав продукта и при необходимости можно создать собственный обработчик данных. Обработчик данных можно связать с привязкой импорта или экспорта на одном из двух уровней: его можно связать со всеми операциями интерфейса экспорта или импорта либо с определенной операцией над запросом или ответом.

Предопределенные обработчики данных

Необходимый обработчик данных можно выбрать в IBM Integration Designer.

В следующей таблице перечислены предопределенные обработчики данных и описано, каким образом они выполняют преобразование входящих и исходящих данных.

Примечание: За исключением специально отмеченных случаев, обработчики данных можно использовать вместе с привязками JMS, базового JMS, MQ JMS, WebSphere MQ и HTTP.
За более подробной информацией обратитесь к разделу “Обработчики данных” справочной системы Integration Designer Information Center.

Таблица 52. Предопределенные обработчики данных

Обработчик данных	Внутренние данные в бизнес-объект	Бизнес-объект во внутренние данные
ATOM	Преобразует ленты новостей ATOM в бизнес-объект ленты ATOM.	Сериализует бизнес-объект ленты ATOM и преобразует его в ленты новостей ATOM.
С разделителем	Преобразует данные с разделителем в бизнес-объект.	Сериализует бизнес-объект и преобразует его в данные с разделителем, в том числе CSV.
Фиксированной ширины	Преобразует данные фиксированной ширины в бизнес-объект.	Сериализует бизнес-объект и преобразует его в данные фиксированной ширины.
Обрабатывается WTX	Передает данные для преобразования в WebSphere Transformation Extender (WTX). Имя карты связей WTX определяется обработчиком данных.	Передает данные для преобразования в WebSphere Transformation Extender (WTX). Имя карты связей WTX определяется обработчиком данных.
Обрабатывается WTX Invoker	Передает данные для преобразования в WebSphere Transformation Extender (WTX). Имя карты связей WTX предоставляется пользователем.	Передает данные для преобразования в WebSphere Transformation Extender (WTX). Имя карты связей WTX предоставляется пользователем.
JAXB	Преобразует объекты Java Bean в бизнес-объект с помощью правил преобразования, определенных в спецификации Java Architecture for XML Binding (JAXB).	Десериализует бизнес-объект и преобразует его в объекты Java Bean с помощью правил преобразования, определенных в спецификации JAXB.
JAXWS **Примечание:** Обработчик данных JAXWS можно использовать только в привязке EJB.	Применяется в привязке EJB для преобразования объекта Java с ответом или исключительной ситуацией в бизнес-объект ответа согласно правилам преобразования, определенным в спецификации Java API for XML Web Services (JAX-WS).	Применяется в привязке EJB для преобразования бизнес-объекта в параметры метода Java в соответствии с правилами преобразования, определенными в спецификации JAX-WS.
JSON	Преобразует данные JSON в бизнес-объект.	Сериализует бизнес-объект и преобразует его в данные JSON.
Внутреннее тело	Преобразует внутренние байты, текст, карту, поток или объект в бизнес-объект одного из пяти типов (текст, байты, карта, поток или объект).	Преобразует любой из пяти типов базовых бизнес-объектов в байты, текст, карту, поток или объект.
SOAP	Преобразует сообщение SOAP (включая заголовок) в бизнес-объект.	Сериализует бизнес-объект и преобразует его в сообщение SOAP.
XML	Преобразует данные XML в бизнес-объект.	Сериализует бизнес-объект и преобразует его в данные XML.
UTF8XMLDataHandler	Преобразует данные XML в кодировке UTF-8 в бизнес-объект.	Сериализует бизнес-объект и преобразует его в данные XML в кодировке UTF-8 при отправке сообщения.

Создание обработчика данных

Подробная информация о создании обработчика данных приведена в разделе “Разработка обработчиков данных” справочной системы Integration Designer Information Center.

Привязки данных:

Привязки данных настраиваются для привязок импорта и экспорта и предназначены для преобразования данных в другой формат. Для каждого протокола предусмотрена своя привязка данных. Несколько привязок данных входят в состав продукта и при необходимости можно создать собственную привязку данных. Привязку данных можно связать с привязкой импорта или экспорта на одном из двух уровней: ее можно связать со всеми операциями интерфейса экспорта или импорта либо с определенной операцией над запросом или ответом.

Для выбора необходимой привязки данных и создания собственной привязки данных используется IBM Integration Designer. Инструкции по созданию привязок данных приведены в разделе “Обзор привязок JMS, MQ JMS и базового JMS” в справочной системе IBM Integration Designer Information Center.

Привязки JMS

В следующей таблице перечислены привязки данных, которые можно использовать вместе со следующими объектами:

- Привязки JMS
- Базовые привязки JMS
- Привязки WebSphere MQ JMS

В таблице приведено описание задач, выполняемых привязками данных.

Таблица 53. Предопределенные привязки данных для привязок JMS

Привязка данных	Внутренние данные в бизнес-объект	Бизнес-объект во внутренние данные
Сериализованный объект Java	Преобразует сериализованный объект Java в бизнес-объект (который преобразуется в тип входа или выхода в WSDL).	Преобразует бизнес-объект в сериализованный объект Java из сообщения объекта JMS.
Байты с оболочкой	Извлекает байты из входящего байтового сообщения JMS и преобразует их в бизнес-объект JMSBytesBody.	Извлекает байты из бизнес-объекта JMSBytesBody и добавляет их внутрь исходящего байтового сообщения JMS
Запись карты связей с оболочкой	Извлекает имя, значение и тип из каждой записи входящего сообщения карты связей JMS и создает список бизнес-объектов MapEntry. Полученный список добавляется в бизнес-объект JMSMapBody	Извлекает имя, значение и тип из списка MapEntry, расположенного внутри бизнес-объекта JMSMapBody, и создает соответствующие записи в исходящем сообщении карты связей JMS.
Объект с оболочкой	Извлекает объект из входящего сообщения объекта JMS и добавляет его в бизнес-объект JMSObjectBody.	Извлекает объект из бизнес-объекта JMSObjectBody и добавляет его в исходящее сообщение объекта JMS.
Текст с оболочкой	Извлекает текст из входящего текстового сообщения JMS и добавляет его в бизнес-объект JMSTextBody.	Извлекает текст из бизнес-объекта JMSTextBody и добавляет его в исходящее текстовое сообщение JMS.

Привязки WebSphere MQ

В следующей таблице перечислены привязки данных, которые можно использовать вместе с WebSphere MQ, и описаны задачи, выполняемые этими привязками.

Таблица 54. Предопределенные привязки данных для привязок WebSphere MQ

Привязка данных	Внутренние данные в бизнес-объект	Бизнес-объект во внутренние данные
Сериализованный объект Java	Преобразует сериализованный объект Java из входящего сообщения в бизнес-объект (который преобразуется в тип входа или выхода в WSDL).	Преобразует бизнес-объект в сериализованный объект Java в исходящем сообщении
Байты с оболочкой	Извлекает байты из неструктурированного байтового сообщения MQ и преобразует их в бизнес-объект JMSBytesBody.	Извлекает байты из бизнес-объекта JMSBytesBody и преобразует байты в исходящее неструктурированное байтовое сообщение MQ.
Текст с оболочкой	Извлекает текст из неструктурированного текстового сообщения MQ и преобразует его в бизнес-объект JMSTextBody.	Извлекает текст из бизнес-объекта JMSTextBody и преобразует его в неструктурированное текстовое сообщение MQ.
Запись потока с оболочкой	Извлекает имя и тип из каждой записи входящего потокового сообщения JMS и создает список бизнес-объектов StreamEntry. Полученный список добавляется в бизнес-объект JMSStreamBody.	Извлекает имя и тип из списка StreamEntry, содержащегося в бизнес-объекте JMSStreamBody, и создает соответствующие записи в исходящем сообщении JMSStreamMessage.

В дополнение к привязкам данных, описанным в разделе Табл. 25 на стр. 62, WebSphere MQ поддерживает привязки данных заголовка. Дополнительная информация приведена в IBM Integration Designer Information Center.

Привязки HTTP

В следующей таблице перечислены привязки данных, которые могут применяться при работе с HTTP, и описаны задачи, выполняемые этими привязками данных.

Таблица 55. Предопределенные привязки данных для привязок HTTP

Привязка данных	Внутренние данные в бизнес-объект	Бизнес-объект во внутренние данные
Байты с оболочкой	Извлекает байты из текла входящего сообщения HTTP и добавляет их в бизнес-объект HTTPBytes.	Извлекает байты из бизнес-объекта HTTPBytes и добавляет их в тело исходящего сообщения HTTP.
Текст с оболочкой	Извлекает текст из тела входящего сообщения HTTP и добавляет его в бизнес-объект HTTPText.	Извлекает текст из бизнес-объекта HTTPText и добавляет его в тело исходящего сообщения HTTP.

Селекторы функций в привязках экспорта

Селектор функции указывает, какую операцию над данными необходимо выполнить при получении сообщения с запросом. Селекторы функций настраиваются в привязке экспорта.

Для примера рассмотрим экспорт SCA, предназначенный для экспорта интерфейса. Интерфейс содержит две операции: Create и Update. Экспорт использует привязку JMS, которая считывает сообщения из очереди.

Когда сообщение поступает в очередь, данные о нем передаются в объект экспорта, но дополнительно требуется определить, какую из двух операций интерфейса следует вызвать для связанного компонента. Операция выбирается с помощью селектора функций и конфигурации привязки экспорта.

Селектор функций возвращает внутреннее имя функции (имя функции в клиентской системе, отправившей сообщение). Внутреннее имя функции преобразуется в имя операции или функции интерфейса, связанного с экспортом. Например, на следующем рисунке селектор функций возвращает внутреннее имя функции (CRT)

из полученного сообщения, это имя преобразуется в операцию Create, после чего компоненту SCA отправляется бизнес-объект с операцией Create.

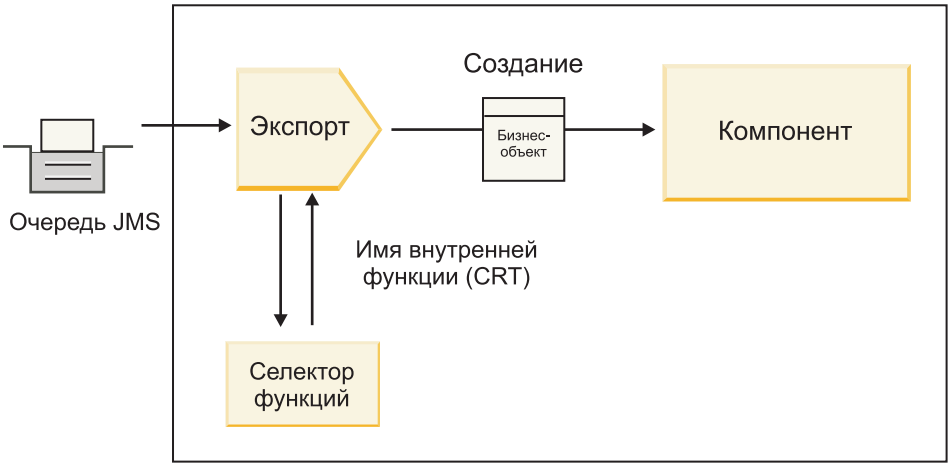


Рисунок 52. Селектор функций

Если интерфейс содержит только одну операцию, то селектор функций указывать не нужно.

В составе продукта поставляется несколько готовых селекторов функций, которые описаны в приведенных ниже разделах.

Привязки JMS

В следующей таблице указаны селекторы функций, которые можно использовать вместе со следующими объектами:

- Привязки JMS
- Базовые привязки JMS
- Привязки JMS WebSphere MQ

Таблица 56. Предопределенные селекторы функций для привязок JMS

Селектор функций	Описание
Селектор функций JMS для простых привязок данных JMS	Для выбора имени операции применяется свойство сообщения JMSType.
Селектор функций для свойства заголовка JMS	Возвращает значение строкового свойства JMS с именем TargetFunctionName из заголовка.
Селектор функций для шлюза служб JMS	Определяет тип запрашиваемой операции (однонаправленная или двунаправленная), используя заданное клиентом свойство JMSReplyTo.

Привязки WebSphere MQ

В следующей таблице указаны селекторы функций, которые можно применять в привязках WebSphere MQ.

Таблица 57. Предопределенные селекторы функций для привязок WebSphere MQ

Селектор функций	Описание
Селектор функций handleMessage MQ	Возвращает handleMessage в качестве значения, которое преобразуется с помощью привязок метода объекта экспорта в имя операции интерфейса.

Таблица 57. Предопределенные селекторы функций для привязок WebSphere MQ (продолжение)

Селектор функций	Описание
В MQ используется селектор функций по умолчанию JMS	Считывает внутреннюю операцию из свойства TargetFunctionName папки в заголовке MQRFH2.
В MQ используется формат тела сообщения в качестве внутренней функции	Находит поле формата в последнем заголовке и возвращает его значение в виде строки.
Селектор функций для типа MQ	Создает метод в привязке экспорта путем получения данных по URL, содержащему свойства Msd, Set, Type и Format и указанному в заголовке MQRFH2.
Селектор функций для шлюза служб MQ	Определяет имя операции с помощью свойства MsgType из заголовка MQMD.

Привязки HTTP

В следующей таблице указаны селекторы функций, которые можно использовать в привязках HTTP.

Таблица 58. Предопределенные селекторы функций для привязок HTTP

Селектор функций	Описание
Селектор функций HTTP, основанный на заголовке TargetFunctionName	Использует настроенное клиентом свойство заголовка HTTP TargetFunctionName для определения операции, которую следует вызывать во время выполнения из объекта экспорта.
Селектор функций HTTP, основанный на URL и методе HTTP	Использует относительный путь из URL и дополняет его методом HTTP, указанным клиентом, для выбора внутренней операции, определенной в экспорте.
Селектор функций для шлюза служб HTTP, основанный на URL и имени операции	Выбирает вызываемый метод по URL, если в URL запроса указано "operationMode = oneway".

Примечание: Используя IBM Integration Designer, можно создать собственный селектор функций. Информация о создании селектора функций приведена в справочной системе IBM Integration Designer Information Center. Например, инструкции по созданию селектора функций для привязок WebSphere MQ приведены в разделе “Обзор селекторов функций MQ”.

Обработка сбоев

Привязки импорта и экспорта можно настроить для обработки сбоев (например, исключительных ситуаций бизнес-уровня), возникающих во время работы, указав обработчики данных сбоев. Обработчик данных сбоев можно настроить на трех уровнях: обработчик данных сбоев можно связать со сбоем, с операцией или со всеми операциями привязки.

Обработчик данных сбоев обрабатывает данные сбоя и преобразует их в тот формат, в котором они могут быть отправлены привязкой импорта или экспорта.

- Для привязки экспорта обработчик данных сбоев преобразует бизнес-объект исключительной ситуации, полученный от компонента, в ответное сообщение, которое может быть обработано приложением-клиентом.
- Для привязки импорта обработчик данных сбоев преобразует данные о сбое или ответное сообщение, полученное от службы, в бизнес-объект исключительной ситуации, который может быть обработан компонентом SCA.

Привязка импорта вызывает селектор сбоев, который определяет, является ли ответное сообщение обычным ответом, сбоем бизнес-уровня или исключительной ситуацией времени выполнения.

С помощью привязки можно настроить обработчик данных сбоев для отдельного сбоя, операции или всех операций.

- Если обработчик данных сбоев настроен на всех трех уровнях, то вызывается тот из них, который связан со сбоем.
- Если обработчик данных сбоев настроен на уровне операции и привязки, то вызывается обработчик, связанный с операцией.

Для настройки обработки сбоев в IBM Integration Designer используются два редактора. В редакторе интерфейса можно определить сбои операции. После создания привязки с этим интерфейсом редактор на панели свойств позволяет настроить способ обработки сбоев. За дополнительной информацией обратитесь к разделу “Селектор сбоя” в справочной системе IBM Integration Designer Information Center.

Обработка сбоев в привязках экспорта:

Если во время обработки запроса приложения-клиента возникает сбой, то привязка экспорта может вернуть информацию о нем клиенту. Параметры обработки и возврата сбоев настраиваются в привязке экспорта.

Привязка экспорта настраивается с помощью IBM Integration Designer.

Во время обработки запроса клиент вызывает объект экспорта, передавая ему запрос, а объект экспорта вызывает компонент SCA. Во время обработки запроса компонент SCA может вернуть ответ с данными или сгенерировать исключительную ситуацию бизнес-уровня либо исключительную ситуацию времени выполнения. В последнем случае привязка экспорта преобразует исключительную ситуацию в сообщение о сбое и отправляет его клиенту, как показано на следующем рисунке и описано в приведенных ниже разделах.

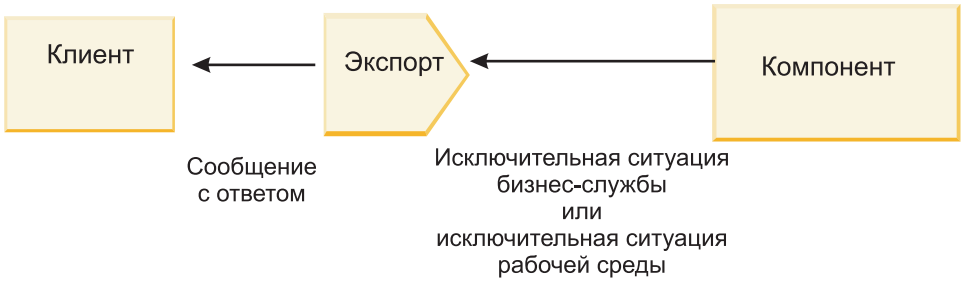


Рисунок 53. Как информация о сбое передается из компонента в клиент через привязку экспорта

Для обработки сбоев можно создать пользовательский обработчик данных или привязку данных.

Сбои бизнес-уровня

Сбои бизнес-уровня - это ошибки или исключительные ситуации в работе бизнеса, которые возникают во время обработки.

Изучите следующий интерфейс, содержащий операцию createCustomer. В этой операции определено два сбоя бизнес-уровня: CustomerAlreadyExists и MissingCustomerId.

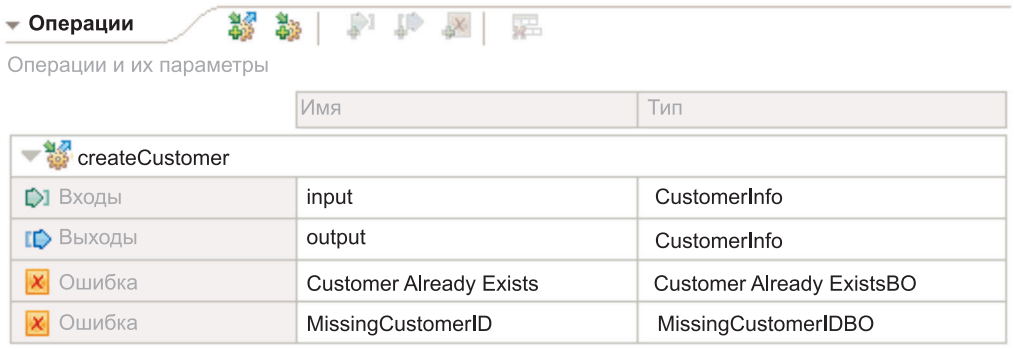


Рисунок 54. Интерфейс с двумя сбоями

В этом примере при попытке клиента создать уже существующего получателя (для компонента SCA) компонент генерирует сбой CustomerAlreadyExists и передает его в объект экспорта. Объект экспорта должен вернуть этот сбой бизнес-уровня тому клиенту, который отправил запрос. Для этого используется обработчик данных сбоя, который настроен в привязке экспорта.

Если привязка экспорта получает сбой бизнес-уровня, выполняются следующие действия:

1. Привязка определяет, какой обработчик данных сбоя следует вызвать для обработки данного сбоя. Если исключительная ситуация бизнес-уровня содержит имя сбоя, то вызывается настроенный для этого сбоя обработчик данных. В противном случае имя сбоя определяется по типу сбоя.
2. Привязка вызывает обработчик данных сбоя и передает в него объект данных из исключительной ситуации бизнес-уровня.
3. Обработчик данных сбоя преобразует объект данных сбоя в ответное сообщение и возвращает его в привязку экспорта.
4. Объект экспорта возвращает ответное сообщение клиенту.

Если исключительная ситуация бизнес-уровня содержит имя сбоя, то вызывается настроенный для этого сбоя обработчик данных. В противном случае имя сбоя определяется по типу сбоя.

Исключительные ситуации времени выполнения

Исключительная ситуация времени выполнения - это исключительная ситуация, которая возникает в приложении SCA во время обработки запроса и не связана со сбоем бизнес-уровня. В отличие от сбоев бизнес-уровня исключительные ситуации времени выполнения не требуется определять в интерфейсе.

В ряде случаев имеет смысл передавать исключительные ситуации времени выполнения в приложение-клиент, для того чтобы это приложение могло выполнить соответствующие действия.

Например, если клиент отправляет запрос на создание получателя в компонент SCA, и во время обработки запроса возникает ошибка прав доступа, то компонент генерирует исключительную ситуацию времени выполнения. Эту исключительную ситуацию необходимо вернуть в клиент, для того чтобы он принял необходимые меры по получению прав доступа. Для этого применяется обработчик данных исключительной ситуации времени выполнения, настроенный в привязке экспорта.

Примечание: Обработчик данных исключительной ситуации времени выполнения можно настроить только в привязках HTTP.

Исключительная ситуация времени выполнения обрабатывается так же, как и сбой бизнес-уровня. Если обработчик данных исключительной ситуации времени выполнения настроен, то выполняются следующие действия:

1. Привязка экспорта вызывает необходимый обработчик данных и передает в него исключительную ситуацию времени выполнения.

2. Обработчик данных преобразует объект данных сбоя в ответное сообщение и возвращает его в привязку экспорта.
3. Объект экспорта возвращает ответное сообщение клиенту.

Обработка сбоев и исключительных ситуаций времени выполнения является необязательной. Если сбои или исключительные ситуации не требуется передавать клиенту, то не настраивайте обработчик данных сбоев или исключительной ситуации времени выполнения.

Обработка ошибок привязками импортов:

Некоторый компонент использует импорт для отправки запроса некоторой службе вне модуля. Если во время обработки запроса происходит ошибка, то служба возвращает ошибку привязке импорта. Привязку импорта можно настроить с указанием, как должна быть обработана ошибка и возвращена компоненту.

Для настройки привязки импорта используется IBM Integration Designer. Также можно указать обработчик данных ошибок (или привязку данных), а также указать селектор ошибок.

Обработчики данных ошибок

Служба, которая обрабатывает отправки запросов привязке импорта, информацию об ошибке в виде исключительной ситуации или ответное сообщение, которое содержит данные ошибки.

Привязка импорта преобразует исключительную ситуацию службы или ответное сообщение в исключительную ситуации бизнес-процесса службы или исключительную ситуацию во время выполнения службы, как показано на следующем рисунке и описано в следующих разделах.

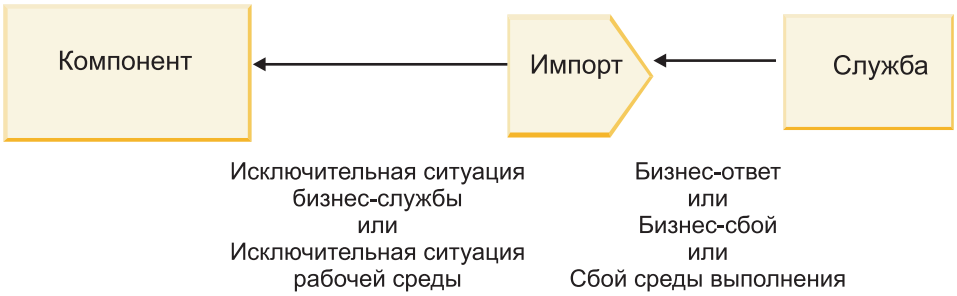


Рисунок 55. Отправка информации об ошибке из службы через импорт компоненту

Для обработки ошибок можно создать пользовательский обработчик данных или привязку данных.

Селекторы сбоев

При настройке привязки импорта можно задать селектор ошибок. Селектор ошибок определяет, является ли ответ импорта действительно ответом, исключительной ситуацией бизнес-процесса или ошибкой во время выполнения. Из тела или заголовка ответа он также определяет собственное имя ошибки, которое преобразуется конфигурацией привязки в имя ошибки в связанном интерфейсе.

Для работы с импортами JMS, MQ JMS, Generic JMS, WebSphere MQ и HTTP доступны два типа готовых селекторов ошибок:

Таблица 59. Готовые селекторы ошибок

Тип селектора ошибок	Описание
На основе заголовков	На основе заголовков входящего ответного сообщения определяет, является ли ответное сообщение ошибкой бизнес-процесса, исключительной ситуацией во время выполнения или нормальным сообщением.
SOAP	Определяет, является ли ответное сообщение SOAP нормальным ответом, ошибкой бизнес-процесса или исключительной ситуацией во время выполнения.

Далее приведены примеры селекторов ошибок на основе заголовков и селектора ошибок SOAP.

- Селектор ошибок на основе заголовков

  Для того чтобы приложение могло показать, что входящее сообщение является ошибкой бизнес-процесса, для ошибок бизнес-процесса во входящем сообщении должно быть два заголовка, которые выглядят следующим образом:

  ```
  Имя заголовка = FaultType, Значение заголовка = Бизнес-процесс
  Имя заголовка = FaultName, Значение заголовка = <пользовательское собственное имя ошибки>
  ```

  Для того чтобы приложение могло показать, что входящее сообщение является исключительной ситуацией во время выполнения, во входящем сообщении должен быть один заголовок, который выглядят следующим образом:

  ```
  Имя заголовка = FaultType, Значение заголовка = Во время выполнения
  ```

- Селектор ошибок SOAP

  Ошибка бизнес-процесса может быть передана в составе сообщения SOAP со следующим пользовательским заголовком SOAP. "CustomerAlreadyExists" – это имя ошибки в данном случае.

  ```
  <ibmSoap:BusinessFaultName
  xmlns:ibmSoap="http://www.ibm.com/soap">CustomerAlreadyExists
  <ibmSoap:BusinessFaultName>
  ```

Селектор ошибок произволен. Если селектор ошибок не указан, то привязка импорта не сможет определить тип ответа. При этом привязка рассматривает его как ответ бизнес-процесса и вызывает обработчик данных ответа или привязку данных.

Можно создать пользовательский селектор ошибок. Этапы создания пользовательского селектора ошибок приведены в разделе “Разработка пользовательского селектора ошибок” справочной системы IBM Integration Designer Information Center.

Ошибки бизнес-процессов

Ошибка бизнес-процесса может наблюдаться, когда в обработке запроса присутствует ошибка. Например, в случае отправки запроса на создание клиента, который уже существует, служба отправляет исключительную ситуацию бизнес-процесса привязке импорта.

После получения исключительной ситуации бизнес-процесса привязкой этапы обработки зависят от того, настроен ли селектор ошибок для привязки.

- Если селектор ошибок не настроен, то привязка вызывает обработчик данных ответа или привязку данных.
- Если селектор ошибок настроен, то выполняется следующая обработка:
  1. Привязка импорта вызывает селектор ошибок для определения, является ли ответ ошибкой бизнес-процесса, ответом бизнес-процесса или ошибкой во время выполнения.
  2. Если ответ является ошибкой бизнес-процесса, то привязка импорта вызывает селектор ошибок для определения собственного имени ошибки.

3. Привязка импорта определяет ошибку WSDL соответственно собственному имени ошибки, возвращенному селектором ошибок.
4. Привязка импорта определяет обработчик данных ошибки, который настроен на ошибку WSDL.
5. Привязка импорта вызывает обработчик данных ошибки с данными ошибки.
6. Обработчик данных ошибки преобразует данные ошибки в объект данных и возвращает его привязке импорта.
7. Привязка импорта создает объект исключительной ситуации бизнес-процесса службы с объектом данных и именем ошибки.
8. Импорт возвращает объект исключительной ситуации бизнес-процесса службы компоненту.

Исключительные ситуации выполнения

Исключительная ситуация во время выполнения может наблюдаться, когда имеет место неполадка со связью со службой. Обработка исключительной ситуации во время выполнения аналогична обработке исключительной ситуации бизнес-процесса. Если селектор ошибок настроен, то выполняется следующая обработка:

1. Привязка импорта вызывает соответствующий обработчик данных исключительной ситуации во время выполнения с данными исключительной ситуации.
2. Обработчик данных исключительной ситуации во время выполнения преобразует данные исключительной ситуации в объект исключительной ситуации во время выполнения службы и возвращает его привязке импорта.
3. Импорт возвращает объект исключительной ситуации во время выполнения службы компоненту.

Взаимодействие между модулями SCA и службами Open SCA

IBM WebSphere Application Server V7.0 Feature Pack for Service Component Architecture (SCA) предоставляет простую, но мощную программную модель для создания приложений на основе спецификаций Open SCA. Модули SCA IBM Business Process Manager используют привязки импортов и экспортов для взаимодействия со службами Open SCA, разработанными в среде Rational Application Developer и находящимися в WebSphere Application Server Feature Pack for Service Component Architecture.

Приложение SCA вызывает приложение Open SCA при помощи привязки импорта. Приложение SCA получает вызов от приложения Open SCA при помощи привязки экспорта. Список поддерживаемых привязок приведен в “Вызов служб через стыковочные привязки” на стр. 71.

Вызов служб Open SCA из модулей SCA

Приложения SCA, разработанные с помощью IBM Integration Designer, могут вызывать приложения Open SCA, разработанные в среде Rational Application Developer. В этом разделе приведен пример вызова службы Open SCA из модуля SCA с помощью привязки импорта SCA.

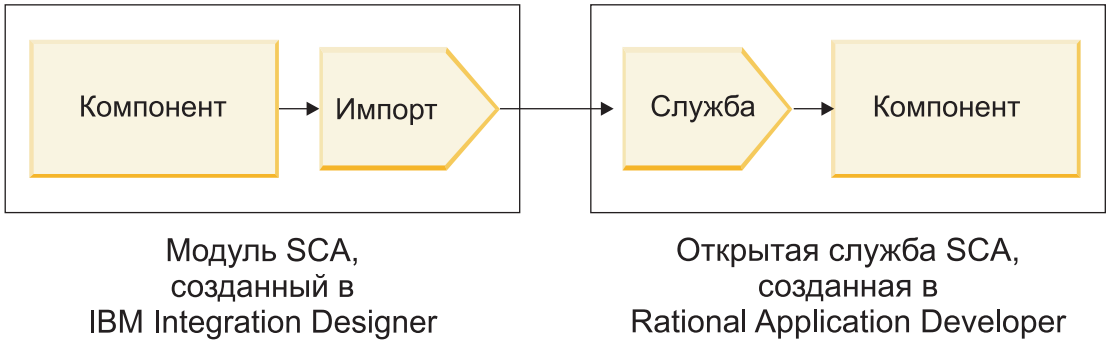


Рисунок 56. Компонент в модуле SCA, вызывающий службу Open SCA

Для вызова службы Open SCA не требуется никакой специальной настройки.

Для соединения со службой Open SCA при помощи привязки импорта SCA в привязке импорта указываются имя компонента и имя службы Open SCA.

1. Для получения имени целевого компонента и службы из комплекса Open SCA выполните следующее:
   a. Откройте вкладку **Свойства**, выбрав **Окно > Показать панель > Свойства**.
   b. Откройте редактор комплекса, дважды щелкнув на диаграмме комплекса, содержащей компонент и службу. Например, для компонента с именем **customer** схемой комплекса является **customer.диаграмма-комплекса**.
   c. Щелкните на целевом компоненте.
   d. В поле **Имя** на вкладке **Свойства** укажите имя целевого компонента.
   e. Щелкните на значке службы, связанной с компонентом.
   f. В поле **Имя** на вкладке **Свойства** укажите имя службы.
2. Для настройки импорта IBM Business Process Manager на соединение со службой Open SCA выполните следующее:
   a. В IBM Integration Designer перейдите на вкладку **Свойства** импорта SCA, используемого для соединения со службой Open SCA.
   b. В поле **Имя модуля** введите имя компонента из этапа 1d на стр. 70.
   c. В поле **Имя экспорта** введите имя службы из этапа 1f на стр. 70.
   d. Сохраните свою работу нажатием клавиш Ctrl+S.

Вызов модулей SCA из служб Open SCA

Приложения Open SCA, разработанные в среде Rational Application Developer, могут вызывать приложения SCA, разработанные с помощью IBM Integration Designer. В этом примере приведен пример вызова модуля SCA (при помощи привязки экспорта SCA) из службы Open SCA.

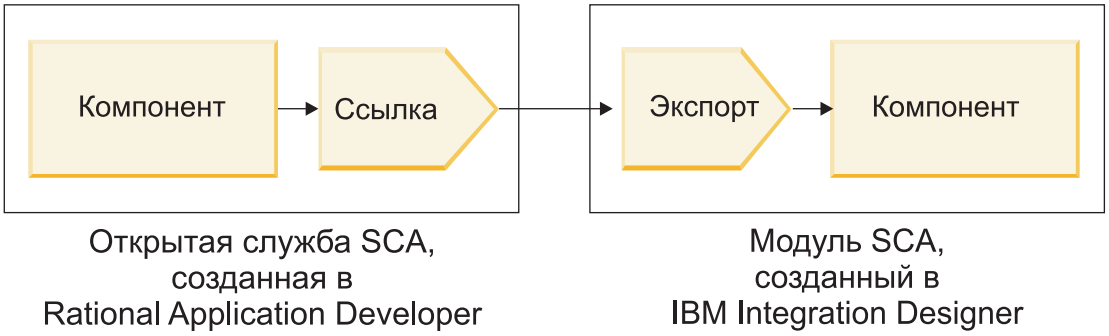


Рисунок 57. Откройте вызывающий службу SCA компонент в модуле SCA

Для соединения с компонентом SCA при помощи привязки ссылки Open SCA укажите имя модуля и имя экспорта.

1. Для получения имени целевого модуля и экспорта выполните следующее:
   a. В IBM Integration Designer откройте модуль в редакторе сборки, дважды щелкнув на модуле.
   b. Щелкните на экспорте.
   c. В поле **Имя** на вкладке **Свойства** укажите имя экспорта.
2. Настройте ссылку SCA, которая будет использоваться для соединения с модулем IBM Business Process Manager и экспортом:
   a. В Rational Application Developer Откройте редактор комплекса, дважды щелкнув на диаграмме комплекса, содержащей компонент и службу.
   b. Щелкните на значке ссылки, чтобы открыть свойства ссылки на вкладке **Свойства**.
   c. Откройте вкладку **Привязка** на левой стороне страницы.
   d. Щелкните на **Привязки** и затем на **Добавить**.
   e. Выберите привязку **SCA**.
   f. В поле **Uri** введите имя модуля IBM Business Process Manager, косую черту (“/”) и затем имя экспорта (которое было определено на этапе 1c на стр. 71).
   g. Нажмите кнопку **OK**.
   h. Сохраните свою работу нажатием клавиш Ctrl+S.

Вызов служб через стыковочные привязки

Следующие привязки поддерживаются для возможности взаимодействия со службой Open SCA.

- Привязка SCA

  В IBM Business Process Manager, когда модуль SCA вызывает службу Open SCA при помощи привязки импорта SCA, поддерживаются следующие стили вызовов:

  – Асинхронный (односторонний)
  – Синхронный (запрос/ответ)

  Интерфейс импорта SCA и интерфейс службы Open SCA должны использовать стыкуемость веб-служб (WS-I), совместимую с интерфейсом WSDL.

  Отметим, что привязка SCA поддерживает транзакцию и контекстное распространение защиты.

- Привязка веб-службы (JAX-WS) с протоколом SOAP1.1/HTTP или SOAP1.2/HTTP

  Интерфейс импорта SCA и интерфейс службы Open SCA должны использовать стыкуемость веб-служб (WS-I), совместимую с интерфейсом WSDL.

  Кроме того, поддерживаются следующие качества службы:

  – Атомарная транзакция веб-служб

  - Защита веб-служб
- Привязка EJB

  Когда используется привязка EJB, интерфейс Java используется для определения взаимодействия между модулем SCA и службой Open SCA.

  Отметим, что привязка EJB поддерживает транзакцию и контекстное распространение защиты.
- Привязки JMS

  Интерфейс импорта SCA и интерфейс службы Open SCA должны использовать стыкуемость веб-служб (WS-I), совместимую с интерфейсом WSDL.

  Поддерживаются следующие провайдеры JMS:
  - WebSphere Platform Messaging (привязка JMS)
  - WebSphere MQ (привязка MQ JMS)

Примечание: Business Graphs не задействуется ни через какие привязки SCA и поэтому не поддерживаются в интерфейсах, используемых для взаимодействия с WebSphere Application Server Feature Pack for Service Component Architecture.

Типы привязок

Вместе с объектами импорта и экспорта используются *привязки* определенных протоколов, определяющие способ передачи данных в модуль и из него.

Выбор типа связывания

При создании приложения необходимо знать, как выбрать привязку, отвечающую требованиям приложения.

В IBM Integration Designer доступны привязки, предоставляющие различные возможности. Используя этот список, можно определить тип привязки, который в наибольшей степени отвечает требованиям приложения.

Привязку *Service Component Architecture (SCA)* рекомендуется выбирать при соблюдении следующих условий:

- Все службы содержатся в модулях, то есть внешние службы не используются.
- Требуется разделить функции по различным модулям SCA, которые взаимодействуют непосредственно друг с другом.
- Используются сильносвязанные модули

Привязку *Веб-служба* рекомендуется использовать, если выполняются следующие условия:

- Требуется обеспечить доступ к внешней службе или предоставить службу по Интернету.
- Используются слабосвязанные службы.
- Предпочтителен синхронный обмен данными, т.е. запрос от одной службы может ожидать ответа другой.
- Внешняя служба или предоставляемая служба использует протокол SOAP/HTTP или SOAP/JMS.

Связывание *HTTP* рекомендуется использовать, если выполняются следующие условия:

- Требуется общаться к внешней службе через Интернет или предоставлять службу через Интернет, и при этом используются другие веб-службы, такие как GET, PUT и DELETE.
- Используются слабосвязанные службы.
- Предпочтителен синхронный обмен данными, т.е. запрос от одной службы может ожидать ответа другой.

Привязку *Enterprise JavaBeans (EJB)* рекомендуется выбирать при соблюдении следующих условий:

- Связывание предназначено для импортированной службы EJB или службы, к которой осуществляют доступ клиенты EJB.
- Импортированная служба слабосвязанная.
- Взаимодействия EJB с сохранением состояния не требуются.

- Предпочтителен синхронный обмен данными, т.е. запрос от одной службы может ожидать ответа другой.

Привязку *Информационная система предприятия (EIS)* рекомендуется выбирать при соблюдении следующих условий:

- Требуется обеспечить доступ к службе в системе EIS с использованием адаптера ресурса.
- Синхронный обмен данными предпочтителен по сравнению с асинхронным.

Привязку *Служба обмена сообщениями Java (JMS)* рекомендуется выбирать при соблюдении следующих условий:

Важное замечание: Существует несколько типов связываний JMS. Если обмен сообщениями SOAP будет осуществляться с помощью JMS, воспользуйтесь привязкой веб-служб с протоколом SOAP/JMS. См. раздел “Привязки веб-служб” на стр. 74.

- Требуется обеспечить доступ к системе обмена сообщениями.
- Используются слабосвязанные службы.
- Асинхронный обмен данными предпочтителен по сравнению с синхронным.

Привязку *Базовая служба обмена сообщениями Java (JMS)* рекомендуется выбирать при соблюдении следующих условий:

- Требуется обеспечить доступ к системе обмена сообщениями, выпущенной не компанией IBM.
- Используются слабосвязанные службы.
- Надежность важнее, чем быстродействие: асинхронный обмен данными подходит лучше, чем синхронный.

Привязку *Очередь сообщений (MQ)* рекомендуется выбирать при соблюдении следующих условий:

- Требуется обеспечить доступ к системе обмена сообщениями WebSphere MQ, а также использование внутренних функций MQ.
- Используются слабосвязанные службы.
- Надежность важнее, чем быстродействие: асинхронный обмен данными подходит лучше, чем синхронный.

Связывание *MQ JMS* рекомендуется использовать, если выполняются следующие условия:

- Требуется обеспечить доступ к системе обмена сообщениями WebSphere MQ в контексте JMS; то есть набор функций JMS является достаточным для приложения.
- Используются слабосвязанные службы.
- Надежность важнее, чем быстродействие: асинхронный обмен данными подходит лучше, чем синхронный.

Привязки SCA

Привязка архитектуры компонентов служб (SCA) позволяет службе взаимодействовать с другими службами в других модулях. Импорт с привязкой SCA позволяет обращаться к службе в другом модуле SCA. Экспорт с привязкой SCA обеспечивает доступ других модулей к данной службе.

IBM Integration Designer применяется для создания и настройки привязок SCA для импортов и экспортов в модулях SCA.

Если модули выполняются на одном и том же сервере или развернуты в одном и том же кластере, то привязка SCA является простейшей и скорейшей для использования.

После развертывания модуля с привязкой SCA на сервере административную консоль можно использовать для просмотра информации о привязке или, в случае привязки импорта, для изменения выбранных свойств привязки.

Привязки веб-служб

Привязки веб-служб - это способ передачи сообщений из компонента архитектуры компонентов служб (SCA) в веб-службу и обратно.

Обзор привязок веб-служб: Advanced

Привязка импорта веб-службы позволяет вызывать внешнюю веб-службу из компонентов архитектуры компонентов служб (SCA). Привязка экспорта веб-службы позволяет делать компоненты SCA доступными клиентам в виде веб-служб.

С помощью привязки веб-службы можно обращаться к внешним службам посредством согласованных сообщений SOAP и QoS.

Integration Designer применяется для создания и настройки привязок веб-служб для импортов и экспортов в модулях SCA. Доступны следующие типы привязок веб-служб:

- SOAP1.2/HTTP и SOAP1.1/HTTP

  Эти типы привязок основаны на Java API for XML Web Services (JAX-WS), программном API на Java для создания веб-служб.

  – Используйте SOAP1.2/HTTP, если веб-служба соответствует спецификации SOAP 1.2.
  – Используйте SOAP1.1/HTTP, если веб-служба соответствует спецификации SOAP 1.1.

  Важное замечание: При развертывании приложения со связыванием веб-службы (JAX-WS) на целевом сервере необходимо выключить опцию **Запускать компоненты по необходимости**. Дополнительная информация приведена в разделе “Проверка конфигурации сервера” на стр. 82.

  Выбрав один из типов связывания, можно отправлять сообщения SOAP с вложениями.

  Связывание веб-службы работает со стандартными сообщениями SOAP. Тем не менее, различные типы связывания веб-службы (JAX-WS) позволяют настроить способ обработки или составления сообщений SOAP. Например, можно обрабатывать нестандартные элементы в сообщениях SOAP или применять дополнительную обработку для сообщения SOAP. В ходе настройки связывания указывается пользовательский обработчик данных, который выполняет такую обработку сообщений SOAP.

  В связывании веб-службы (JAX-WS) можно использовать наборы стратегий. Набор стратегий является набором типов стратегий, каждый из которых обеспечивает качество обслуживания (QoS). Например, набор стратегий WSAddressing обеспечивает независимую от транспортного протокола адресацию веб-служб и сообщений. Для выбора набора стратегий для связывания можно использовать Integration Designer.

  Примечание: Для использования набора стратегий на языке Security Assertion Markup Language (SAML) необходимо выполнить дополнительные действия по настройке, описанные в разделе “Импорт наборов стратегий SAML” на стр. 80.

- SOAP1.1/HTTP

  Эта привязка позволяет создавать веб-службы с сообщением в формате SOAP с использованием Java API for XML-based RPC (JAX-RPC).

- SOAP1.1/JMS

  Эта привязка позволяет отправлять и принимать сообщения SOAP с помощью адресатов службы сообщений Java (JMS).

Независимо от того, какой протокол (HTTP или JMS) используется для передачи сообщения SOAP, связывание веб-службы всегда обрабатывает взаимодействия запрос/ответ синхронно. Вызывающая поставщика службы нить блокируется до получения ответа от поставщика. Дополнительная информация приведена в разделе “Синхронный вызов”.

Важное замечание: Следующие сочетания связывания веб-служб не могут использоваться в экспортах в одном и том же модуле. Если требуется экспортировать компоненты с помощью более чем одной из этих привязок экспорта, их необходимо поместить в разные модули и затем связать эти модули с компонентами посредством связывания SCA:

- SOAP 1.1/JMS и SOAP 1.1/HTTP с использованием JAX-RPC

- SOAP 1.1/HTTP с использованием JAX-RPC и SOAP 1.1/HTTP с использованием JAX-WS
- SOAP 1.1/HTTP с использованием JAX-RPC и SOAP 1.2/HTTP с использованием JAX-WS

После развертывания на сервере модуля SCA, содержащего связывание веб-службы, в административной консоли можно просмотреть информацию о связывания или изменить свойства связывания.

Примечание: Веб-службы позволяют приложениям взаимодействовать посредством стандартных описаний служб и стандартных форматов обмена сообщениями. Например, привязки импорта и экспорта веб-службы могут взаимодействовать со службами, реализованными на основе web Services Enhancements (WSE) версии 3.5 и Windows Communication Foundation (WCF) версии 3.5 для Microsoft .NET. При взаимодействии с такими службами необходимо обеспечить следующее:

- Файл WSDL, используемый для доступа к экспорту веб-службы, должен включать непустое действие SOAP для каждой операции в интерфейсе.
- При отправке сообщений в экспорт веб-службы клиент веб-службы должен содержать или заголовок SOAPAction, или заголовок wsa:Action.

Распространение заголовка SOAP: Advanced

При обработке сообщений SOAP может потребоваться получить доступ к информации каких-либо заголовков SOAP полученных сообщений, передать сообщения с заголовками SOAP, которым присвоены заданные значения, или разрешить прохождение заголовков SOAP через модуль.

При настройке связывания веб-службы в Integration Designer можно указать, что заголовки SOAP должны передаваться.

- Когда информация приходят в экспорт или ответы - в импорт, становится доступным заголовок SOAP. При этом модуль может работать со значениями заголовка и изменять заголовок.
- Когда запросы отправлены из экспорта или ответы - из импорта, в соответствующие сообщения можно включать заголовки SOAP.

Формат и наличие передаваемых заголовков SOAP могут модифицироваться наборами стратегий, настроенными в импорте или экспорте. См. раздел Табл. 31 на стр. 76.

Для того чтобы настроить передачу заголовков SOAP для импорта или экспорта, на панели Свойства Integration Designer откройте вкладку **Передача заголовка протокола** и настройте требуемые параметры.

Заголовок WS-Addressing

Заголовок WS-Addressing может передаваться связыванием веб-службы (JAX-WS).

При передаче заголовка WS-Addressing необходимо учитывать следующее:

- Если передача заголовка WS-Addressing включена, то заголовок будет передаваться в модуль в следующих ситуациях:
  - Когда запросы получены в экспорте
  - Когда ответы получены в импорте
- Заголовок WS-Addressing header не передается в исходящие сообщения из IBM Business Process Manager, то есть он не передается когда запросы отправлены из импорта или ответы - из экспорта.

Заголовок WS-Security

Заголовок WS-Security может передаваться как связыванием веб-службы (JAX-WS), так и связыванием веб-службы (JAX-RPC).

Спецификация WS-Security веб-служб описывает расширения для обмена сообщениями SOAP, отвечающие за сохранение целостности сообщений, конфиденциальность и идентификацию для отдельных сообщений. Эти механизмы можно использовать для подключения ряда моделей защиты и технологий шифрования.

При передаче заголовка WS-Security необходимо учитывать следующее:

- Если передача заголовка WS-Security включена, то заголовок будет передаваться через модуль в следующих ситуациях:
  - Когда запросы получены в экспорте
  - Когда запросы отправлены из импорта
  - Когда ответы получены в импорте
- По умолчанию заголовок *не* будет передаваться, когда ответы отправлены из экспорта. Однако если свойство JVM **WSSECURITY.ECHO.ENABLED** задано равным **true**, то заголовок будет передаваться, когда ответы отправлены из экспорта. В этом случае, если заголовок WS-Security не изменяется в пути запроса, то он может автоматически перейти из запросов в ответы.
- Точный формат сообщения SOAP, отправленного из импорта для запроса или из экспорта для ответа, может не совпадать с полученным в исходном сообщении SOAP. Потому цифровая подпись должна считаться утратившей силу. Если цифровая подпись необходима для отправляемых сообщений, то ее необходимо создать с помощью соответствующих процедур стратегий защиты, а заголовки WS-Security, связанные с цифровой подписью в полученных сообщениях, должны быть удалены в модуле.

Для передачи заголовка WS-Security необходимо включить схему WS-Security в модуль приложения. Дополнительная информация о включении схемы приведена в разделе “Включение схемы WS-Security в модуль приложения” на стр. 77.

Как передаются заголовки

Способ передачи заголовков зависит от параметров стратегии защиты, настроенных для связывания импорта или экспорта, как показано в Табл. 31 на стр. 76:

Таблица 60. Как передаются заголовки защиты

	Связывание экспорта в отсутствие стратегии защиты	**Связывание экспорта при наличии стратегии защиты**
Связывание импорта в отсутствие стратегии защиты	Заголовки защиты передаются без изменений через модуль. Они не расшифровываются. В исходящем сообщении эти заголовки точно соответствуют полученным. При этом цифровая подпись может быть нарушена.	Заголовки защиты расшифровываются и передаются через модуль с проверкой подписи и идентификацией. Расшифрованные заголовки отправляются как исходящие. При этом цифровая подпись может быть нарушена.
Связывание импорта при наличии стратегии защиты	Заголовки защиты передаются без изменений через модуль. Они не расшифровываются. Заголовки не должны передаваться с импорт. В противном случае возникает ошибка из-за дублирования.	Заголовки защиты расшифровываются и передаются через модуль с проверкой подписи и идентификацией. Заголовки не должны передаваться с импорт. В противном случае возникает ошибка из-за дублирования.

Настройте соответствующие наборы стратегий для связывания импорта и экспорта. Это изолирует клиента служб от изменений конфигурации или требований QoS поставщика служб. Если стандартные заголовки

SOAP видимы в модуле, то это влияет на операции в модуле, например, на протокол и трассировку. Передача заголовков SOAP через модуль из полученного сообщения в отправленное означает, что преимущества изоляции теряются.

Стандартные заголовки, такие как заголовки WS-Security, не следует передавать в запросе в импорт или в ответе в экспорт, когда с импортом или экспортом связан набор стратегий, отвечающий за создание таких заголовков. В противном случае возникнет ошибка из-за дублирования заголовков. Вместо этого необходимо явно удалить эти заголовки или настроить связывание импорта или экспорта так, чтобы эти заголовки протокола не передавались.

Доступ к заголовкам SOAP

Когда сообщение, содержащее заголовки SOAP, приходит от импорта или экспорта веб-службы, заголовки помещаются в раздел заголовков объекта сообщения службы (SMO). Доступ к информации заголовка описан в разделе “Доступ к информации заголовка SOAP в SMO”.

Включение схемы WS-Security в модуль приложения

Ниже описана процедура включения схемы в модуль приложения:

- Если компьютер, на котором работает Integration Designer, подключен к Интернету, то выполните следующие действия:
  1. В проекции Бизнес-интеграция выберите **Зависимости** для своего проекта.
  2. Откройте раздел **Стандартные ресурсы** и выберите или **Файлы схемы WS-Security 1.0**, или **Файлы схемы WS-Security 1.1**, чтобы импортировать схему в модуль.
  3. Очистите и заново скомпонуйте проект.
- Если компьютер, на котором работает Integration Designer, не подключен к Интернету, то схему можно загрузить на другой компьютер, подключенный к Интернету. Затем можно будет скопировать схему на компьютер, на котором работает Integration Designer.
  1. Загрузите схему на компьютер, у которого есть подключение к Интернету:
     a. Выберите **Файл > Импорт > Бизнес-интеграция > WSDL и XSD**.
     b. Выберите **Удаленный WSDL** или **Файл XSD**.
     c. Импортируйте следующие схемы:

        http://www.w3.org/2003/05/soap-envelope/

        http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd

        http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd
  2. Скопируйте схемы на компьютер, у которого отсутствует подключение к Интернету.
  3. Импортируйте схемы на компьютер, у которого отсутствует подключение к Интернету.
     a. Выберите **Файл > Импорт > Бизнес-интеграция > WSDL и XSD**.
     b. Выберите **Локальный WSDL** или **Файл XSD**.
  4. Измените расположение схемы для oasis-wss-wssecurity_secext-1.1.xsd:
     a. Откройте схему из файла *расположение-рабочей-среды*/*имя-модуля*/StandardImportFilesGen/oasis-wss-wssecurity-secext-1.1.xsd.
     b. Измените:

        ```
        <xs:import namespace='http://www.w3.org/2003/05/soap-envelope'
        schemaLocation='http://www.w3.org/2003/05/soap-envelope/'/>
        ```

        на:

        ```
        <xs:import namespace='http://www.w3.org/2003/05/soap-envelope'
        schemaLocation='../w3/_2003/_05/soap_envelope.xsd'/>
        ```

     c. Измените:

        ```
        <xs:import namespace='http://www.w3.org/2001/04/xmlenc#'
        schemaLocation='http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd'/>
        ```

на:

```
<xs:import namespace='http://www.w3.org/2001/04/xmlenc#'
schemaLocation='../w3/tr/_2002/rec_xmlenc_core_20021210/xenc-schema.xsd'/>
```

5. Измените расположение схемы для oasis-200401-wss-wssecurity-secext-1.0.xsd:
   a. Откройте схему из файла *расположение-рабочей-среды*/*имя-модуля*/StandardImportFilesGen/oasis-200401-wss-wssecurity-secext-1.0.xsd.
   b. Измените:

```
<xsd:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
```

   на:

```
<xsd:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="../w3/tr/_2002/rec_xmldsig_core_20020212/xmldsig-core-schema.xsd"/>
```

6. Очистите и заново скомпонуйте проект.

Передача заголовков транспортного протокола: Advanced

При обработке сообщений SOAP может потребоваться получить доступ к информации каких-либо транспортных заголовков полученных сообщений, передать сообщения с транспортными заголовками, которым присвоены заданные значения, или разрешить прохождение транспортных заголовков через модуль.

При настройке привязки веб-службы в Integration Designer можно указать, что транспортные заголовки должны передаваться.

- Когда запросы приходят в экспорт или ответы - в импорт, становится доступным заголовок транспортного протокола. При этом модуль может работать со значениями заголовка и изменять заголовок.
- Когда ответы отправляются из экспорта или запросы - из импорта, в соответствующие сообщения можно включать заголовки транспортного протокола.

Настройка передачи заголовков

Для того чтобы настроить передачу транспортных заголовков для импорта или экспорта, выполните следующие действия:

1. На панели Свойства Integration Designer выберите **Связывание > Передача**.
2. Настройте параметры передачи транспортных заголовков.

Примечание: По умолчанию передача транспортных заголовков выключена и может применяться только в среде выполнения версии не ниже 7.0.0.3. Обратите внимание, что в версии 7.0.0.3 передача транспортных заголовков возможна только для заголовков HTTP.

Если включена передача транспортных заголовков, то заголовки будут передаваться через модуль из полученных сообщений, и если их явно не удалить, то заголовки будут использоваться в последующих вызовах в той же нити.

Примечание: Передача транспортных заголовков невозможна при использовании связывания веб-службы (JAX-RPC).

Доступ к информации заголовка

Когда для принимаемых сообщений включена передача транспортных заголовков, все транспортные заголовки, включая пользовательские, доступны в объекте сообщения службы (SMO). Значения заголовков можно изменить, также можно создать новые заголовки. Однако проверка значений не выполняется, и недопустимые или ошибочные заголовки могут привести к сбою веб-службы.

При настройке заголовков HTTP необходимо учитывать следующее:

- Все изменения заголовков, зарезервированных для веб-службы, не будут сохраняться в исходящем сообщении. Например, модуль веб-службы сам устанавливает значения заголовков HTTP version, method, Content-Type, Content-Length и SOAPAction.
- Если значение заголовка - это число, то необходимо непосредственно присвоить число (а не строку). Например, следует использовать **Max-Forwards = 5** (а не **Max-Forwards = Max-Forwards: 5**) и **Age = 300** (а не **Age = Age: 300**).
- Если размер сообщения-запроса не превышает 32 КБ, то веб-служба удаляет заголовок Transfer-Encoding и вместо него в заголовке Content-Length указывает фиксированный размер сообщения.
- WAS.channel.http сбрасывает значение Content-language в пути ответа.
- Ошибочное значение Upgrade приводит к ошибке с кодом 500.
- Следующие заголовки дописывают значение, зарезервированное модулем веб-службы, к пользовательским параметрам заголовка:
  - User-Agent
  - Cache-Control
  - Pragma
  - Accept
  - Connection

Получить данные заголовка можно одним из следующих способов:

- Используя примитив передачи для доступа к структурам SMO

  Дополнительная информация по работе с примитивами передачи приведена по ссылкам “Связанная информация”.
- Используя SPI службы контекста

  В следующем примере кода транспортные заголовки HTTP читаются из службы контекста:

```
HeadersType headerType = ContextService.INSTANCE.getHeaders();
HTTPHeaderType httpHeaderType = headerType.getHTTPHeader();
List HTTPHeader  httpHeaders = httpHeaderType.getHeader();
if(httpHeaders!=null){
 for(HTTPHeader httpHeader: httpHeaders){
   String httpHeadername = httpHeader.getName();
   String httpHeaderValue = httpHeader.getValue();
     }
}
List PropertyType  properties = headerType.getProperties();
if(properties!=null){
  for(PropertyType property: properties){
   String propertyName = property.getName();
   String propertyValue = property.getValue().toString();
     }
}
```

Устранение неполадок

При неполадках с отправкой новых заголовков можно перехватить сообщение TCP/IP такими инструментами, как TCP/IP Monitor в Integration Designer. Открыть TCP/IP Monitor можно, выбрав **Запуск/Отладка > TCP/IP Monitor** на странице Параметры.

Просмотреть значения заголовков можно также с помощью трассировки JAX-WS: **org.apache.axis2.*=all: com.ibm.ws.websvcs.*=all:**

Работа с привязками веб-служб (JAX-WS): Advanced

В привязки веб-служб (JAX-WS), применяемые в приложениях, можно добавить описание QoS на языке Security Assertion Markup Language (SAML). Для импорта набора стратегий сначала необходимо использовать административную консоль. Также административная консоль позволяет проверить, правильно ли настроен сервер для работы с привязкой веб-службы (JAX-WS).

Импорт наборов стратегий SAML: Advanced

Security Assertion Markup Language (SAML) - это основанный на XML стандарт OASIS для обмена данными из атрибутов пользовательских профайлов и защиты. Во время настройки привязки веб-службы (JAX-WS) в Integration Designer можно указать набор стратегий SAML. Сначала с помощью административной консоли IBM Business Process Manager открывается доступ к наборам стратегий SAML для того чтобы их можно было импортировать в Integration Designer.

Наборы стратегий SAML обычно располагаются в каталоге конфигурации профайла:

каталог-профайла/config/templates/PolicySets

Перед началом этой процедуры убедитесь в наличии следующих каталогов (которые содержат наборы стратегий) в каталоге конфигурации профайла:

- SAML11 Bearer WSHTTPS default
- SAML20 Bearer WSHTTPS default
- SAML11 Bearer WSSecurity default
- SAML20 Bearer WSSecurity default
- SAML11 HoK Public WSSecurity default
- SAML20 HoK Public WSSecurity default
- SAML11 HoK Symmetric WSSecurity default
- SAML20 HoK Symmetric WSSecurity default
- Имя-пользователя WSHTTPS default

Если каталоги не находятся в каталоге конфигурации профайла, скопируйте их в этот каталог из следующего расположения:

корневой-каталог-сервера-приложений/profileTemplates/default/documents/config/templates/PolicySets

Импортируйте наборы стратегий в административную консоль, выберите из них те, к которым требуется открыть доступ для Integration Designer, а затем сохраните файл .zip для каждого из этих наборов стратегий в расположении, доступном для Integration Designer.

1. Импортируйте наборы стратегий, выполнив следующие действия:
   a. В административной консоли выберите **Службы > Наборы стратегий > Наборы стратегий приложений**.
   b. Выберите действия **Импорт > Из хранилища по умолчанию**.
   c. Выберите наборы стратегий SAML по умолчанию и нажмите кнопку **OK**.
2. Экспортируйте наборы стратегий для того чтобы приложение Integration Designer смогло их использовать:
   a. На странице Наборы стратегий приложения выберите набор стратегий, который требуется импортировать, и нажмите кнопку **Экспортировать**.

      Примечание: Если страница Наборы стратегий приложений на данный момент не открыта, выберите **Службы > Наборы стратегий > Наборы стратегий приложений** в административной консоли.

b. На следующей странице щелкните на ссылке на файл .zip набора стратегий.
c. В окне Загрузка файла нажмите кнопку **Сохранить** и укажите расположение, доступное для Integration Designer.
d. Нажмите кнопку **Назад**.
e. Выполните действия с 2a на стр. 80 по 2d на стр. 81 для каждого набора стратегий, который требуется экспортировать.

Наборы стратегий SAML сохраняются в файлах .zip и готовы к импорту в Integration Designer.

Импортируйте наборы стратегий в Integration Designer, как описано в разделе “Наборы стратегий”.

Вызов веб-служб, которые требуют простой идентификации HTTP: Advanced

Простая идентификация HTTP использует имя пользователя и пароль для идентификации клиента службы на защищенной точке. Можно настроить простую идентификацию HTTP для отправки и получения запросов веб-служб.

Настройка простой идентификации HTTP для получения запросов веб-служб производится путем настройки привязки экспорта API Java для веб-служб XML (JAX-WS), как описано в разделе Создание и назначение ролей защиты спискам экспорта веб-служб.

Можно включить простую идентификацию HTTP для запросов веб-служб, отправляемых привязкой импорта JAX-WS, одним из двух способов:

- при настройке привязки импорта в модуле SCA можно выбрать существующий набор стратегий идентификации HTTP с именем BPMHTTPBasicAuthentication (который поставляется с привязкой импорта веб-службы (JAX-WS)) или любой другой набор стратегий, содержащий стратегию HTTPTransport;
- при конструировании модуля SCA можно использовать возможности потока передачи для динамического создания нового заголовка идентификации HTTP и указать в заголовке имя пользователя и пароль.

Примечание: Набор стратегий имеет приоритет перед значением, указанным в заголовке. Если требуется использовать значение из заголовка идентификации HTTP во время выполнения, не прикрепляйте набор стратегий, содержащий стратегию HTTPTransport. В особенности, не используйте стандартный набор стратегий BPMHTTPBasicAuthentication и, если набор стратегий определен, убедитесь, что в нем отсутствует стратегия HTTPTransport.

Дополнительные сведения о наборах стратегий веб-служб и привязках стратегий, а также об их использовании приведены в разделе Наборы стратегий веб-служб в справочной системе WebSphere Application Server Information Center.

- Для использования поставляемого набора стратегий выполните следующие действия:
  1. Необязательно: В административной консоли создайте общую привязку стратегии клиента или измените существующую привязку, которая содержит стратегию HTTPTransport с требуемым ИД пользователя и паролем.
  2. В IBM Integration Designer создайте привязку импорта веб-службы (JAX-WS) и прикрепите набор стратегий BPMHTTPBasicAuthentication.
  3. Выполните *одно* из следующих действий:
     – В IBM Integration Designer в свойствах привязки импорта веб-службы (JAX-WS) укажите имя существующей общей привязки стратегии клиента, которая содержит стратегию HTTPTransport.
     – После развертывания модуля SCA в административной консоли выберите существующую привязку стратегии клиента или создайте привязку и свяжите ее с привязкой импорта.
  4. Необязательно: В административной консоли сервера процессов измените выбранную привязку набора стратегий, указав требуемые ИД пользователя и пароль.

- Для того чтобы указать имя пользователя и пароль в заголовке идентификации HTTP выполните следующие действия:
  - с помощью примитива передачи Задать заголовок HTTP в IBM Integration Designer создайте заголовок идентификации HTTP и укажите имя пользователя и пароль;
  - если требуется дополнительная логика, используйте код Java в пользовательском примитиве передачи (как показано в следующем примере) для выполнения следующих действий:
    1. создания заголовка идентификации HTTP;
    2. ввода имени пользователя и пароля;
    3. добавления нового заголовка идентификации HTTP в HTTPControl;
    4. возврата обновленного HTTPControl в службу Контекст.

```
//Получить HeaderInfoType из contextService
ContextService contextService = (ContextService) ServiceManager.INSTANCE
.locateService("com/ibm/bpm/context/ContextService");
HeaderInfoType headers = contextService.getHeaderInfo();
if(headers == null){
    headers = ContextObjectFactory.eINSTANCE.createHeaderInfoType();
}
//Получить заголовок HTTP и элемент управления HTTP из HeaderInfoType
HTTPHeaderType httpHeaderType = headers.getHTTPHeader();
HTTPControl cp = httpHeaderType.getControl();
HeadersFactory factory = HeadersFactory.eINSTANCE;
if(cp == null){
    cp = factory.createHTTPControl();
}
//Создать HTTPAuthentication и задать HTTPCredentials
HTTPAuthentication authorization = factory.createHTTPAuthentication();
HTTPCredentials credentials = factory.createHTTPCredentials();
authorization.setAuthenticationType(HTTPAuthenticationType.BASIC_LITERAL);
credentials.setUserId("ИМЯ");
credentials.setPassword("ПАРОЛЬ");
authorization.setCredentials(credentials);
cp.setAuthentication(authorization);
httpHeaderType.setControl(cp);
//Возвратить информацию заголовка в текущий контекст выполнения.
contextService.setHeaderInfo(headers);
```

Проверка конфигурации сервера: Advanced

Во время развертывания приложения с привязкой веб-службы (JAX-WS) необходимо убедиться, что в конфигурации сервера, на котором развертывается приложение, не выбрана опция **Запускать компоненты по необходимости**.

Проверить, выбрана ли эта опция, можно путем выполнения следующих действий в административной консоли:

1. Выберите пункт **Серверы > Типы серверов > Серверы приложений WebSphere**.
2. Щелкните на имени сервера.
3. На вкладке Конфигурация определите, выбран ли параметр **Запускать компоненты по необходимости**.
4. Выполните одно из следующих действий:
   - Если параметр **Запускать компоненты по необходимости** выбран, отмените выбор и нажмите кнопку **Применить**.
   - Если параметр **Запускать компоненты по необходимости** на выбран, нажмите кнопку **Отмена**.

Вложения в сообщениях SOAP: Advanced

Можно отправлять и получать сообщения SOAP, включающие двоичные данные (такие как файлы PDF или изображения JPEG) в виде вложений. Вложения могут быть *ссылочными*, то есть явным образом

представленными как фрагменты сообщения в интерфейсе службы, или *нессылочными*, в которые может быть включено произвольное число вложений различных типов.

Ссылочное вложение представляется одним из следующих способов:

- Вложения MTOM используют кодировку, заданную механизмом MTOM протокола SOAP (http://www.w3.org/TR/soap12-mtom/). Для включения поддержки вложений MTOM необходимо настроить параметр привязок импорта и экспорта. Это рекомендуемый способ кодировки вложений в новых приложениях.
- Как элемент с типом wsi:swaRef в схеме сообщения

  Вложения, определенные с типом wsi:swaRef, соответствуют спецификации Web Services Interoperability Organization (WS-I) *Attachments Profile Version 1.0* (http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html), где описано, как связаны элементы сообщения и фрагменты MIME.
- Как главный фрагмент сообщения с использованием двоичного типа схемы

  Вложения, являющиеся главным фрагментом сообщения, соответствуют спецификации *сообщений SOAP с вложениями* (http://www.w3.org/TR/SOAP-attachments).

  Вложения, представленные как фрагмент сообщения верхнего уровня, можно настроить так, чтобы документ WSDL и сообщения, генерируемые привязкой, соответствовали спецификации WS-I *Attachments Profile Version 1.0* и WS-I *Basic Profile Version 1.1* (http://www.ws-i.org/Profiles/BasicProfile-1.1.html).

Нессылочное вложение в сообщении SOAP никак не представлено в схеме сообщения.

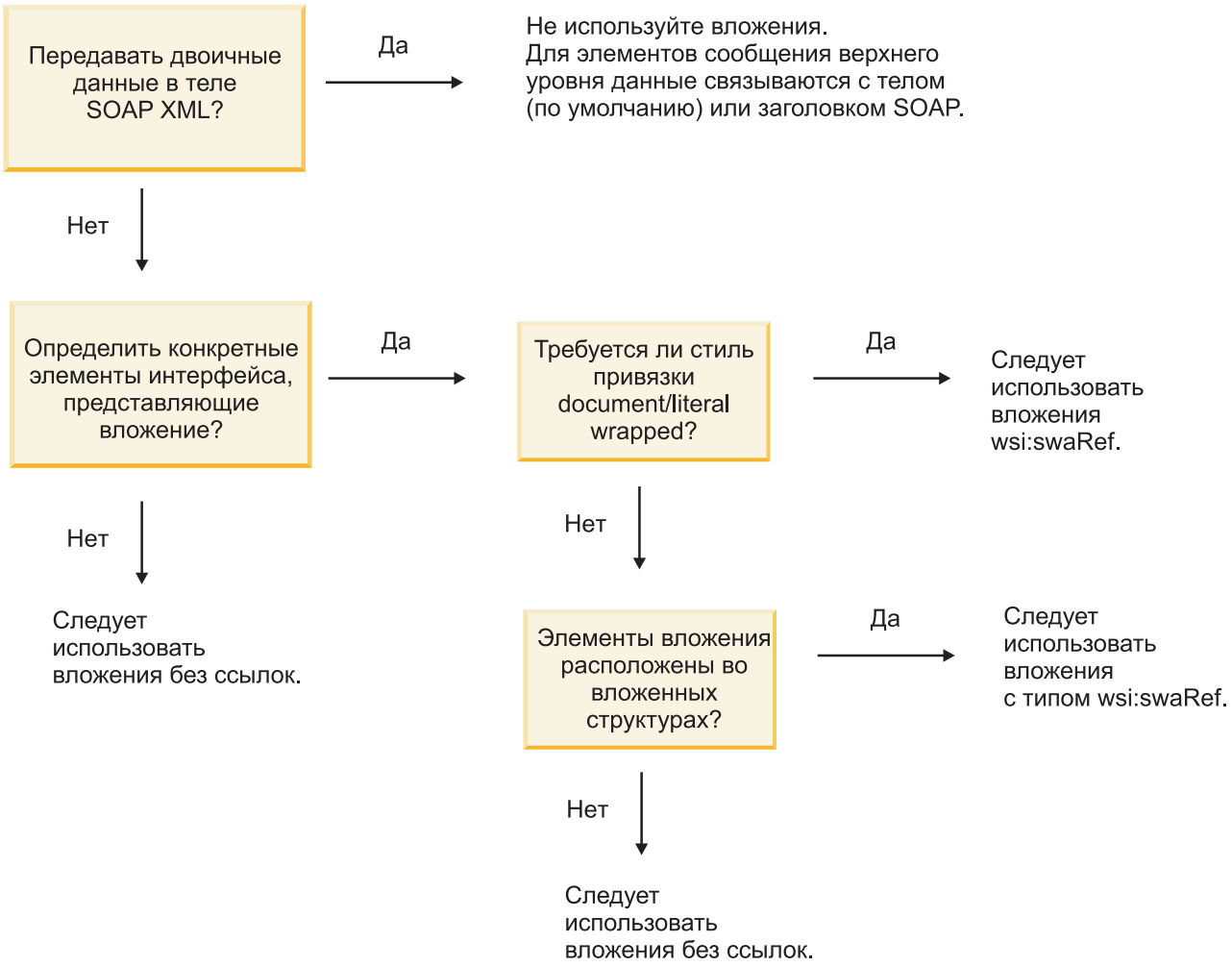
Во всех случаях, кроме вложений MTOM, привязка WSDL SOAP должна включать привязку MIME для вложений, а максимальный размер вложений не должен превышать 20 МБ..

Примечание: Для того чтобы отправлять сообщения SOAP с вложениями или получать их, необходимо использовать один из вариантов привязки веб-служб, основанный на Java API for XML Web Services (JAX-WS).

Способы выбора стиля соответствующего вложения: Advanced

При проектировании нового интерфейса службы, включающего двоичные данные, необходимо спланировать, как двоичные данные будут передаваться в сообщениях SOAP, отправляемых и получаемых службой.

Для вложений должен использоваться механизм MTOM, если он поддерживается подключенным приложением веб-службы. В противном случае используйте следующую диаграмму для выбора стиля вложения. Используйте следующий перечень вопросов, чтобы определить необходимый тип вложения:



Вложения MTOM: фрагменты сообщений верхнего уровня:

Поддерживается отправка и прием сообщений веб-служб с вложениями MTOM. В сообщении SOAP с несколькими фрагментами MIME тело SOAP является первым фрагментом сообщения, а вложение или вложения передаются в последующих фрагментах.

При отправке или приеме ссылочного вложения в сообщении SOAP двоичные данные вложения (часто большого размера) хранятся отдельно от тела сообщения SOAP, поэтому их не требуется обрабатывать как данные XML. При этом производительность обработки по сравнению с обработкой данных в элементах XML повышается.

Ниже приведен пример сообщения MTOM SOAP:

```
... другие транспортные заголовки ...
Content-Type: multipart/related; boundary=MIMEBoundaryurn_uuid_0FE43E4D025F0BF3DC11582467646812; type="application/xop+xm
<0.urn:uuid:0FE43E4D025F0BF3DC11582467646813@apache.org>"; start-info="text/xml"; charset=UTF-8

--MIMEBoundaryurn_uuid_0FE43E4D025F0BF3DC11582467646812
content-type: application/xop+xml; charset=UTF-8; type="text/xml";
content-transfer-encoding: binary
content-id:
   <0.urn:uuid:0FE43E4D025F0BF3DC11582467646813@apache.org>

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
      <soapenv:Header/>
```

```
      <soapenv:Body>
         <sendImage xmlns="http://org/apache/axis2/jaxws/sample/mtom">
            <input>
            <imageData><xop:Include xmlns:xop="http://www.w3.org/2004/08/xop/include"
href="cid:1.urn:uuid:0FE43E4D025F0BF3DC11582467646811@apache.org"/></imageData>
            </input>
         </sendImage>
      </soapenv:Body>
   </soapenv:Envelope>
--MIMEBoundaryurn_uuid_0FE43E4D025F0BF3DC11582467646812
content-type: text/plain
content-transfer-encoding: binary
content-id:
         <1.urn:uuid:0FE43E4D025F0BF3DC11582467646811@apache.org>

... двоичные данные ...
--MIMEBoundaryurn_uuid_0FE43E4D025F0BF3DC11582467646812--
```

Обратите внимание, что в примере MTOM параметр content-type в конверте SOAP равен **application/xop+xml**, а двоичные данные заменены элементом **xop:Include** следующего вида:

```
<xop:Include xmlns:xop="http://www.w3.org/2004/08/xop/include"
href="cid:1.urn:uuid:0FE43E4D025F0BF3DC11582467646811@apache.org"/>
```

Обработка входящих ссылочных вложений

Когда клиент передает сообщение SOAP с вложением в компонент SCA, сначала привязка экспорта веб-службы (JAX-WS) удаляет вложение. Затем она обрабатывает часть SOAP сообщения и создает бизнес-объект. Наконец, привязка создает двоичное вложение в бизнес-объекте.

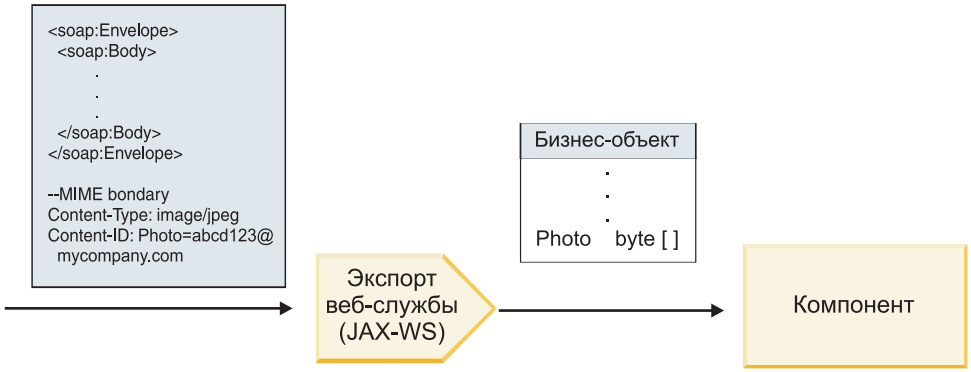


Рисунок 58. Каким образом привязка экспорта веб-службы (JAX-WS) обрабатывает сообщение SOAP со ссылочным вложением

Атрибуты вложения MTOM

- MTOM поддерживает элементы вложений во встроенных структурах.
- MTOM доступно только для типа base64Binary.
- MTOM поддерживает элементы вложений во встроенных структурах, то есть **bodyPath** вложений MTOM задает расположение **xpath** элемента, в котором хранится вложение MTOM. Логика вычисления **bodyPath** в точности следует схеме формирования расположения **xpath**, как показано в приведенных ниже примерах:
  - Для типа, отличного от массива (**maxOccurs** равно 1): /sendImage/input/imageData
  - Для массива (**maxOccurs** > 1): /sendImage/input/imageData[1]
- Смешанные типы вложений не поддерживаются, то есть если поддержка MTOM включена в привязке импорта, то вложение MTOM будет создано. Если поддержка MTOM выключена или в параметре настройки MTOM оставлено значение по умолчанию из привязки экспорта, то входящее сообщение MTOM не поддерживается.

Ссылочные вложения: элементы типа swaRef: Advanced

Можно отправлять и получать сообщения SOAP, включающие вложения, которые объявлены в интерфейсе службы как элементы типа swaRef.

Элемент типа swaRef определен в спецификации Web Services Interoperability Organization (WS-I) *Attachments Profile* Version 1.0 (http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html) и указывает, как элементы сообщений связаны с фрагментами MIME.

Тело сообщения SOAP содержит элемент типа swaRef, который задает ИД содержимого вложения.

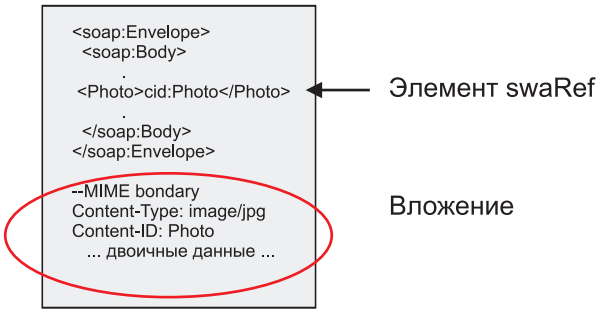


Рисунок 59. Сообщение SOAP с элементом swaRef

WSDL для этого сообщения SOAP содержит элемент с типом swaRef внутри фрагмента сообщения, определяющий вложение.

```
<element name="sendPhoto">
  <complexType>
     <sequence>
       <element name="Photo" type="wsi:swaRef"/>
     </sequence>
  </complexType>
 </element>
```

WSDL также должен включать связывание MIME, которое указывает, что используются мультифрагментные сообщения MIME.

Примечание: WSDL *не* включает связывание MIME для отдельных элементов с типом swaRef, поскольку связывание MIME применяется только к фрагменту сообщения верхнего уровня.

Вложения, представленные как элементы с типом swaRef, могут передаваться только по компонентам потока передачи. Если вложение должно быть доступно компоненту другого тип, или если его требуется передать в другой тип компонента, то используйте компонент потока передачи для перемещения вложения в расположение, доступное такому компоненту.

Обработка входящих вложений

Для настройки связывания экспорта можно использовать Integration Designer для приема сообщения. Сначала создается модуль, его интерфейс и операции, включая элемент типа swaRef. Затем создается связывание веб-службы (JAX-WS).

Примечание: Дополнительная информация приведена в разделе “Работа с вложениями” приведена в разделе Integration Designer Information Center.

Когда клиент передает сообщение SOAP с вложением типа swaRef в компонент SCA, сначала привязка экспорта веб-службы (JAX-WS) удаляет вложение. Затем оно обрабатывает часть SOAP сообщения и создает

бизнес-объект. Наконец, связывание присваивает значение content-ID вложения в бизнес-объекте.

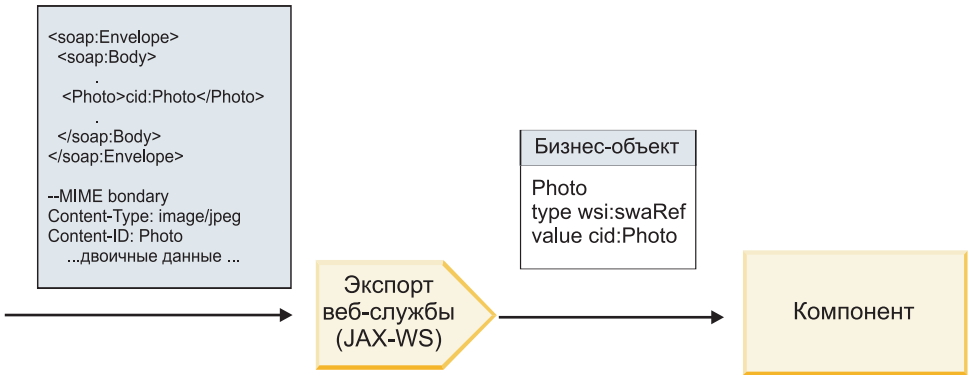


Рисунок 60. Каким образом привязка экспорта веб-службы (JAX-WS) обрабатывает сообщение SOAP с вложением типа swaRef

Доступ к мета-данным вложения в компоненте потока передачи

Как показано на рисунке рис. 16 на стр. 87, когда компоненты обращаются к вложениям типа swaRef, идентификатор содержимого вложения представляется как элемент типа swaRef.

Каждое вложение в сообщении SOAP имеет соответствующий элемент **attachments** в SMO. При использовании типа swaRef WS-I элемент **attachments** включает тип содержимого и content-ID вложения, а также сами двоичные данные вложения.

Поэтому для того чтобы получить значение вложения swaRef, необходимо получить значение элемента с типом swaRef и затем найти элемент **attachments** с соответствующим значением **contentID**. Обратите внимание, что в значении **contentID** префикс **cid:**, как право, удаляется из значения swaRef.

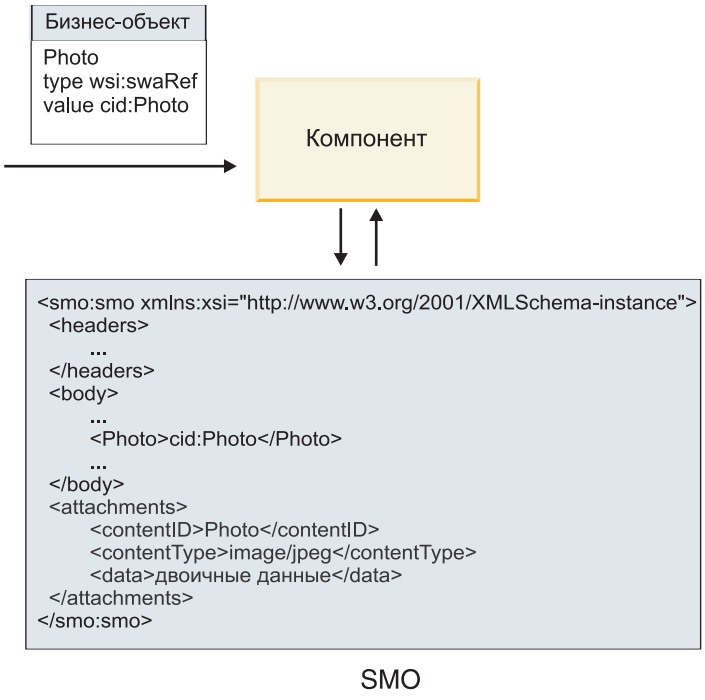


Рисунок 61. Как вложения типа swaRef представлены в SMO

Обработка исходящих

Integration Designer позволяет настроить привязку импорта веб-службы (JAX-WS) для вызова внешней веб-службы. Привязка импорта настраивается с документом WSDL, который описывает вызываемую веб-службу и определяет, какое вложение будет передаваться в веб-службу.

Когда сообщение SCA принимается привязкой веб-службы (JAX-WS), элементы с типом swaRef отправляются как вложения, если импорт связан с компонентом потока передачи и элемент типа swaRef имеет соответствующий элемент **attachments**.

В исходящих операциях элемент с типом swaRef всегда отправляются с их значениями content-ID, однако модуль передачи должен обеспечить наличие соответствующего элемента **attachments** с совпадающим значением **contentID**.

Примечание: Для соответствия спецификации WS-I Attachments Profile значение **content ID** должно следовать правилам для "content-id part encoding", описанным в разделе 3.8 спецификации WS-I *Attachments Profile* 1.0.

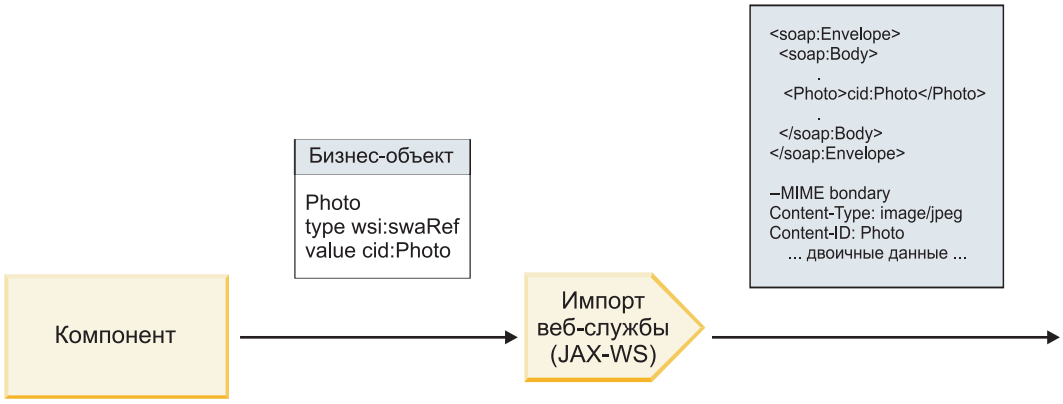


Рисунок 62. Каким образом привязка импорта веб-службы (JAX-WS) создает сообщение SOAP с вложением типа swaRef

Настройка мета-данных вложения в компоненте потока передачи

Если в SMO есть элемент с типом swaRef и элемент **attachments**, то связывание создает сообщение SOAP с вложением и отправляет его получателю.

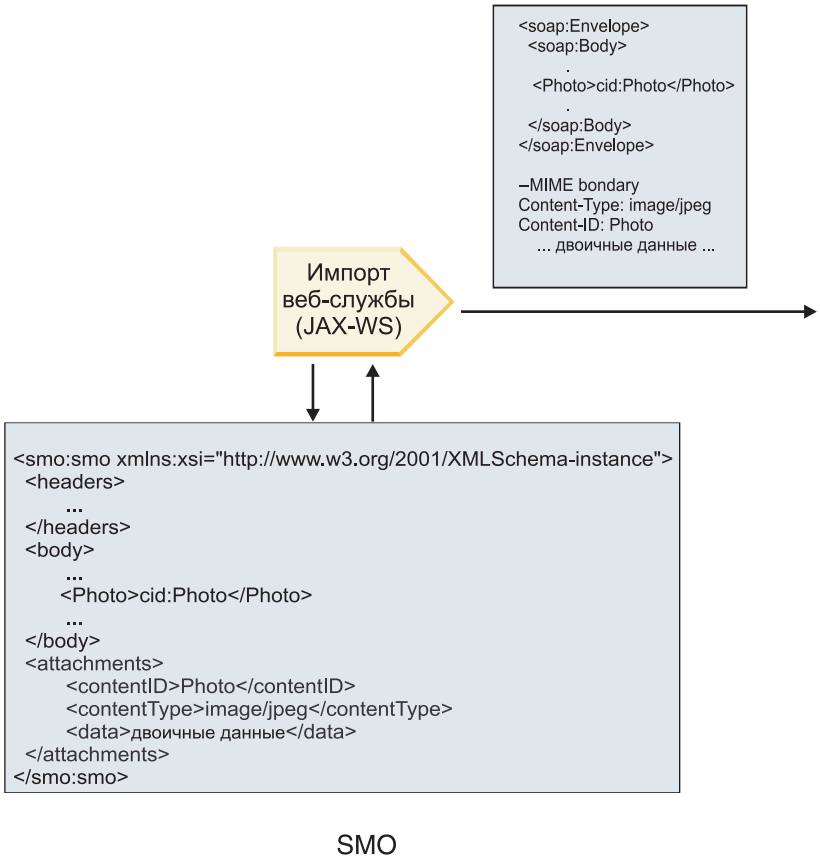


Рисунок 63. Как осуществляется доступ к вложению типа swaRef в SMO для создания сообщения SOAP

Элемент **attachments** присутствует в SMO только тогда, когда компонент потока передачи прямо соединен с импортом или экспортом. Он не передается через другие типы компонентов. Если такие значения требуется передать в модуль, содержащий другие типы компонентов, то необходимо использовать компонент потока передачи для копирования значений в расположение, где они будут доступны модулю, затем использовать другой компонент потока передачи для настройки нужных значений перед вызовом исходящей операции посредством импорта веб-службы.

Важное замечание: Как описано в разделе “Представление SMO в формате XML” примитив передачи Mapping преобразует сообщения посредством преобразования XSLT 1.0. Это преобразование работает с сериализованным SMO в формате XML. Примитив передачи Mapping позволяет задавать корневой элемент сериализации, который соответствует корневому элементу документа XML.

При отправке сообщений SOAP с вложениями выбранный корневой элемент определяет, как передаются сообщения.

- Если корневой элемент в карте XML - это “/body”, то по умолчанию все вложения передаются по карте.
- Если корневой элемент в карте - это “/”, то передачей вложений можно управлять.

Ссылочные вложения: фрагменты сообщений верхнего уровня: Advanced

Можно отправлять и получать сообщения SOAP, включающие двоичные вложения, которые объявлены как фрагменты в интерфейсе службы.

В сообщении SOAP с несколькими фрагментами MIME тело SOAP является первым фрагментом сообщения, а вложение или вложения передаются в последующих фрагментах.

В чем преимущество отправки или получения ссылочного вложения в сообщении SOAP? Двоичные данные вложения (часто большого размера) хранятся отдельно от тела сообщения SOAP, поэтому их не требуется обрабатывать как данные XML. При этом производительность обработки по сравнению с обработкой данных в элементах XML повышается.

Типы сообщений SOAP с ссылочными вложениями

Начиная с IBM Business Process Manager версии 7.0.0.3 пользователь может выбрать способ, каким генерируется сообщение SOAP:

- **Сообщения, соответствующие спецификации WS-I**

  Среда выполнения может создавать сообщения SOAP, которые соответствуют спецификации WS-I *Attachments Profile Version 1.0* и WS-I *Basic Profile Version 1.1*. В сообщении SOAP, совместимом с этими профилями, только один фрагмент сообщения связан с телом SOAP. Если фрагмент сообщения связан как вложение, то элемент content-id (как описано в спецификации WS-I *Attachments Profile Version 1.0*) используется для сопоставления вложения с фрагментом сообщения.
- **Сообщения, не соответствующие спецификации WS-I**

  Среда выполнения может создавать сообщения SOAP, которые не соответствуют спецификации профилей WS-I, но совместимы с сообщениями, созданными в версиях 7.0 или 7.0.0.2 продукта IBM Business Process Manager. Сообщения SOAP используют элементы верхнего уровня с именем, совпадающим с именем фрагмента сообщения с атрибутом **href**, содержащим **content-id** вложения, но кодировка фрагмента content-id не используется (как описано в спецификации WS-I *Attachments Profile Version 1.0*).

Настройка совместимости экспорта веб-службы с WS-I

Для настройки связывания экспорта можно использовать Integration Designer. Сначала создается модуль, его интерфейс и операции. Затем создается связывание веб-службы (JAX-WS). На странице Ссылочные вложения показаны все двоичные фрагменты из созданной операции, из которых можно выбрать вложения. Затем на странице Настроить совместимость с WS-I AP 1.0 Integration Designer выберите один из вариантов:

- **Использовать сообщение SOAP, соответствующее спецификации WS-I AP 1.0**

  Если выбрана эта опция, то также можно указать фрагмент сообщения, связанный с телом SOAP.

  Примечание: Эту опцию можно использовать, только если соответствующий файл WSDL также совместим с WS-I.

  Файл WSDL, создаваемый в Integration Designer 7.0.0.3, совместим с WS-I. Однако при импорте файла WSDL, несовместимого с WS-I, эта опция недоступна.
- **Использовать сообщение SOAP, не соответствующее спецификации WS-I AP 1.0**

  Если выбрана эта опция (значение по умолчанию), то первый фрагмент сообщения связан с телом SOAP.

Примечание: Только фрагменты сообщения верхнего уровня, то есть элементы, определенные в portType WSDL как части входящего или исходящего сообщения, имеющие двоичный тип (base64Binary или hexBinary), могут отправляться или приниматься как ссылочные вложения.
Дополнительная информация приведена в разделе “Работа с вложениями” в Integration Designer Information Center.

Для сообщений, совместимых с WS-I, элемент content-ID, генерируемый для сообщения SOAP, составляется из следующих элементов:

- Значение атрибута **name** элемента **wsdl:part**, на который ссылается **mime:content**
- Символ **=**
- Глобально уникальное значение, такое как UUID
- Символ **@**
- Допустимое имя домена

Обработка входящих ссылочных вложений

Когда клиент передает сообщение SOAP с вложением в компонент SCA, сначала связывание экспорта веб-службы (JAX-WS) удаляет вложение. Затем оно обрабатывает часть SOAP сообщения и создает бизнес-объект. Наконец, связывание создает двоичное вложение в бизнес-объекте.

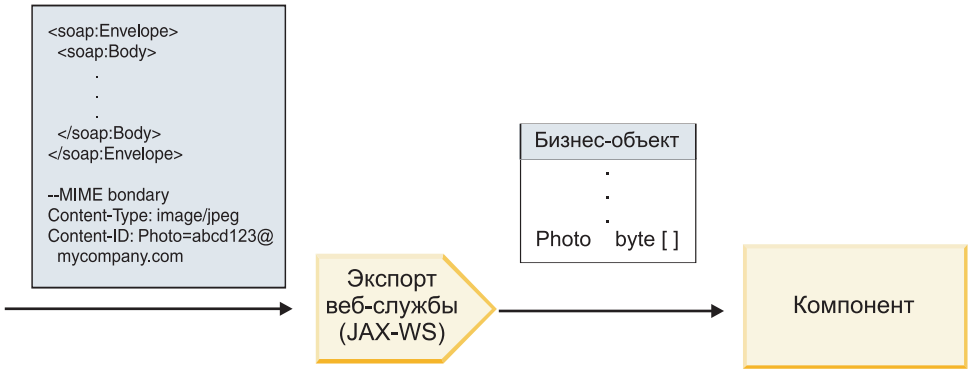


Рисунок 64. Как связывание экспорта веб-службы (JAX-WS) обрабатывает сообщение SOAP, совместимое с WS-I, с ссылочным вложением

Доступ к мета-данным вложения в компоненте потока передачи

Как показано на рисунке рис. 19 на стр. 91, когда компоненты обращаются к ссылочным вложениям, данные вложения представляются как байтовый массив.

Каждое ссылочное вложение в сообщении SOAP имеет соответствующий элемент **attachments** в SMO. Элемент **attachments** включает тип содержимого вложения и путь к элементу тела сообщения, где хранится вложение.

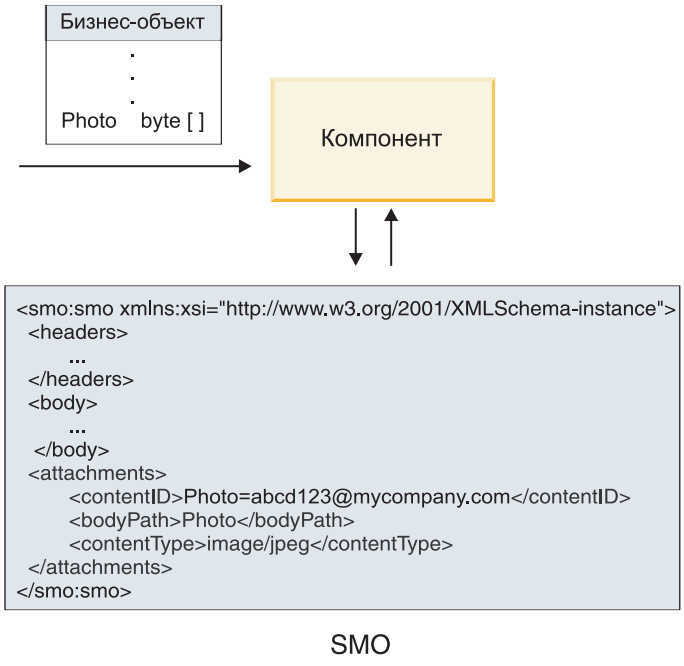


Рисунок 65. Как ссылочные вложения представлены в SMO

Важное замечание: Путь к телу сообщения не обновляется автоматически, если сообщение преобразуется и вложение перемещается. Для обновления пути в элементе **attachments**, например, в ходе преобразования или с помощью отдельного объекта составления сообщения, можно использовать поток передачи.

Как составляются исходящие сообщения SOAP

Integration Designer позволяет настроить связывание импорта веб-службы (JAX-WS) для вызова внешней веб-службы. Связывание импорта настраивается с документом WSDL, который описывает вызываемую веб-службу и определяет, какие фрагменты сообщения будут передаваться как вложения. Затем на странице Настроить совместимость с WS-I AP 1.0 Integration Designer можно выбрать один из вариантов:

- **Использовать сообщение SOAP, соответствующее спецификации WS-I AP 1.0**

  Если выбрана эта опция, то также можно указать фрагмент сообщения, связанный с телом SOAP. Все прочие фрагменты связываются с вложениями или заголовками. Сообщения, отправляемые связыванием, не включают в тело SOAP элементы, ссылающиеся на вложения. Это отношение выражается посредством content-ID вложения, включающего имя фрагмента сообщения.
- **Использовать сообщение SOAP, не соответствующее спецификации WS-I AP 1.0**

  Если выбрана эта опция (значение по умолчанию), то первый фрагмент сообщения связан с телом SOAP. Все прочие фрагменты связываются с вложениями или заголовками. Сообщения, отправляемые связыванием, включают в тело SOAP один или несколько элементов, ссылающихся на вложения посредством атрибута **href**.

Примечание: Компонент, представляющий вложение, согласно определению WSDL, должен иметь простой тип (base64Binary или hexBinary). Компонент составного типа не может быть связан как вложение.

Обработка исходящих ссылочных вложений

Привязка импорта использует информацию SMO для определения способа отправки двоичных фрагментов сообщения верхнего уровня в виде вложений.

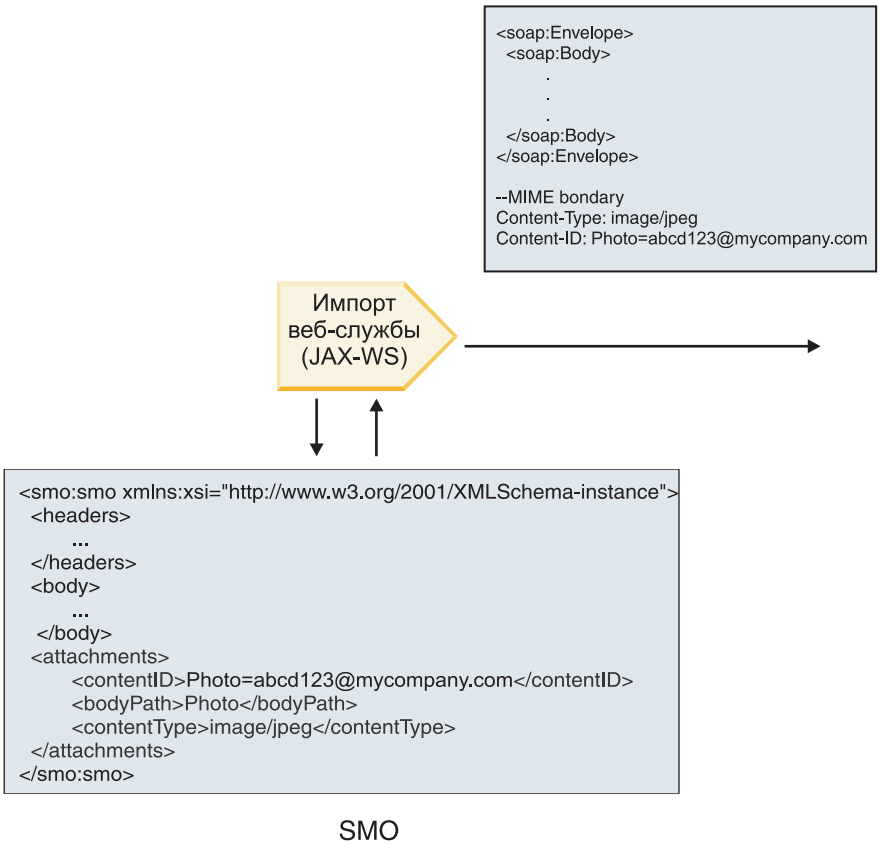


Рисунок 66. Как осуществляется доступ к ссылочному вложению в SMO для создания сообщения SOAP

Элемент **attachments** присутствует в SMO только тогда, когда компонент потока передачи прямо соединен с импортом или экспортом. Он не передается через другие типы компонентов. Если такие значения требуется передать в модуль, содержащий другие типы компонентов, то необходимо использовать компонент потока передачи для копирования значений в расположение, где они будут доступны модулю, затем использовать другой компонент потока передачи для настройки нужных значений перед вызовом исходящей операции посредством импорта веб-службы.

Для определения способа отправки сообщения в связывании анализируются следующие условия:

- Есть ли связывание MIME WSDL для двоичного фрагмента сообщения верхнего уровня, и если да, то как определен тип содержимого
- Есть ли элемент **attachments** в SMO, у которого значение **bodyPath** ссылается на двоичный фрагмент верхнего уровня

Как составляются вложения, если элемент attachment существует в SMO

В следующей таблице показано, как составляется и отправляется вложение, если SMO содержит элемент **attachment** с **bodyPath**, соответствующим части с именем сообщения:

Таблица 61. Как генерируется вложение

Состояние связывания MIME WSDL для главного двоичного фрагмента сообщения	**Как составляется и отправляется сообщение**
Выполняется одно из следующих условий: • Не определен тип содержимого для фрагмента сообщения • Определено несколько типов содержимого • Определен тип содержимого с символом подстановки	Фрагмент сообщения отправляется как вложение. Для **Content-Id** используется значение в элементе attachments, если оно задано, в противном случае оно генерируется. Для **Content-Type** используется значение в элементе attachments, если оно задано, в противном случае - **application/octet-stream**.
Присутствует, фрагмент сообщения содержит одну часть без символов подстановки	Фрагмент сообщения отправляется как вложение. Для **Content-Id** используется значение в элементе attachments, если оно задано, в противном случае оно генерируется. Для **Content-Type** используется значение в элементе attachments, если оно задано, в противном случае используется тип, определенный в элементе содержимого MIME WSDL.
Отсутствует	Фрагмент сообщения отправляется как вложение. Для **Content-Id** используется значение в элементе attachments, если оно задано, в противном случае оно генерируется. Для **Content-Type** используется значение в элементе attachments, если оно задано, в противном случае - **application/octet-stream**. **Примечание:** Отправка фрагментов сообщения в виде вложений, не определенных как таковые в WSDL, может нарушить соответствие спецификации WS-I Attachments Profile 1.0, и это следует избегать по мере возможности.

Как составляются вложения, если элемент attachment не существует в SMO

В следующей таблице показано, как составляется и отправляется вложение, если SMO не содержит элемент **attachment** с **bodyPath**, соответствующим части с именем сообщения:

Таблица 62. Как генерируется вложение

Состояние связывания MIME WSDL для главного двоичного фрагмента сообщения	**Как составляется и отправляется сообщение**
Выполняется одно из следующих условий: • Не определен тип содержимого для фрагмента сообщения • Определено несколько типов содержимого • Определен тип содержимого с символом подстановки	Фрагмент сообщения отправляется как вложение. Генерируется **Content-Id**. Для **Content-Type** присваивается значение **application/octet-stream**.
Присутствует, фрагмент сообщения содержит одну часть без символов подстановки	Фрагмент сообщения отправляется как вложение. Генерируется **Content-Id**. Для **Content-Type** присваивается тип, заданный в элементе содержимого MIME WSDL.
Отсутствует	Фрагмент сообщения не отправляется как вложение.

Важное замечание: Как описано в разделе “Представление SMO в формате XML” примитив передачи Mapping преобразует сообщения посредством преобразования XSLT 1.0. Это преобразование работает с сериализованным SMO в формате XML. Примитив передачи Mapping позволяет задавать корневой элемент сериализации, который соответствует корневому элементу документа XML.

При отправке сообщений SOAP с вложениями выбранный корневой элемент определяет, как передаются сообщения.

- Если корневой элемент в карте XML - это “/body”, то по умолчанию все вложения передаются по карте.
- Если корневой элемент в карте - это “/”, то передачей вложений можно управлять.

Нессылочные вложения: Advanced

Можно отправлять и получать *нессылочные вложения*, то есть вложения, не объявленные в интерфейсе службы.

В сообщении SOAP с несколькими фрагментами MIME тело SOAP является первым фрагментом сообщения, а вложения передаются в последующих фрагментах. В тело SOAP не включается ссылка на вложение.

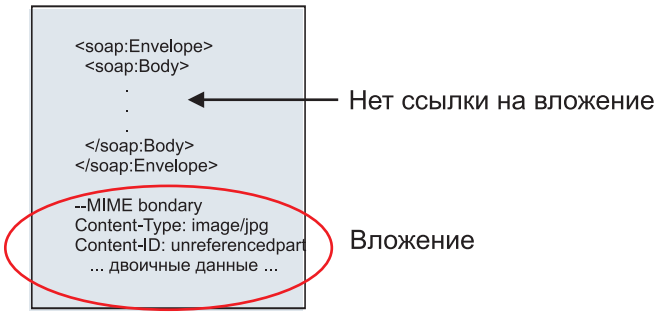


Рисунок 67. Сообщение SOAP с нессылочным вложением

Сообщение SOAP с нессылочным вложением можно отправить через экспорт веб-службы в импорт веб-службы. Исходящее сообщение, отправляемое в целевую веб-службу, содержит вложение.

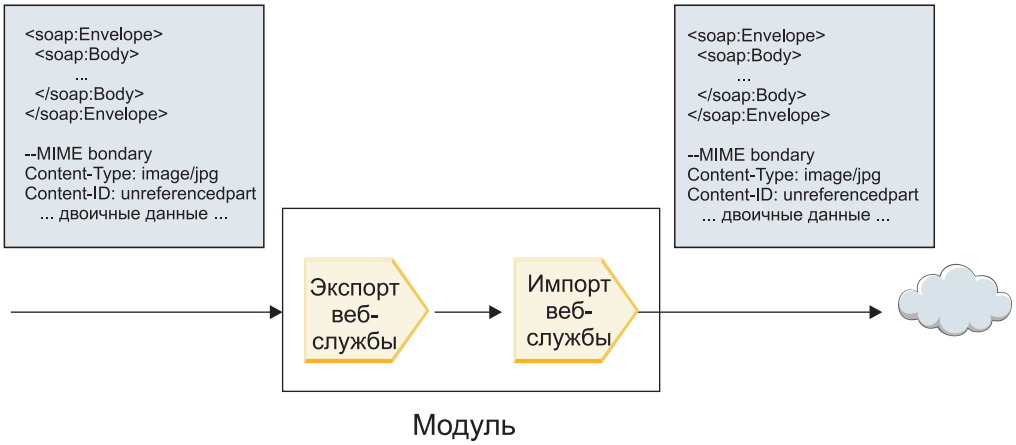


Рисунок 68. Прохождение вложения через модуль SCA

В рис. 23 на стр. 95 сообщение SOAP, содержащее вложение, проходит без изменений.

Изменить сообщение SOAP можно с помощью компонента потока передачи. Так, компонент потока передачи может извлечь данные из сообщения SOAP (двоичные данные в теле сообщения) и создать

сообщение SOAP с вложением. Данные обрабатываются как часть элемента вложения объекта сообщения службы (SMO).

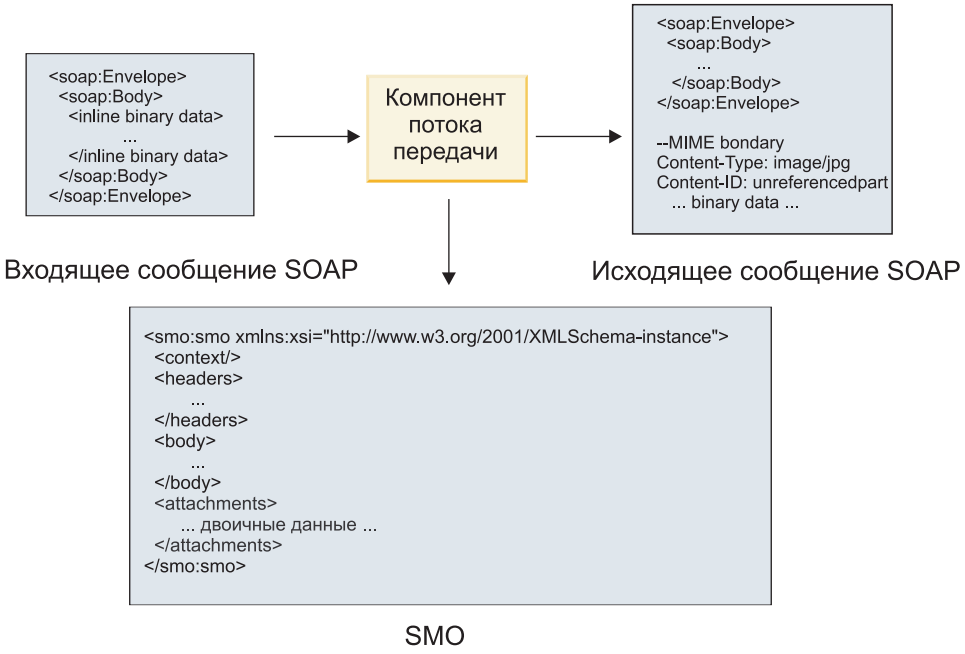


Рисунок 69. Сообщение, обработанное компонентом потока передачи

Компонент потока передачи также может преобразовывать входящее сообщение, извлекая из него вложение, кодируя его и передавая сообщение без вложений.

Вместо того чтобы извлекать данные из входящего сообщения SOAP, данные вложения можно получить из внешнего источника, например, из базы данных, и создать сообщение SOAP с вложениями.

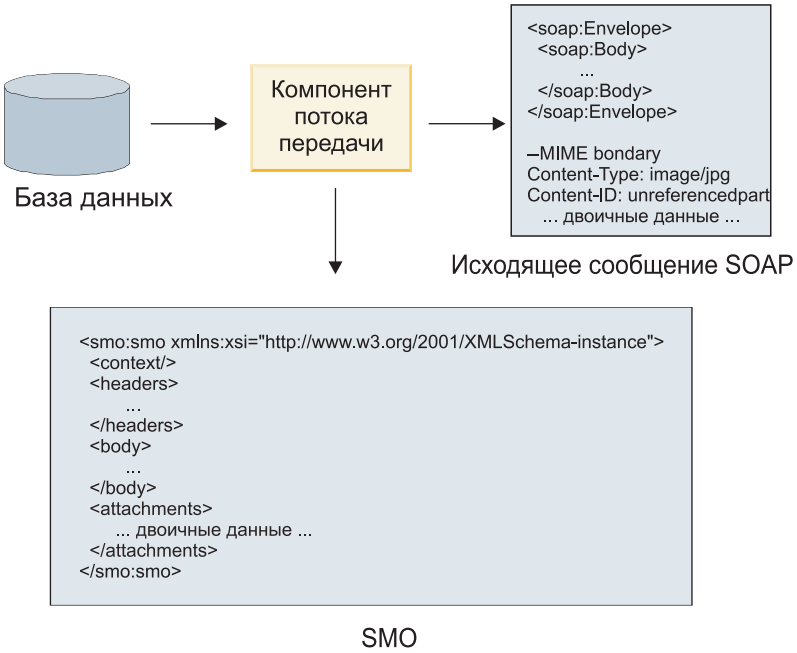


Рисунок 70. Вложение, полученное из базы данных и добавленное в сообщение SOAP

Напротив, компонент потока передачи может извлечь вложение из входящего сообщения SOAP и обработать сообщение, например, сохранив его в базе данных.

Нессылочные вложения могут передаваться только по компонентам потока передачи. Если вложение должно быть доступно компоненту другого тип, или если его требуется передать в другой тип компонента, то используйте компонент потока передачи для перемещения вложения в расположение, доступное такому компоненту.

Важное замечание: Как описано в разделе “Представление SMO в формате XML” примитив передачи Mapping преобразует сообщения посредством преобразования XSLT 1.0. Это преобразование работает с сериализованным SMO в формате XML. Примитив передачи Mapping позволяет задавать корневой элемент сериализации, который соответствует корневому элементу документа XML.

При отправке сообщений SOAP с вложениями выбранный корневой элемент определяет, как передаются сообщения.

- Если корневой элемент в карте XML - это “/body”, то по умолчанию все вложения передаются по карте.
- Если корневой элемент в карте - это “/”, то передачей вложений можно управлять.

Использование связывания стиля документа WSDL с многофрагментными сообщениями: Advanced

Организация Web Services Interoperability Organization (WS-I) определила правила для описания веб-служб посредством WSDL и способы формирования соответствующих сообщений SOAP, обеспечивающие возможность взаимодействия.

Эти правила описаны в спецификации WS-I *Basic Profile Version 1.1* (http://www.ws-i.org/Profiles/BasicProfile-1.1.html). В спецификации WS-I Basic Profile 1.1 R2712 говорится: "Связывание document-literal НЕОБХОДИМО сериализовать как ENVELOPE с soap:Body, дочерний элемент которого является экземпляром объявления глобального элемента, на который ссылается соответствующий фрагмент wsdl:message".

Это означает, что при использовании стиля документа для связывания SOAP в операциях с сообщениями (ввод, вывод или ошибка), для которых определено несколько фрагментов, только один из этих фрагментов будет связан с телом SOAP для обеспечения совместимости со спецификацией WS-I Basic Profile 1.1.

Далее в спецификации WS-I Attachments Profile 1.0 R2941 говорится: "wsdl:binding в DESCRIPTION ДОЛЖЕН связывать каждый из элементов wsdl:part в wsdl:message из wsdl:portType, к которому он относится, с одним из элементов soapbind:body, soapbind:header, soapbind:fault, soapbind:headerfault или mime:content".

Это означает, что при использовании стиля document для связывания SOAP в операциях с сообщениями (ввод, вывод или ошибка), для которых определено несколько фрагментов, все фрагменты, за исключением связанного с телом SOAP, должны быть связаны как вложения или заголовки.

Следующая процедура применяется, когда описания WSDL создаются для экспортов со связыванием веб-службы (JAX-WS и JAX-RPC):

- Если в сообщении есть несколько элементов с типом, отличным от двоичного, то можно выбрать, какой фрагмент сообщения будет связан с телом SOAP. Если есть только один элемент с типом, отличным от двоичного, то он автоматически будет связан с телом SOAP.
- Для связывания JAX-WS все прочие фрагменты сообщения с типом "hexBinary" или "base64Binary" связываются как ссылочные вложения. См. раздел “Ссылочные вложения: фрагменты сообщений верхнего уровня” на стр. 89.
- Все прочие фрагменты сообщения связываются как заголовки SOAP.

Связывание импорта JAX-RPC и JAX-WS учитывает связывание SOAP в существующем документе WSDL с сообщениями в стиле multipart-document, даже если эти множественные фрагменты не связаны с телом SOAP. Однако создавать клиентов веб-службы для таких документов WSDL в Rational Application Developer будет невозможно.

Примечание: Связывание JAX-RPC не поддерживает вложения.

Поэтому рекомендуется следующая процедура для работы с многофрагментными сообщениями в операциях со связыванием SOAP стиля document:

1. Используйте оболочечный стиль document/literal. В этом случае сообщение всегда будет иметь один фрагмент, однако вложения будут нессылочными (как описано в разделе “Нессылочные вложения” на стр. 95) или иметь тип swaRef (как описано в разделе “Ссылочные вложения: элементы типа swaRef” на стр. 86).
2. Используйте стиль RPC/literal. В этом случае связывание WSDL не налагает никаких ограничений на число фрагментов, связанных с телом SOAP. Итоговое сообщение SOAP будет иметь один дочерний элемент, который представляет вызываемую операцию, а фрагменты сообщения будут дочерними по отношению к этому элементу.
3. Для связывания JAX-WS создавайте не более одного фрагмента сообщения с типом, отличающимся от "hexBinary" или "base64Binary", в противном случае остальные недвоичные фрагменты будут связываться с заголовками SOAP.
4. Все прочие случаи подпадают под описанное поведение.

Примечание: При использовании сообщений SOAP, не соответствующих спецификации WS-I *Basic Profile Version 1.1*, существуют дополнительные ограничения.

- Первый фрагмент сообщения не должен быть двоичным.
- При приеме сообщений SOAP в стиле multipart-document со ссылочными вложениями связывание JAX-WS ожидает, что каждое ссылочное вложение будет представлено дочерним элементом тела SOAP с атрибутом href, который идентифицирует вложение по его content-ID. Связывание JAX-WS отправляет ссылочные вложения для таких сообщений аналогичным образом. Такое поведение не соответствует спецификации WS-I Basic Profile.

  Для того чтобы сообщения соответствовали спецификации Basic Profile, следуйте процедуре 1 на стр. 98 или 2 на стр. 98 из предыдущего списка или вместо ссылочных вложений используйте в таких сообщениях нессылочные вложения или вложения с типом swaRef.

Привязки HTTP

Привязка HTTP предназначена для поддержки соединения SCA с HTTP. Она дает возможность встроить существующие и новые приложения HTTP в среду SCA.

Протокол HTTP широко используется для передачи информации в Интернете. Для работы с внешним приложением, которое использует протокол HTTP, необходима привязка HTTP. Привязка HTTP преобразует данные, полученные в сообщении во внутреннем формате, в бизнес-объект в приложении SCA. Также привязка HTTP преобразует данные, возвращенные в виде бизнес-объекта, во внутренний формат, который поддерживается внешним приложением.

Примечание: Для взаимодействия с клиентами и службами, использующими протокол SOAP/HTTP веб-служб, рекомендуется использовать одну из привязок веб-служб, которая предоставляет более широкий набор возможностей для обеспечения стандартного качества обслуживания веб-службы.

Ниже описаны некоторые стандартные сценарии использования привязки HTTP:

- Службы SCA могут вызывать приложения HTTP, используя объект импорта HTTP.
- Службы SCA могут экспортировать себя в виде приложений HTTP, которые могут применяться клиентами HTTP, с помощью объектов экспорта HTTP.

- IBM Business Process Manager и Process Server могут взаимодействовать друг с другом, используя инфраструктуру HTTP, поэтому пользователи могут управлять этим взаимодействием в соответствии с корпоративными стандартами.
- IBM Business Process Manager и Process Server могут выполнять роль посредников в обмене данными по протоколу HTTP, которые преобразуют и маршрутизируют сообщения для улучшения интеграции приложений.
- IBM Business Process Manager и Process Server могут выполнять функции моста между протоколом HTTP и другими протоколами, такими как веб-службы SOAP/HTTP, адаптеры ресурсов Java Connector Architecture (JCA), JMS и т. п.

Подробная информация о создании привязок импорта и экспорта HTTP приведена в справочной системе Integration Designer Information Center. Обратитесь к разделам в категории **Разработка приложений интеграции > Доступ к внешним службам по HTTP>**.

Обзор привязок HTTP:

Привязка HTTP обеспечивает связь с приложениями HTTP. Она передает данные между приложениями HTTP и позволяет вызывать приложения HTTP в модуле.

Привязки импорта HTTP

Привязка импорта HTTP обеспечивает передачу исходящих данных из приложений SCA на сервер или в приложения HTTP.

Объект импорта вызывает URL конечной точки HTTP. URL может быть указан одним из трех способов:

- URL может настраиваться динамически в заголовках HTTP с помощью динамически переопределяемого URL.
- URL может настраиваться динамически в элементе целевого адреса SMO.
- URL может быть задан в свойстве конфигурации объекта импорта.

Такой вызов по сути всегда является синхронным.

Хотя вызов HTTP всегда выполняется по схеме запрос-ответ, объект импорта HTTP поддерживает как однонаправленные, так и двунаправленные операции, игнорируя ответ в случае однонаправленной операции.

Привязки экспорта HTTP

Привязка экспорта HTTP обеспечивает передачу входящих данных из приложений HTTP в приложение SCA.

URL задается в объекте экспорта HTTP. Если приложению HTTP необходимо отправить сообщение с запросом в объект экспорта, то они вызывают объект экспорта, используя этот URL.

Объект экспорта HTTP также поддерживает вызовы ping.

Привязки HTTP во время выполнения

Во время выполнения объект импорта с привязкой HTTP отправляет запрос, который может и не содержать данных в теле сообщения, из приложения SCA во внешнюю веб-службу. Приложение SCA отправляет запрос внешней веб-службе, как показано на рисунке рис. 26 на стр. 100.

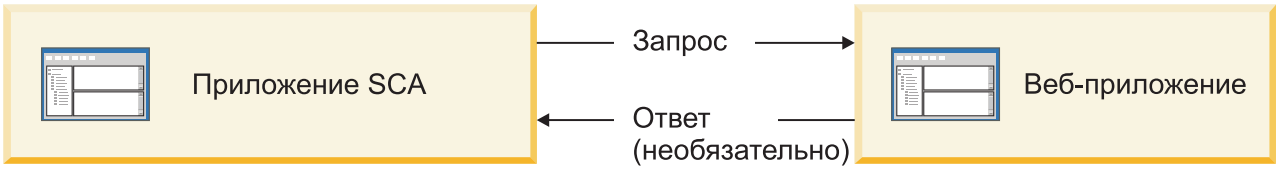


Рисунок 71. Передача запроса из приложения SCA в веб-приложение

При необходимости объект импорта с привязкой HTTP получает данные, возвращенные веб-приложением в ответ на запрос.

В случае экспорта приложение-клиент запрашивает веб-службу, как показано на рисунке рис. 27 на стр. 100.

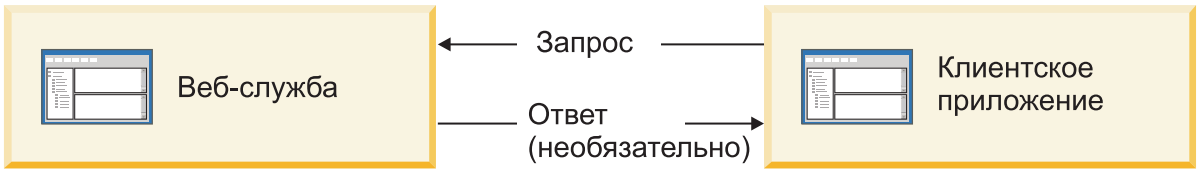


Рисунок 72. Передача запроса от клиентского приложения в веб-службу.

Веб-служба - это веб-приложение, работающее на сервере. Объект экспорта реализуется в веб-приложении в виде сервлета, поэтому клиент отправляет запрос на URL-адрес. Сервлет передает запрос в приложение SCA в среде выполнения.

При необходимости объект экспорта отправляет в приложение-клиент данные в ответ на запрос.

Заголовки HTTP:

В привязках импорта и экспорта HTTP можно настраивать заголовки HTTP и их значения, применяемые в исходящих сообщениях. В объектах импорта HTTP эти заголовки применяются для запросов, а в объектах экспорта HTTP - для ответов.

Статические заголовки и управляющая информация имеют больший приоритет, чем значения, которые динамически настраиваются во время выполнения. Однако значения метода, версии и URL для динамического переопределения используются вместо статических значений, которые во всех остальных случаях применяются по умолчанию.

Привязка поддерживает динамическую настройку URL импорта HTTP, определяя значение метода, версии и целевого URL HTTP во время выполнения. Для определения этих значений извлекается значение ссылки на конечную точку, URL для динамического переопределения, версии и метода.

- Для настройки ссылку на конечную точку используйте API com.ibm.websphere.sca.addressing.EndpointReference или укажите поле /headers/SMOHeader/Target/address в заголовке SMO.
- Для настройки URL для динамического переопределения, версии и метода используйте раздел управляющих параметров HTTP в сообщении SCA. Обратите внимание, что URL для динамического переопределения имеет более высокий приоритет, чем ссылка на целевую конечную точку, однако ссылка на конечную точку не зависит от привязки, поэтому рекомендуется использовать именно ее.

Привязки экспорта и импорта HTTP обрабатывают управляющую информацию и заголовки исходящих сообщений в следующем порядке:

1. Заголовок и управляющая информация, включая URL для динамического переопределения, версию и метод из сообщения SCA (наименее приоритетные)
2. Изменения, внесенные на уровне объекта импорта или экспорта в административной консоли

3. Изменения, внесенные на уровне метода объекта импорта или экспорта в административной консоли
4. Целевой адрес, заданный в ссылке на конечную точку или заголовке SMO
5. URL для динамического переопределения, версия и метод из сообщения SCA
6. Заголовки и управляющая информация из обработчика данных или привязки данных (наиболее приоритетные)

Объекты импорта и экспорта HTTP заполняют заголовки и управляющие параметры при приеме данных, используя содержимое входящего сообщения (HTTPExportRequest или HTTPImportResponse), если параметр передачи заголовков протокола равен **True**. И наоборот, объекты импорта и экспорта HTTP считывают и обрабатывают заголовки и управляющие параметры в исходящих данных (HTTPExportResponse или HTTPImportRequest), если параметр передачи заголовков протокола равен **True**.

Примечание: Изменения заголовков и управляющих параметров в ответах импорта и запросах экспорта с помощью обработчиков данных или привязок данных не изменяют порядок обработки сообщения в привязках импорта или экспорта и могут использоваться только для передачи измененных значений в нижележащие компоненты SCA.

Служба контекста отвечает за передачу информации о контексте (в том числе заголовков протокола, таких как заголовок HTTP, и контекста пользователя, такого как ИД учетной записи) вместе с путем для вызова SCA. Во время разработки в IBM Integration Designer для управления передачей контекста можно использовать свойства объектов импорта и экспорта. Более подробные сведения приведены в описании привязок импорта и экспорта в справочной системе IBM Integration Designer Information Center.

Структуры заголовков HTTP и их поддержка

Табл. 34 на стр. 101 содержит перечень параметров для запросов и ответов объектов импорта и экспорта HTTP.

Таблица 63. Предоставляемая информация заголовка HTTP

Имя управляющего параметра	**Запрос импорта HTTP**	**Ответ импорта HTTP**	**Запрос экспорта HTTP**	**Ответ экспорта HTTP**
URL	Игнорируется	Не задано	Считывается из сообщения с запросом. **Примечание:** Строка запроса также является частью управляющего параметра URL.	Игнорируется
Версия (возможные значения: 1.0, 1.1; значение по умолчанию - 1.1)	Игнорируется	Не задано	Считывается из сообщения с запросом	Игнорируется
Метод	Игнорируется	Не задано	Считывается из сообщения с запросом	Игнорируется

Таблица 63. Предоставляемая информация заголовка HTTP (продолжение)

Имя управляющего параметра	Запрос импорта HTTP	Ответ импорта HTTP	Запрос экспорта HTTP	Ответ экспорта HTTP
URL для динамического переопределения	Если задан в обработчике данных или привязке данных, то переопределяет URL импорта HTTP. Сохраняется в сообщении в строке запроса. **Примечание:** Строка запроса также является частью управляющего параметра URL.	Не задано	Не задано	Игнорируется
Версия для динамического переопределения	Если задан, то переопределяет версию импорта HTTP. Сохраняется в сообщении в строке запроса.	Не задано	Не задано	Игнорируется
Метод для динамического переопределения	Если задан, то переопределяет метод импорта HTTP. Сохраняется в сообщении в строке запроса.	Не задано	Не задано	Игнорируется
Тип среды передачи (этот управляющий параметр содержит фрагмент значения заголовка HTTP Content-Type)	Если задан, то сохраняется в сообщении в заголовке Content-Type. **Примечание:** Это значение управляющего элемента должно предоставляться обработчиком данных или привязкой данных.	Считывается из заголовка Content-Type сообщения с ответом	Считывается из заголовка Content-Type сообщения с запросом	Если задан, то сохраняется в сообщении в заголовке Content-Type. **Примечание:** Это значение управляющего элемента должно предоставляться обработчиком данных или привязкой данных.
Набор символов (по умолчанию: UTF-8)	Если задан, то сохраняется в сообщении в заголовке Content-Type. **Примечание:** Это значение управляющего элемента должно предоставляться привязкой данных.	Считывается из заголовка Content-Type сообщения с ответом	Считывается из заголовка Content-Type сообщения с запросом	Поддерживается; сохраняется в сообщении в заголовке Content-Type. **Примечание:** Это значение управляющего элемента должно предоставляться привязкой данных.

Таблица 63. Предоставляемая информация заголовка HTTP (продолжение)

Имя управляющего параметра	**Запрос импорта HTTP**	**Ответ импорта HTTP**	**Запрос экспорта HTTP**	**Ответ экспорта HTTP**
Кодировка передачи (возможные значения: chunked, identity; значение по умолчанию - identity)	Если задан, то сохраняется в сообщении в виде заголовка и определяет кодировку при преобразовании сообщения.	Считывается из сообщения с ответом	Считывается из сообщения с запросом	Если задан, то сохраняется в сообщении в виде заголовка и определяет кодировку при преобразовании сообщения.
Кодировка содержимого (возможные значения: gzip, x-gzip, deflate, identity; значение по умолчанию - identity)	Если задан, то сохраняется в сообщении в виде заголовка и определяет кодировку данных.	Считывается из сообщения с ответом	Считывается из сообщения с запросом	Если задан, то сохраняется в сообщении в виде заголовка и определяет кодировку данных.
Длина содержимого	Игнорируется	Считывается из сообщения с ответом	Считывается из сообщения с запросом	Игнорируется
Код состояния (по умолчанию: 200)	Не поддерживается	Считывается из сообщения с ответом	Не поддерживается	Если задан, то сохраняется в сообщении в строке ответа
Причина (по умолчанию: OK)	Не поддерживается	Считывается из сообщения с ответом	Не поддерживается	Это управляющее значение игнорируется. Значение в строке ответа генерируется с учетом кода состояния.
Идентификация (содержит несколько свойств)	Если задан, то используется для создания заголовка простой идентификации. **Примечание:** Значение этого заголовка кодируется только в протоколе HTTP. В SCA оно раскодируется и передается в виде обычного текста.	Неприменимо	Считывается из заголовка Basic Authentication сообщения с запросом. Наличие этого заголовка не говорит о том, что пользователь был идентифицирован. Для управления идентификацией должна применяться конфигурация сервлета. **Примечание:** Значение этого заголовка кодируется только в протоколе HTTP. В SCA оно раскодируется и передается в виде обычного текста.	Неприменимо
Proxy (содержит несколько свойств: Хост, Порт, Идентификация)	Если задан, то используется для установления соединения через Proxy.	Неприменимо	Неприменимо	Неприменимо

Таблица 63. Предоставляемая информация заголовка HTTP (продолжение)

Имя управляющего параметра	Запрос импорта HTTP	Ответ импорта HTTP	Запрос экспорта HTTP	Ответ экспорта HTTP
SSL (содержит несколько свойств: Хранилище ключей, Пароль хранилища ключей, Хранилище доверенных сертификатов, Пароль хранилища доверенных сертификатов, Идентификация клиента)	Если параметр задан, и целевой URL использует HTTPS, то параметр применяется для установления соединения SSL.	Неприменимо	Неприменимо	Неприменимо

Привязки данных HTTP:

Для каждой схемы преобразования сообщений SCA в сообщения протокола HTTP необходимо настроить обработчик данных или привязку данных HTTP. Обработчики данных предоставляют интерфейс, не зависящий от типа привязки, который может использоваться в любых привязках транспортных протоколов, и поэтому рекомендуется использовать именно их. Привязки данных предназначены для определенной привязки транспортного протокола. В составе продукта предусмотрены классы привязок данных для протокола HTTP, и их можно дополнить собственными обработчиками или привязками данных.

Примечание: Три описанных здесь класса привязок данных HTTP (HTTPStreamDataBindingSOAP, HTTPStreamDataBindingXML и HTTPServiceGatewayDataBinding) устарели и больше не используются в IBM Business Process Manager версии 7.0. Вместо описанных в этом разделе привязок данных рекомендуется использовать следующие обработчики данных:

- SOAPDataHandler вместо HTTPStreamDataBindingSOAP.
- UTF8XMLDataHandler вместо HTTPStreamDataBindingXML
- GatewayTextDataHandler вместо HTTPServiceGatewayDataBinding

Для использования в объектах импорта и экспорта HTTP предусмотрены следующие привязки данных: привязка двоичных данных, привязка данных XML и привязка данных SOAP. Привязка данных ответа не требуется для однонаправленных операций. Для представления привязки данных используется имя класса Java, экземпляры которого могут выполнять преобразование данных HTTP в ServiceDataObject и обратное преобразование. Для определения необходимой привязки данных и выполняемой операции в объекте экспорта должен применяться селектор функций в сочетании с привязками методов. Предоставляются следующие привязки данных:

- Привязки двоичных данных, рассматривающие тело как неструктурированные двоичные данные. Ниже приведена реализация схемы XSD привязки двоичных данных:

```
<xsd:schema elementFormDefault="qualified"
  targetNamespace="http://com.ibm.websphere.http.data.bindings/schema"
  xmlns:tns="http://com.ibm.websphere.http.data.bindings/schema"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:complexType name="HTTPBaseBody">
    <xsd:sequence/>
  </xsd:complexType>

  <xsd:complexType name="HTTPBytesBody">
    <xsd:complexContent>
      <xsd:extension base="tns:HTTPBaseBody">
        <xsd:sequence>
          <xsd:element name="value" type="xsd:hexBinary"/>
```

```
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```

- Привязки данных XML, которые поддерживают тело, содержащее данные XML. Реализация привязки данных XML аналогична реализации привязки данных JMS XML и не накладывает никакие ограничения на схему интерфейса.
- Привязки данных SOAP, поддерживающие тело, содержащее данные SOAP. Реализация привязки данных SOAP не накладывает никакие ограничения на схему интерфейса.

Реализация пользовательских привязок данных HTTP

В этом разделе приведена информация о том, как реализовать пользовательскую привязку данных HTTP.

Примечание: Рекомендуется реализовать пользовательский обработчик данных вместо привязки, поскольку его можно использовать в разных привязках транспортных протоколов.

HTTPStreamDataBinding - это интерфейс для обработки пользовательских сообщений HTTP. Этот интерфейс разработан для поддержки обработки данных большого размера. Однако для его работы необходимо, чтобы привязка данных возвращала управляющую информацию и заголовки перед записью сообщения в поток.

Пользовательская привязка данных должна реализовывать следующие методы в указанном порядке.

Для настройки привязки данных создайте класс, реализующий HTTPStreamDataBinding. Привязка данных должна содержать четыре частных свойства:

- private DataObject pDataObject
- private HTTPControl pCtrl
- private HTTPHeaders pHeaders
- private yourNativeDataType nativeData

Привязка HTTP будет вызывать методы привязки данных в следующем порядке:

- Обработка исходящих данных (преобразование DataObject во внутренний формат):
  1. setDataObject(...)
  2. setHeaders(...)
  3. setControlParameters(...)
  4. setBusinessException(...)
  5. convertToNativeData()
  6. getControlParameters()
  7. getHeaders()
  8. write(...)
- Обработка входящих данных (преобразование из внутреннего формата в DataObject):
  1. setControlParameters(...)
  2. setHeaders(...)
  3. convertFromNativeData(...)
  4. isBusinessException()
  5. getDataObject()
  6. getControlParameters()
  7. getHeaders()

Метод setDataObject(...) должен вызываться в методе convertFromNativeData(...) для настройки значения dataObject, которое необходимо преобразовать из внутреннего формата в содержимое частного свойства "pDataObject".

```
public void setDataObject(DataObject dataObject)
   throws DataBindingException {
  pDataObject = dataObject;

}
public void setControlParameters(HTTPControl arg0) {
  this.pCtrl = arg0;
}

public void setHeaders(HTTPHeaders arg0) {
  this.pHeaders = arg0;
}
/*
* Добавить заголовок http "IsBusinessException" в pHeaders.
* Выполняется в два шага:
* 1. Удалить все заголовки с именем IsBusinessException (без учета регистра).
*    Это гарантирует наличие только одного заголовка.
* 2. Добавить новый заголовок "IsBusinessException"
*/
public void setBusinessException(boolean isBusinessException) {
  //удалить все заголовки с именем IsBusinessException (в любом регистре)
  //это обеспечит наличие только одного заголовка
  //добавить новый заголовок "IsBusinessException", например:
  HTTPHeader header=HeadersFactory.eINSTANCE.createHTTPHeader();
  header.setName("IsBusinessException");
  header.setValue(Boolean.toString(isBusinessException));
  this.pHeaders.getHeader().add(header);

}
public HTTPControl getControlParameters() {
  return pCtrl;
}
public HTTPHeaders getHeaders() {
  return pHeaders;
}
public DataObject getDataObject() throws DataBindingException {
  return pDataObject;
}
/*
* Получить заголовок "IsBusinessException" из pHeaders, вернуть его булевское значение
*/
public boolean isBusinessException() {
  String headerValue = getHeaderValue(pHeaders,"IsBusinessException");
  boolean result=Boolean.parseBoolean(headerValue);
  return result;
}
public void convertToNativeData() throws DataBindingException {
  DataObject dataObject = getDataObject();
  this.nativeData=realConvertWorkFromSDOToNativeData(dataObject);
}
public void convertFromNativeData(HTTPInputStream arg0){
  //Пользовательский метод для
  //чтения данных из HTTPInputStream
  //Преобразовать их в DataObject
  DataObject dataobject=realConvertWorkFromNativeDataToSDO(arg0);
  setDataObject(dataobject);
}
public void write(HTTPOutputStream output) throws IOException {
  if (nativeData != null)
  output.write(nativeData);
}
```

Привязки EJB

Привязки импортов Enterprise JavaBeans (EJB) позволяют компонентам архитектуры компонентов служб (SCA) вызывать службы, предоставляемые бизнес-логикой Java EE, работающей на сервере Java EE. Привязки экспортов EJB позволяют компонентам SCA экспортироваться как EJB, чтобы бизнес-логика Java EE могла вызывать компоненты SCA, которые в противном случае недоступны ей.

Привязки импортов EJB:

Привязки импортов EJB позволяют модулю SCA вызывать реализации EJB определением способа привязки использующего модуля к внешнему EJB. Импорт служб из внешней реализации EJB позволяет пользователям встроить свою бизнес-логику в среду IBM Business Process Manager и и принимать участие в бизнес-процессе.

Для создания привязок импортов EJB используется Integration Designer. Для создания привязок можно использовать любую из следующих процедур:

- Создание импорта EJB с помощью мастера внешних служб

  Для создания импорта EJB на основе существующей реализации можно использовать мастер внешних служб в Integration Designer. Мастер внешних служб создает службы на основе заданных критериев. Затем он создает бизнес-объекты, интерфейсы и файлы импорта на основе найденных служб.
- Создание импорта EJB с помощью редактора сборки

  Импорт EJB можно создать в пределах диаграммы сборки с помощью редактора сборки Integration Designer. Для создания привязки EJB из палитры можно использовать Импорт или класс Java.

Созданный импорт имеет привязки данных, которые устанавливают связь Java-WSDL вместо вызова компонента моста Java. Можно напрямую соединить компонент со ссылкой языка описания веб-служб (WSDL) с импортом, который связывается со службой на основе EJB с помощью интерфейса Java.

Импорт EJB может взаимодействовать с бизнес-логикой Java EE с помощью программной модели EJB 2.1 или EJB 3.0.

Вызов бизнес-логики Java EE может быть локальным (только для EJB 3.0) или удаленным.

- Локальный вызов используется для вызова бизнес-логики Java EE, которая находится на том же сервере, что и импорт.

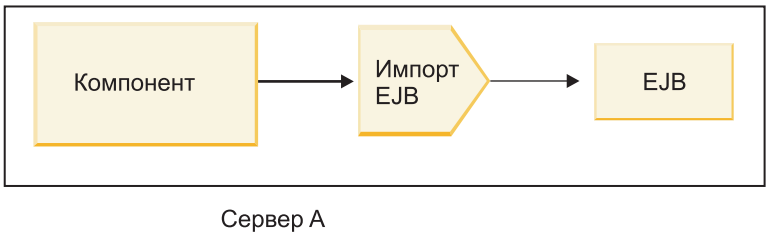


Рисунок 73. Локальный вызов EJB (только для EJB 3.0)

- Удаленный вызов используется для вызова бизнес-логики Java EE, которая не находится на том же сервере, что и импорт.

  Например, на следующем рисунке импорт EJB использует удаленный метод вызова через протокол Internet InterORB (RMI/IIOP) для вызова метода EJB на другом сервере.

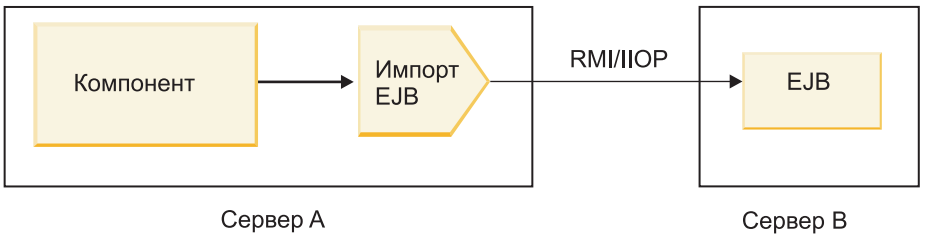


Рисунок 74. Удаленный вызов EJB

При настройке привязки EJB Integration Designer использует имя JNDI для определения уровня программной модели EJB и типа вызова (локального или удаленного).

Привязки импортов EJB содержат следующие основные компоненты:

- Обработчик данных JAX-WS
- Селектор ошибок EJB
- Селектор функций импорта EJB

Если пользовательский сценарий не основан на преобразовании JAX-WS, то могут потребоваться обработчик пользовательских данных, селектор функций и селектор по умолчанию для выполнения задач, которые в противном случае выполняются компонентами привязок импортов EJB. Это включает преобразование, обычно выполняемое пользовательским алгоритмом преобразования.

Привязки экспортов EJB:

Внешние приложения Java EE могут вызывать компонент SCA при помощи привязки экспорта EJB. Экспорт EJB позволяет экспортировать компоненты SCA, чтобы внешние приложения Java EE могли вызывать эти компоненты с помощью программной модели EJB.

Примечание: Экспорт EJB является EJB без сохранения состояния.

Для создания привязок EJB используется Integration Designer. Для создания привязок можно использовать любую из следующих процедур:

- Создание привязок экспортов EJB с помощью мастера внешних служб

  Для создания службы экспорта EJB на основе существующей реализации можно использовать мастер внешних служб в Integration Designer. Мастер внешних служб создает службы на основе заданных критериев. Затем он создает бизнес-объекты, интерфейсы и файлы экспорта на основе найденных служб.
- Создание привязок экспортов EJB с помощью редактора сборки

  Экспорт EJB можно создать с помощью редактора сборки Integration Designer.

Важное замечание: Клиент Java 2 Platform, Standard Edition (J2SE) не может вызывать клиент экспорта EJB, который создан в Integration Designer.

Привязку можно создать из существующего компонента SCA или можно создать экспорт с привязкой EJB для интерфейса Java.

- При создании экспорта для существующего компонента SCA с существующим интерфейсом WSDL экспорту назначается интерфейс Java.
- При создании экспорта для интерфейса Java для экспорта можно выбрать интерфейс WSDL или Java.

  Примечание: Интерфейс Java, используемый для создания экспорта EJB, имеет следующие ограничения в отношении объектов (параметров ввода и вывода и исключительных ситуаций), передаваемых в качестве параметров по удаленному вызову:
  - Они должны быть конкретного типа (а не интерфейсом или абстрактным типом).

– Они должны соответствовать спецификации EJB (Enterprise JavaBean). Они должны быть сериализуемы и иметь безаргументный конструктор по умолчанию, а все свойства должны быть доступны через методы getter и setter.

  Информация о спецификации EJB (Enterprise JavaBean) приведена на веб-сайте Sun Microsystems, Inc., по адресу http://java.sun.com.

Кроме того, исключительная ситуация должна быть проверенной исключительной ситуацией, унаследованной из java.lang.Exception, и она должна быть единственной (то есть выброс проверенных исключительных ситуаций нескольких типов не поддерживается).

Отметим также, что бизнес-интерфейс Java EnterpriseBean является простым интерфейсом Java и не должен расширять javax.ejb.EJBObject или javax.ejb.EJBLocalObject. Методы бизнес-интерфейса не должны выбрасывать java.rmi.Remote.Exception.

Привязки экспорта EJB могут взаимодействовать с бизнес-логикой Java EE с помощью программной модели EJB 2.1 или EJB 3.0.

Вызов может быть локальным (только для EJB 3.0) или удаленным.

- Локальный вызов используется, когда бизнес-логика Java EE вызывает компонент SCA, который находится на том же сервере, что и экспорт.
- Удаленный вызов используется, когда бизнес-логика Java EE не находится на том же сервере, что и экспорт.

  Например, на следующем рисунке EJB использует RMI/IIOP для вызова компонента SCA на другом сервере.

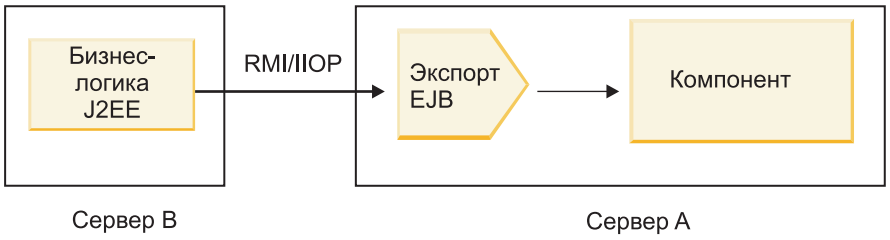


Рисунок 75. Удаленный вызов из клиента компонента SCA при помощи экспорта EJB

При настройке привязки EJB Integration Designer использует имя JNDI для определения уровня программной модели EJB и типа вызова (локального или удаленного).

Привязки экспортов EJB содержат следующие основные компоненты:

- Обработчик данных JAX-WS
- Селектор функций экспорта EJB

Если пользовательский сценарий не основан на преобразовании JAX-WS, то могут потребоваться обработчик пользовательских данных и селектор функций для выполнения задач, которые в противном случае выполняются компонентами привязок экспортов EJB. Это включает преобразование, обычно выполняемое пользовательским алгоритмом преобразования.

Свойства привязки EJB:

Привязки импортов EJB используют настроенные для них имена JNDI для определения программной модели EJB и типа вызова (локального или удаленного). Привязки импортов и экспортов EJB используют обработчик данных JAX-WS для преобразования данных. Привязки импортов EJB использует селектор функций импорта EJB и селектор ошибок EJB, а привязка экспорта EJB использует селектор функций экспорта EJB.

Имена JNDI и привязки импортов EJB:

При настройке привязки EJB на импорте Integration Designer использует имя JNDI для определения уровня программной модели EJB и типа вызова (локального или удаленного).

Если имя JNDI не задано, то используется привязка интерфейса EJB по умолчанию. Создаваемые имена по умолчанию зависят от вызываемого объекта: EJB 2.1 JavaBean или EJB 3.0 JavaBean.

Примечание: Более подробная информация о соглашении об именах содержится в разделе "Обзор связывания приложений EJB 3.0" справочной системе WebSphere Application Server Information Center.

- EJB 2.1 JavaBean

  Имя JNDI по умолчанию, предварительно выбираемое Integration Designer , представляет собой привязку EJB 2.1 по умолчанию, which takes the которое принимает вид **ejb/** плюс домашний интерфейс, разделенные косыми чертами.

  Например, для домашнего интерфейса EJB 2.1 JavaBean для com.mycompany.myremotebusinesshome привязкой по умолчанию является:

  ```
  ejb/com/mycompany/myremotebusinesshome
  ```

  Для EJB 2.1 поддерживается только удаленный вызов EJB.

- EJB 3.0 JavaBean

  Именем JNDI по умолчанию, предварительно выбираемым Integration Designer для локального JNDI является полное имя класса локального интерфейса, перед которым стоит **ejblocal:**. Например, для полного имени интерфейса локального интерфейса com.mycompany.mylocalbusiness предварительно выбранным EJB 3.0 JNDI является:

  ```
  ejblocal:com.mycompany.mylocalbusiness
  ```

  Для удаленного интерфейса com.mycompany.myremotebusiness предварительно выбранным EJB 3.0 JNDI является полное имя интерфейса:

  ```
  com.mycompany.myremotebusiness
  ```

  Привязки приложений EJB 3.0 по умолчанию описываются в следующем расположении: Привязки приложений EJB 3.0 – обзор.

  Integration Designer будет использовать "короткое" имя в качестве расположения JNDI по умолчанию для EJB, использующих программную модель версии 3.0.

  Примечание: Если развернутая ссылка JNDI целевого EJB отличается от расположения привязки JNDI по умолчанию из-за использования пользовательского преобразования или настройки, то целевое имя JNDI должно быть задано правильно. Можно задать имя в Integration Designer до развертывания или для привязки импорта можно изменить имя в административной консоли (после развертывания) для соответствия имени JNDI целевого EJB.

Дополнительная информация о создании привязок EJB приведена в разделе Работа с привязками EJB справочной системы Integration Designer Information Center.

Обработчик данных JAX-WS:

Привязка импорта Enterprise JavaBean (EJB) использует обработчик данных JAX-WS для превращения бизнес-объекта запроса в параметры объекта Java и для превращения возвращенного значения объекта Java в бизнес-объект ответа. Привязка экспорта EJB использует обработчик данных JAX-WS для превращения EJB запросов в бизнес-объекты запросов и для превращения бизнес-объекта ответа в возвращенное значение.

Этот обработчик данных преобразует данные из интерфейса WSDL со спецификацией SCA в целевой интерфейс EJB Java (и наоборот) с помощью Java API для спецификации XML Web Services (JAX-WS) и Java Architecture для спецификации XML Binding (JAXB).

Примечание: В настоящее время поддерживаются только спецификации JAX-WS 2.1.1 и JAXB 2.1.3.

Обработчик данных, указанный на уровне привязки EJB, используется для обработки запросов, ответов, ошибок и исключительных ситуаций во время выполнения.

Примечание: Для ошибок с помощью свойства конфигурации faultBindingType можно определить конкретный обработчик данных для каждой ошибки. Это переопределяет значение, заданное на уровне привязки EJB.

Обработчик данных JAX-WS используется по умолчанию, когда привязка EJB содержит интерфейс WSDL. Этот обработчик данных не может использоваться для преобразования сообщения SOAP, представляющего вызов JAX-WS объекта данных.

Привязка импорта EJB использует обработчик данных для преобразования объекта данных в массив Java Object (Object[]). Во время исходящей связи происходит следующая обработка:

1. Привязка EJB задает ожидаемый тип, ожидаемый элемент и имя целевого метода в BindingContext согласно заданным в WSDL.
2. Привязка EJB вызывает метод преобразования для объекта данных, требующего преобразования данных.
3. Обработчик данных возвращает Object[], представляющий параметры метода (в порядке их определения в методе).
4. Привязка EJB использует Object[] для вызова метода на целевом интерфейсе EJB.

Привязка также подготавливает Object[] к обработке ответа из вызова EJB.

- Первым элементом в Object[] является возвращенное значение из вызова метода Java.
- Последующие значения представляют параметры ввода для метода.

Это требует поддержки типов In/Out и Out параметров.

Для параметров типа Out значения должны возвращаться в объект данных ответа.

Обработчик данных обрабатывает и преобразует значения в Object[] и затем возвращает ответ объекту данных.

Обработчик данных поддерживает xs:AnyType, xs:AnySimpleType и xs:Any наряду с другими типами данных XSD. Для включения поддержки xs:Any используйте **@XmlAnyElement (lax=true)** для свойства EJB (JavaBean) в коде Java, как показано в следующем примере:

```
public class TestType {
 private Object[] object;

 @XmlAnyElement (lax=true)
 public Object[] getObject() {
  return object;
 )

 public void setObject (Object[] object) {
  this.object=object;
 )
)
```

При этом создается объект свойства в TestType и поле xs:any. Значение класса Java, используемое в поле xs:any, должно иметь аннотацию **@XmlAnyElement**. Например, если адресом является класс Java, используемый для заполнения массива объекта, то класс Адрес должен иметь аннотацию **@XmlRootElement**.

Примечание: Для настройки преобразования из типа XSD в типы Java, определенные в спецификации JAX-WS, измените аннотации JAXB в соответствии с требованиями вашего бизнеса. Обработчик данных JAX-WS поддерживает xs:any, xs:anyType и xs:anySimpleType.

Для обработчика данных JAX-WS применяются следующие ограничения:

- Обработчик данных не включает поддержку аннотации **@WebParam** атрибута заголовка.
- Пространство имен для файлов схем бизнес-объектов (файлов XSD) не включает преобразование по умолчанию из имен пакета Java. Аннотация **@XMLSchema** в package-info.java тоже не работает. Единственным путем создания XSD с пространством имен является использование аннотаций **@XmlType** и **@XmlRootElement**. **@XmlRootElement** определяет целевое пространство имен для глобального элемента в типах JavaBean.
- Мастер импорта EJB не создает файлов XSD для несвязанных классов. Версия 2.0 не поддерживает аннотацию **@XmlSeeAlso**, поэтому если дочерний класс не указывается непосредственно из родительского класса, то XSD не создается. Решением этой проблемы является выполнение SchemaGen для каждого дочернего класса.

  SchemaGen – это утилита командной строки (в каталоге *установочный-каталог-WPS*/bin), предусмотренная для создания файлов XSD для данного EJB. Для того чтобы это решение работало, эти XSD необходимо вручную скопировать в модуль.

Селектор ошибок EJB:

Селектор ошибок EJB определяет, приведет ли вызов EJB к ошибке, исключительной ситуации во время выполнения или к успешному ответу.

При обнаружении ошибки селектор ошибок EJB возвращает собственное имя ошибки среде выполнения привязки, чтобы обработчик данных JAX-WS мог преобразовать объект исключительной ситуации в бизнес-объект ошибки.

При успешном ответе (без ошибок) привязка импорта EJB собирает массив объекта Java (Object[]) для возврата значений.

- Первым элементом в Object[] является возвращенное значение из вызова метода Java.
- Последующие значения представляют параметры ввода для метода.

Это требует поддержки типов In/Out и Out параметров.

Для сценариев с исключительной ситуацией привязка собирает Object[], и первый элемент представляет исключительную ситуацию, выброшенную методом.

Селектор ошибок может возвращать любое из следующих значений:

Таблица 64. Возвращаемые значения

Тип	**Возвращаемое значение**	**Описание**
Ошибка	**ResponseType.FAULT**	Возвращается, когда переданный Object[] содержит объект исключительной ситуации.
Исключительная ситуация во время выполнения	**ResponseType.RUNTIME**	Возвращается, если объект исключительной ситуации не совпадает ни с одним из типов исключительных ситуаций, объявленных в методе.
Нормальный ответ	**ResponseType.RESPONSE**	Возвращается во всех остальных случаях.

Если селектор ошибок возвращает значение **ResponseType.FAULT**, то возвращается собственное имя ошибки. Это собственное имя ошибки используется привязкой для определения соответствующего имени ошибки WSDL из модели и для вызова соответствующего обработчика данных ошибок.

Селектор функций EJB:

Привязки EJB используют селектор функций импорта (для исходящей обработки) или селектор функций экспорта (для входящей обработки) для определения вызываемого метода EJB.

Селектор функций импорта

Для исходящей обработки селектор функций импорта извлекает тип метода EJB на основании имени операции, вызванной компонентом SCA, который соединен с импортом EJB. Селектор функций ищет аннотацию @WebMethod на созданном Integration Designer JAX-WS, аннотированном классом Java, для определения имени связанной целевой операции.

- Если аннотация @WebMethod присутствует, то селектор функций использует аннотацию @WebMethod для определения соответствующего преобразования метода Java для метода WSDL.
- Если аннотация @WebMethod отсутствует, то селектор функций предполагает, что имя метода Java совпадает с именем вызванной операции.

Примечание: Этот селектор функций действует только для интерфейса типа WSDL на импорте EJB, но не для интерфейса типа Java на импорте EJB.

Селектор функций возвращает объект java.lang.reflect.Method, который представляет метод интерфейса EJB.

Селектор функций использует массив Java Object (Object[]) для ответа из целевого метода. Первым элементом в Object[] является метод Java с именем WSDL, а вторым элементом Object[] является бизнес-объект ввода.

Селектор функций экспорта

Для входящей обработки селектор функций экспорта извлекает целевой метод, вызываемый из метода Java.

Селектор функций экспорта преобразует имя операции Java, вызванной клиентом EJB, в имя операции в интерфейсе целевого компонента. Имя метода возвращается как строка и обрабатывается средой выполнения SCA в зависимости от типа интерфейса целевого компонента.

Привязки EIS

Привязки информационной системы предприятия (EIS) обеспечивают связь между компонентами SCA и внешней EIS. Эта связь достигается с помощью экспортов и импортов EIS, которые поддерживают адаптеры ресурсов JCA 1.5 и адаптеры Websphere.

Компоненты SCA могут требовать передачи данных в или из внешней EIS. При создании модуля SCA, требующего такой связи включается (кроме компонента SCA) импорт или экспорт с привязкой EIS для связи с конкретной внешней EIS.

Адаптеры ресурсов в IBM Integration Designer используются в контексте импорта или экспорта. Импорт или экспорт разрабатывается с помощью мастера внешней службы и в процессе разработки включается адаптер ресурсов. Импорт EIS, который позволяет приложению вызывать службу в системе EIS, или экспорт EIS, который позволяет приложению в системе EIS вызывать службу, разработанную в IBM Integration Designer, создается с помощью адаптера ресурсов. Например, можно создать импорт с помощью адаптера JD Edwards для вызова службы в системе JD Edwards.

При использовании мастера внешней службы создается информация о привязке EIS. Для добавления или изменения информации о привязке также можно использовать другой инструмент – редактор сборки. Дополнительная информация приведена в разделе Обращение к внешним службам с помощью адаптеров.

После развертывания модуля с привязкой EIS на сервере административную консоль можно использовать для просмотра информации о привязке или для ее настройки.

Обзор привязок EIS:

Привязка EIS (enterprise information system - информационная система предприятия), используемая совместно с адаптером ресурсов JCA, позволяет получать доступ к службам информационной системы предприятия или предоставлять свои службы для использования в EIS.

В следующем примере показано, каким образом модуль SCA с именем ContactSyncModule синхронизирует информацию о контактах между системами Siebel и SAP.

1. Компонент SCA с именем ContactSync отслеживает (с помощью экспорта приложения EIS с именем Siebel Contact) изменения в контактах Siebel.
2. Компонент SCA ContactSync использует приложение SAP (посредством импорта приложения EIS) для обновления информации о контактах SAP.

Поскольку в системах Siebel и SAP контакты хранятся в разных структурах данных, компонент SCA ContactSync должен выполнять преобразование.

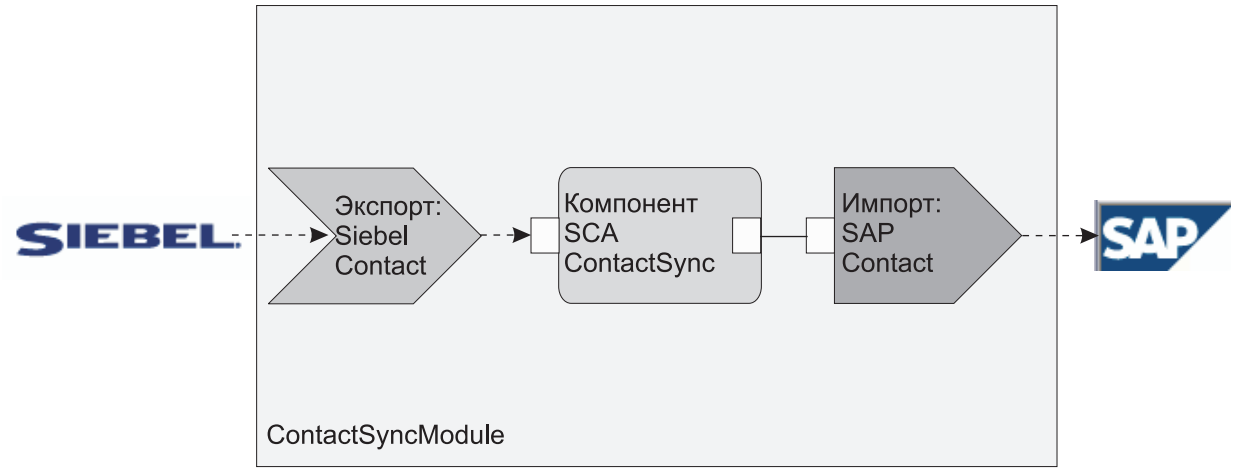


Рисунок 76. Передача данных из системы Siebel в систему SAP

Для экспорта Siebel Contact и импорта SAP Contact настроены соответствующие адаптеры ресурсов.

Основные компоненты привязок EIS:

Импорт EIS является импортом архитектуры компонентов служб (SCA), который позволяет компонентам в модуле SCA использовать приложения EIS, определенные вне модуля SCA. Импорт EIS используется для передачи данных из компонента SCA внешней EIS; экспорт EIS используется для передачи данных из внешней EIS модулю SCA.

Импорты

Ролью импорта EIS является связывание компонентов SCA и внешних систем EIS. Внешние приложения могут рассматриваться как импорт EIS. В этом случае импорт EIS передает данные внешней EIS и может принимать данные в ответ.

Импорт EIS предоставляет компонентам SCA единообразную панель приложений, внешних по отношению к модулю. Это позволяет компонентам связываться с внешней EIS, например SAP, Siebel или PeopleSoft, с помощью совместимой модели SCA.

На стороне клиента импорта имеется интерфейс, предоставляемый приложением импорта EIS, с одним или более методами, каждый из которых получает объекты данных как аргументы и возвращает значения. На стороне реализации имеется общий клиентский интерфейс (CCI), реализуемый адаптером ресурсов.

Динамическая реализация импорта EIS связывает интерфейс стороны клиента с CCI. Импорт преобразует вызов метода на интерфейсе в вызов на CCI.

Привязки создаются на трех уровнях: привязке интерфейса, которая затем использует содержащиеся привязки метода, которые, в свою очередь, используют привязки данных.

Привязка интерфейса связывает интерфейс импорта для соединения с системой EIS, предоставляющей приложение. Это отражает тот факт, что набор приложений, представляемый интерфейсом, предоставляется конкретным экземпляром EIS, а соединение предоставляет доступ к этому экземпляру. Элемент привязки содержит свойства с достаточной информацией для создания соединения (эти свойства являются частью экземпляра javax.resource.spi.ManagedConnectionFactory).

Привязка метода связывает метод с конкретным взаимодействием с системой EIS. Для JCA это взаимодействие характеризуется набором свойств реализации интерфейса javax.resource.cci.InteractionSpec. Элемент взаимодействия привязки метода содержит эти свойства вместе с именем класса, предоставляя таким образом достаточную информацию для осуществления взаимодействия. Привязка метода использует привязки данных, описывающие преобразование аргумента и результат метода интерфейса для представления EIS.

Сценарий выполнения для импорта EIS следующий:

1. Метод на интерфейсе импорта вызывается с помощью программной модели SCA.
2. Запрос, достигающий импорта EIS, содержит имя метода и его аргументы.
3. Импорт сначала создает реализацию привязки интерфейса, а затем, с помощью данных из привязки импорта, создает ConnectionFactory и связывает их. Таким образом, импорт вызывает setConnectionFactory на привязке интерфейса.
4. Создается реализация привязки метода, соответствующая вызываемому методу.
5. Создается и заполняется экземпляр javax.resource.cci.InteractionSpec; затем привязки данных используются для связывания аргументов метода с форматом, понятным адаптеру ресурсов.
6. Интерфейс CCI используется для осуществления взаимодействия.
7. Когда вызов возвращается, привязка данных используется для создания результата вызова, и результат возвращается инициатору.

Экспорты

Ролью экспорта EIS является связывание компонента SCA и внешней EIS. Внешние приложения могут рассматриваться как экспорт EIS. В этом случае внешнее приложение отправляет свои данные в виде периодических уведомлений. Экспорт EIS можно рассматривать как приложение подписки, получающее внешний запрос от EIS. Компонент SCA, который использует экспорт EIS, видит его как локальное приложение.

Экспорт EIS предоставляет компонентам SCA единообразную панель приложений, внешних по отношению к модулю. Это позволяет компонентам связываться с EIS, например SAP, Siebel или PeopleSoft, с помощью совместимой модели SCA.

Экспорт имеет реализацию получателя, получающего запросы от EIS. Получатель реализует интерфейс получателя конкретного адаптера ресурсов. Экспорт также содержит компонент, реализующий интерфейс, предоставляемый EIS через экспорт.

Динамическая реализация экспорта EIS соединяет получателя с компонентом, реализующим интерфейс. Экспорт преобразует запрос EIS в запрос соответствующей операции на компоненте. Привязки создаются на трех уровнях: привязке получателя, которая затем использует содержащуюся привязку внутреннего метода, которая, в свою очередь, используют привязку данных.

Привязка получателя связывает получателя, получающего запросы, с компонентом, предоставляемым через экспорт. Определение экспорта содержит имя компонента; среда выполнения находит его и направляет запросы ему.

Привязка внутреннего метода связывает внутренний метод или тип события, получаемого получателем, с операцией, реализуемой компонентом, предоставленным посредством экспорта. Нет взаимосвязи между методом, вызываемым на получателе, и типом события; все события поступают через один или несколько

методов получателя. Привязка внутреннего метода использует определенный в экспорте селектор функций для извлечения имени внутреннего метода из входящих данных и привязок данных для связывания формата данных EIS с форматом, понятным компоненту.

Сценарий выполнения для экспорта EIS следующий:

1. Запрос EIS запускает вызов метода на реализации получателя.
2. Получатель находит и вызывает экспорт, передавая ему все аргументы вызова.
3. Экспорт создает реализацию привязки получателя.
4. Экспорт создает экземпляр селектора функций и задает его на привязке получателя.
5. Экспорт инициализирует привязки внутренних методов и добавляет их в привязку получателя. Для каждой привязки внутреннего метода также инициализируются привязки данных.
6. Экспорт вызывает привязку получателя
7. Привязка получателя находит экспортированные компоненты и использует селектор функций для извлечения имени внутреннего метода.
8. Это имя используется для поиска привязки внутреннего метода, которая затем вызывает целевой компонент.

Стиль взаимодействия с адаптером позволяет привязке EIS вызывать целевой компонент либо асинхронно (по умолчанию), либо синхронно.

Адаптеры ресурсов

Импорт или экспорт разрабатывается с помощью мастера внешней службы и в процессе разработки включается адаптер ресурсов. Адаптеры, которые поставляются с IBM Integration Designer, используемым для доступа к системам CICS, IMS, JD Edwards, PeopleSoft, SAP и Siebel, предназначаются только для целей разработки и тестирования. Это означает, что они используются для разработки и тестирования приложений.

После развертывания приложения необходимы лицензионные динамические адаптеры для выполнения приложения. Однако при создании службы можно вложить адаптер со службой. Лицензирование адаптера может разрешать использовать вложенный embedded adapter адаптер как лицензионный динамический адаптер. Эти адаптеры согласуются с архитектурой коннекторов Java EE Connector Architecture (JCA 1.5). Открытый стандарт JCA является стандартом Java EE для связи с EIS. JCA предоставляет управляемую среду, то есть Quality of Service (QoS) обеспечивается сервером приложений, который предоставляет управление жизненным циклом и защиту транзакциям. Они также совместимы со спецификацией поиска метаданных предприятия, кроме IBM CICS ECI Resource Adapter и IBM IMS Connector for Java.

Более старый набор адаптеров бизнес-интеграции WebSphere также поддерживается мастером.

Ресурсы EE

Модуль EIS, как модуль SCA, который следует шаблону модуля EIS, можно развернуть на платформе Java EE.

Развертывание модуля EIS на платформе Java EE позволяет получить приложение, которое готово к выполнению, упаковано как файл EAR и развернуто на сервере. Создаются все артефакты и ресурсы Java EE; приложение настраивается и готово к выполнению.

Динамические свойства спецификации взаимодействия и спецификации соединения JCA:

Привязка EIS принимает ввод для спецификаций InteractionSpec и ConnectionSpec, заданный в виде корректно определенного дочернего объекта данных, связанного с полезными данными. Это позволяет взаимодействовать с адаптером ресурсов в режиме запрос-ответ с помощью InteractionSpec или выполнять идентификацию компонента с помощью ConnectionSpec.

javax.cci.InteractionSpec содержит информацию о том, каким образом следует обработать запрос на взаимодействие с адаптером ресурсов. Кроме того, в нем может содержаться информация о ходе взаимодействия после отправки запроса. Такое двустороннее взаимодействие иногда называется *диалогом*.

Привязка EIS ожидает, что полезные данные, которые передаются в качестве аргумента в адаптер ресурсов, содержат дочерний объект с именем **properties**. Этот объект свойств должен содержать пары имя-значение, причем имена свойств спецификации взаимодействия должны быть заданы в определенном формате. Действуют следующие правила форматирования:

- Имена должны начинаться с префикса **IS**, за которым должно следовать имя свойства. Например, если у спецификации взаимодействия есть свойство JavaBeans, которое называется **InteractionId**, то имя этого свойства должно быть указано как **ISInteractionId**.
- Пара имя-значение задает имя и значение простого типа для свойства спецификации взаимодействия.

В этом примере в определении интерфейса указано, что вводом для операции служит объект данных **Account**. Этот интерфейс вызывает приложение привязки импорта EIS для отправки и получения динамического свойства InteractionSpec с именем **workingSet** и значением **xyz**.

Хранящиеся на сервере бизнес-график или бизнес-объекты содержат бизнес-объект **properties**, поддерживающий отправку данных в формате протокола вместе с полезными данными. Это встроенный бизнес-объект **properties**, который не требуется указывать в схеме XML при создании бизнес-объекта. Его можно создать и сразу же использовать. Если на основе схемы XML определены пользовательские типы данных, то укажите элемент **properties**, содержащий ожидаемые пары имя-значение.

```
BOFactory dataFactory = (BOFactory) \
serviceManager.locateService("com/ibm/websphere/bo/BOFactory");
 //Wrapper для интерфейсов с оболочкой doc-lit,
 //перейдите к полезным данным, если doc-lit не используется
 DataObject docLitWrapper = dataFactory.createByElement /
 ("http://mytest/eis/Account", "AccountWrapper");
```

Создать полезные данные.

```
DataObject account = docLitWrapper.createDataObject(0);
  DataObject accountInfo = account.createDataObject("AccountInfo");
  //Настройка полезных данных


  //Создание свойств для динамического взаимодействия

  DataObject properties = account.createDataObject("properties");
```

Задать ожидаемое значение **xyz** для свойства с именем workingSet.

```
properties.setString("ISworkingSet", "xyz");


  //Вызвать службу с аргументом

  Service accountImport = (Service) \
  serviceManager.locateService("AccountOutbound");
  DataObject result = accountImport.invoke("createAccount", docLitWrapper);

  //Получить возвращенное свойство
  DataObject retProperties = result.getDataObject("properties");

  String workingset = retProperties.getString("ISworkingSet");
```

Свойства ConnectionSpec можно использовать для динамической идентификации компонента. При этом действуют указанные выше правила, однако префикс имени свойства должен быть равен **CS** (а не **IS**). Свойства ConnectionSpec нельзя передавать в обоих направлениях. Один объект данных **properties** может содержать свойства IS и CS.

Для применения свойств ConnectionSpec укажите в параметре **resAuth** привязки импорта значение **Application**. Кроме того, адаптер ресурсов должен поддерживать идентификацию компонентов. За дополнительной информацией обратитесь к главе 8 публикации J2EE Connector Architecture Specification.

Внешние клиенты с привязками EIS:

Сервер может обмениваться сообщениями с внешними клиентами через привязки EIS.

Внешний клиент, такой как веб-портал или EIS, должен отправить сообщение модулю SCA на сервере, или же его необходимо вызвать в компоненте на сервере.

Клиент вызывает импорт EIS как и любое другое приложение, используя интерфейс DII или интерфейс Java.

1. Внешний клиент создает экземпляр ServiceManager и выполняет поиск импорта EIS по имени. Результатом поиска является реализация интерфейса службы.
2. Клиент создает входной аргумент - простой объект данных, создаваемый динамически с помощью схемы объекта данных. Это действие выполняется с помощью реализации интерфейса DataFactory объекта Service Data Object.
3. Внешний клиент вызывает EIS и получает необходимые результаты.

Вместо этого клиент может вызвать импорт EIS, используя интерфейс Java.

1. Клиент создает экземпляр ServiceManager и выполняет поиск импорта EIS по имени. Результатом поиска является интерфейс Java импорта EIS.
2. Клиент создает входной аргумент и типизированный объект данных.
3. Клиент вызывает EIS и получает необходимые результаты.

Интерфейс экспорта EIS определяет интерфейс экспортированного компонента SCA, который доступен внешним приложениям EIS. Можно считать, что это тот интерфейс, который внешнее приложение (такое как SAP или PeopleSoft) вызывает через реализацию среды выполнения приложения экспорта EIS.

Экспорт использует привязку EISExportBinding для связывания экспортированных служб с внешним приложением EIS. Это дает возможность настроить приложение, содержащее модуль SCA, для отслеживания запросов службы EIS. Привязка экспорта EIS задает правила преобразования определения входящих событий в том формате, который поддерживается адаптером ресурсов (через интерфейсы Java EE Connector Architecture), в вызовы операций SCA.

Для применения EISExportBinding необходимо, чтобы внешние службы EIS использовали контракты на входящие соединения Java EE Connector Architecture 1.5. При использовании EISExportBinding необходимо, чтобы обработчик данных или привязка данных были заданы на уровне привязки или на уровне метода.

Привязки JMS

Провайдер службы сообщений Java (JMS) включает обмен сообщениями на основании API и программной модели службы сообщений Java. Он предоставляет фабрики соединений JMS для создания целевых объектов JMS и для отправки и получения сообщений.

Привязки JMS можно использовать для взаимодействия с привязкой провайдера шины интеграции служб (Service Integration Bus, SIB). Они совместимы с JMS и JCA 1.5.

Привязки импортов и экспортов JMS обеспечивают модулю архитектуры компонентов служб (SCA) возможность обращаться к и получать сообщения из внешних систем JMS.

Привязки импортов и экспортов JMS обеспечивают интеграцию с приложениями JMS с помощью провайдера SIB JMS на основе JCA 1.5, который включен в WebSphere Application Server. Другие адаптеры ресурсов JMS на основе JCA 1.5 не поддерживаются.

Привязки JMS – обзор:

Привязки JMS обеспечивают взаимодействие между средой архитектуры компонентов служб (SCA) и системами JMS.

Привязки JMS

Основные компоненты привязок импортов и экспортов JMS следующие:

- Адаптер ресурсов: обеспечивает управляемую двухстороннюю связь между модулем SCA и внешними системами JMS
- Соединения: охватывают виртуальное соединение между клиентом и приложением провайдера
- Целевые объекты: используются клиентом для определения целевого объекта сообщений, которые он отправляет, или исходного объекта сообщений, которые он получает
- Идентификационные данные: используются для защиты доступа к привязке

Основные компоненты привязок JMS

Специальные заголовки

Свойства специальных заголовков используются в импортах и экспортах JMS для информирования целевого объекта о том, как обрабатывать сообщение.

Например, TargetFunctionName выполняет преобразование из внутреннего метода в рабочий метод.

Ресурсы EE

При развертывании импортов и экспортов JMS в среду Java EE создаются многие ресурсы Java EE.

`ConnectionFactory`
Используется клиентами для соединения с провайдером JMS.

`ActivationSpec`
Используется импортами для получения ответа на запрос; используется экспортами при настройке конечных точек сообщений, которые представляют получателей сообщений в их взаимодействиях с системой обмена сообщениями.

`Целевые объекты`

- Назначение отправления: в случае импорта – это куда отправляется запрос или исходящее сообщение; в случае экспорта – это получатель, которому будет отправлено ответное сообщение в отсутствие замещающего поля заголовка JMSReplyTo во входящем сообщении.
- Назначение получения: это целевой объект размещения входящего сообщения; в случае импорта – это ответ, а в случае экспорта – это запрос.
- Назначение обратного вызова: целевая система SCA JMS для хранения информации о зависимостях. Не производите чтение или запись в это назначение.

Задача установки создает ConnectionFactory и три объекта назначения. Она также создает ActivationSpec для включения динамического получателя сообщений для прослушивания ответов в назначении получения. Свойства этих ресурсов определяются в файле импорта или экспорта.

Интеграция и адаптеры ресурсов JMS:

Служба сообщений Java (JMS) обеспечивает интеграцию через доступный адаптер ресурсов JMS на основе JCA 1.5. Полная поддержка интеграции JMS предусматривается для адаптера ресурсов JMS шины интеграции служб (SIB).

Используйте провайдера JMS для адаптера ресурсов JCA 1.5, когда требуется интеграция с внешней системой JMS, совместимой с JCA 1.5. Внешние службы, совместимые с JCA 1.5, могут получать сообщения и отправлять сообщения для интеграции с вашими компонентами архитектуры компонентов служб (SCA) с использованием адаптера ресурсов SIB JMS.

Использование адаптеров ресурсов JCA 1.5 других провайдеров не поддерживается.

Привязки импорта и экспорта JMS:

С помощью привязок импорта и экспорта JMS можно настроить модули SCA взаимодействовать со службами, предоставляемыми внешними приложениями JMS.

Привязки импортов JMS

Соединения со связанным провайдером JMS целевых объектов JMS устанавливаются с помощью фабрики соединений JMS. Административные объекты фабрики соединений используются для управления фабриками соединений JMS для провайдера обмена сообщениями по умолчанию.

Взаимодействие с внешними системами JMS включает использование целевых объектов для отправки запросов и получения ответов.

Для привязки импорта JMS поддерживаются два типа сценариев использования в зависимости от типа вызываемой операции:

- Односторонний: Импорт JMS помещает сообщение на целевой объект отправления, настроенный в привязке импорта. Поле replyTo заголовка JMS остается пустым.
- Двухсторонний (запрос-ответ): Импорт JMS помещает сообщение на целевой объект отправления и затем сохраняет ответ, который он получает из компонента SCA.

  Привязку импорта можно настроить (с помощью поля **Схема зависимости ответа** в Integration Designer) на ожидание ИД зависимости ответного сообщения, скопированного из ИД сообщения-запроса (по умолчанию) или из ИД зависимости сообщения-запроса. Привязку импорта также можно настроить на использование временного динамического целевого объекта ответа, чтобы связать ответы с запросами. Временный целевой объект создается для каждого запроса, и импорт использует этот целевой объект для получения ответа.

  Целевой объект получения задается в свойстве заголовка replyTo исходящего сообщения. Получатель сообщения развертывается для прослушивания на целевом объекте получения, и по получении ответа получатель сообщения передает этот ответ назад компоненту.

Как для одностороннего, так и двухстороннего сценария могут быть заданы динамические и статические свойства заголовков. Статические свойства можно задать из привязки метода импорта JMS. Некоторые из этих свойств имеют специальные значения для выполнения SCA JMS.

Важно отметить, что JMS является асинхронной привязкой. Если вызывающий компонент вызывает импорт JMS синхронно (в двухсторонней операции), то вызывающий компонент блокируется до возвращения ответа службой JMS.

рис. 32 на стр. 121 иллюстрирует, как импорт связывается с внешней службой.

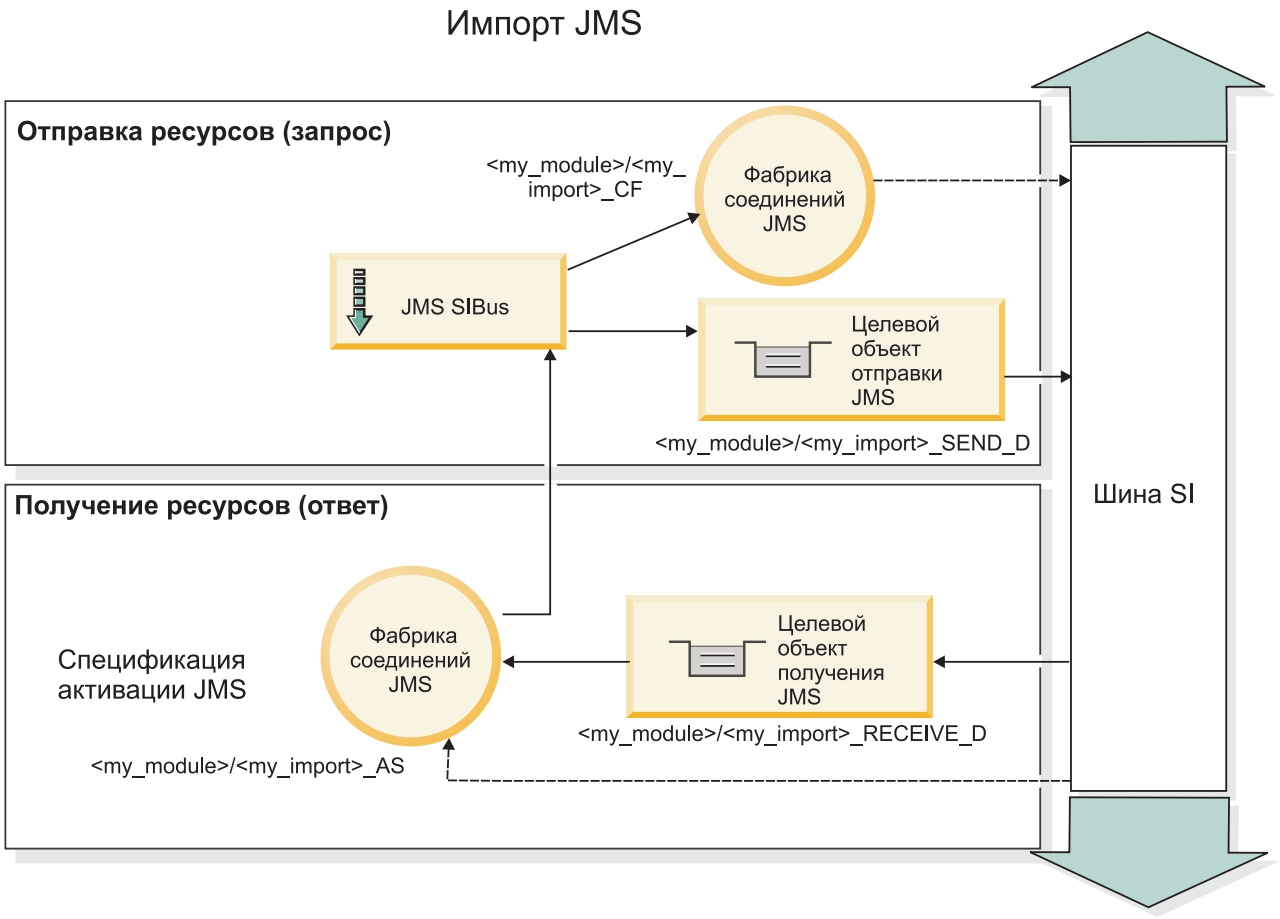


Рисунок 77. Ресурсы привязок импортов JMS

Привязки экспортов JMS

Привязки экспортов JMS обеспечивают средства для модуля SCA по предоставлению служб внешним приложениям JMS.

Соединение, включенное в экспорт JMS, является настраиваемой спецификацией активации.

Экспорт JMS имеет целевые объекты отправления и получения.

- Целевой объект получения – это куда должно быть помещено входящее сообщение для целевого компонента.
- Целевой объект отправления – это куда будет отправлен ответ, если входящее сообщение не переопределило его с помощью свойства заголовка replyTo.

Получатель сообщений развертывается для прослушивания запросов, приходящих на целевой объект получения, указанный в привязке экспорта. Целевой объект, заданный в поле send, используется для отправки ответа на входящий запрос, если вызванный компонент предоставляет ответ. Целевой объект, заданный в поле replyTo входящего сообщения, переопределяет целевой объект, указанный в поле send.

На рисунке рис. 33 на стр. 122 показано, как внешний инициатор запроса связывается с экспортом.

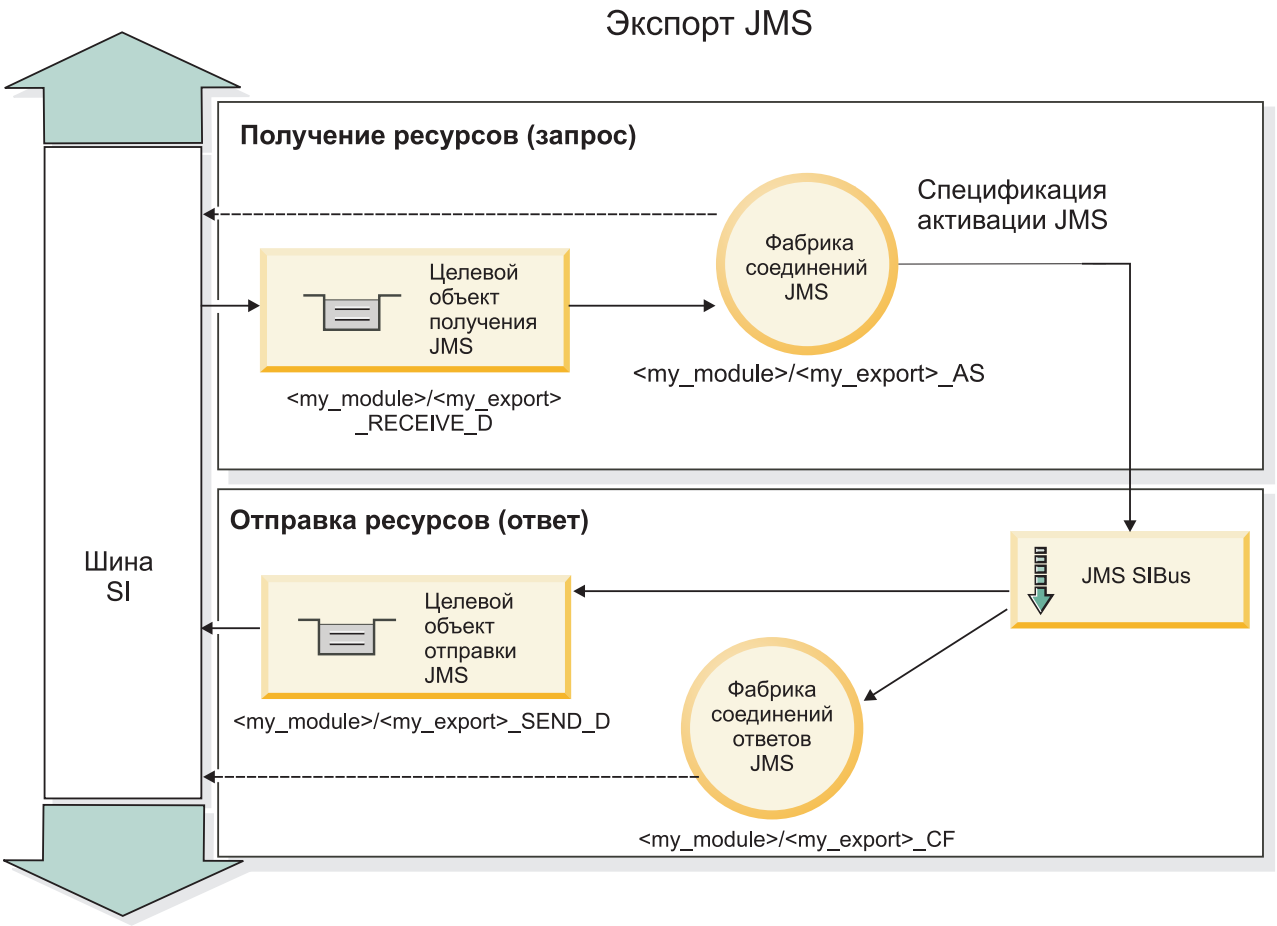


Рисунок 78. Ресурсы привязок экспортов JMS

Заголовки JMS:

Сообщение JMS содержит заголовки двух типов: системный заголовок JMS и несколько свойств JMS. Оба типа заголовков могут быть доступны либо в модуле передачи объекта сообщения службы (SMO), либо с помощью API ContextService.

Системный заголовок JMS

Системный заголовок JMS представляется в SMO элементом JMSHeader, который содержит все поля, типичные для заголовка JMS. Несмотря на то, что они могут изменяться в передаче (или ContextService), некоторые поля системного заголовка JMS, заданные в SMO, не будут передаваться в исходящем сообщении JMS, поскольку они переопределяются системными или статическими значениями.

Основные поля в системном заголовке JMS, которые могут обновляться в передаче (или ContextService), следующие:

- **JMSType** и **JMSCorrelationID** - значения конкретных уже заданных свойств заголовка сообщения
- **JMSDeliveryMode** - значения для режима доставки (постоянные или временные; по умолчанию - постоянные)
- **JMSPriority** - значение приоритета (от 0 до 9; по умолчанию применяется приоритет-JMS-по-умолчанию)

Свойства JMS

Свойства JMS представляются в SMO как записи в списке Свойства. Эти свойства можно добавлять, обновлять или удалять в передаче или с помощью API ContextService.

Свойства также можно задавать статически в привязке JMS. Свойства, которые заданы статически, переопределяют значения (с тем же именем), которые задаются динамически.

Пользовательские свойства, передаваемые из других привязок (например, привязки HTTP), будут выводиться в привязке JMS как свойства JMS.

Параметры распространения заголовков

Распространением системного заголовка и свойств JMS либо из входящего сообщения JMS в последующие компоненты, либо из предыдущих компонентов в исходящее сообщение JMS можно управлять с помощью флага Распространять заголовок протокола на привязке.

Когда флаг Распространять заголовок протокола включен, информация заголовка может передаваться сообщению или целевому компоненту согласно следующему списку:

- Запрос экспорта JMS

  Заголовок JMS, полученный в сообщении, будет распространяться целевым компонентам при помощи контекстной службы. Свойства JMS, полученные в сообщении, будут распространяться целевым компонентам при помощи контекстной службы.
- Ответ экспорта JMS

  Все поля заголовка JMS, заданные в контекстной службе, будут использоваться в исходящем сообщении, если они не переопределяются статическими свойствами, заданными в привязке экспорта JMS. Все свойства, заданные в контекстной службе, будут использоваться в исходящем сообщении, если они не переопределяются статическими свойствами, заданными в привязке экспорта JMS.
- Запрос импорта JMS

  Все поля заголовка JMS, заданные в контекстной службе, будут использоваться в исходящем сообщении, если они не переопределяются статическими свойствами, заданными в привязке импорта JMS. Все свойства, заданные в контекстной службе, будут использоваться в исходящем сообщении, если они не переопределяются статическими свойствами, заданными в привязке импорта JMS.
- Ответ импорта JMS

  Заголовок JMS, полученный в сообщении, будет распространяться целевым компонентам при помощи контекстной службы. Свойства JMS, полученные в сообщении, будут распространяться целевым компонентам при помощи контекстной службы.

Схема зависимостей временного динамического целевого объекта ответа JMS:

Схема зависимостей временного динамического целевого объекта ответа вызывает уникальную динамическую очередь или раздел, создаваемый для каждого отправления запроса.

Статический целевой объект ответа, задаваемый в импорте, используется для получения свойств очереди или раздела временного динамического целевого объекта. Это задается в поле **ReplyTo** запроса, и импорт JMS прослушивает ответы на этом целевом объекте. По получении ответа он снова помещается в очередь статического целевого объекта ответа для асинхронной обработки. Поле **CorrelationID** ответа не используется и не требует определения.

Операционные неполадки

Когда используется временный динамический целевой объект, ответ должен использоваться в той же нити, что и отправленный запрос. Запрос должен отправляться вне глобальной транзакции и должен быть зафиксирован до получения конечной службой и возврата ответа.

Сохранность

Временные динамические очереди являются короткоживущими сущностями и не гарантируют того же уровня сохранности как статическая очередь или раздел. Временная динамическая очередь или раздел не сохраняется после перезапуска сервера, а также и содержащиеся сообщения. После повторного помещения сообщения в очередь статического целевого объекта ответа оно сохраняется в ней в течение времени, определенного в сообщении.

Тайм-аут

Импорт ожидает получения ответа на временном динамическом целевом объекте ответа в течение некоторого фиксированного времени. Этот период времени берется из спецификатора срока действия ответа SCA, если он задан, в противном случае по умолчанию применяется значение 60 секунд. Если время ожидания превышается, то импорт выбрасывает `ServiceTimeoutRuntimeException`.

Внешние клиенты:

Сервер может обмениваться сообщениями с внешними клиентами через привязки JMS.

Внешний клиент (такой как веб-портал или информационная система предприятия) должен отправить сообщение модулю SCA на сервере, или же его необходимо вызвать в компоненте на сервере.

Компоненты экспорта JMS разворачивают получатели сообщений для отслеживания запросов, адресованных целевому получателю, указанному в привязке экспорта. Если вызванное приложение предоставило ответ, то он возвращается получателю, указанному в поле отправки. Благодаря этому внешний клиент может вызывать приложения с привязкой экспорта.

Импорты JMS взаимодействуют с внешними клиентами, отправляя сообщения в очереди JMS или получая сообщения.

Работа со внешними клиентами:

Внешнему клиенту (расположенному вне сервера) может потребоваться взаимодействие с приложением, установленным на сервере.

Рассмотрим очень простой сценарий, в котором внешнему клиенту требуется взаимодействовать с приложением на сервере. На изображении показан обычный простой сценарий.

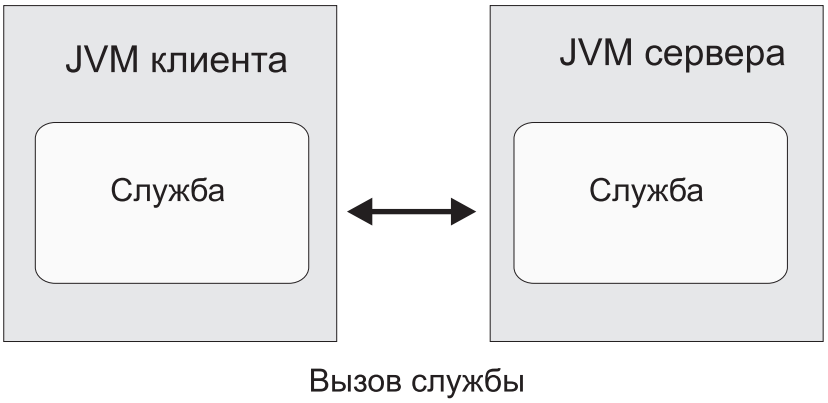


Рисунок 79. Простой сценарий варианта использования: внешний клиент взаимодействует с приложением сервера

Приложение SCA содержит экспорт с привязкой JMS. Таким образом приложение становится доступным для внешних клиентов.

Если внешний клиент расположен на виртуальной машине Java (JVM), которая расположена за пределами сервера, то для установки соединения и обеспечения взаимодействия с экспорт JMS необходимо выполнить некоторые действия. Клиент получает InitialContext с правильными значениями и ищет ресурсы с помощью JNDI. Затем клиент использует клиент спецификации JMS 1.1 для доступа к целевым расположениям и сообщениям, отправляемым и получаемым ими.

Стандартные имена JNDI ресурсов, созданные автоматически средой выполнения, приведены в подразделе о конфигурации в данном разделе. Однако если ресурсы созданы заранее, используйте их имена JNDI.

1. Настройте целевые расположения JMS и фабрику соединений для отправки сообщения.
2. Убедитесь, что контекст JNDI, порт адаптера ресурсов SIB и порт начальной загрузки модуля обмена сообщениями указаны без ошибок.

   Сервер использует некоторые стандартные порты, но если в системе установлено несколько серверов, то во время установки создаются альтернативные порты для того чтобы избежать конфликтов с другими экземплярами сервера. Для того чтобы определить, какие порты использует сервер, можно использовать административную консоль. Перейдите на страницу **Серверы > Серверы приложений > *сервер* > Конфигурация** и выберите **Порты** в разделе **Связь**. Теперь используемый порт можно изменить.
3. Клиент получает исходный контекст с правильными значениями и ищет ресурсы с помощью JNDI.
4. С помощью спецификаций JMS 1.1 клиент получает доступ к целевым расположениям и к сообщениям, отправляемым и получаемым ими.

Устранение неполадок привязок JMS:

Неполадки привязок JMS можно диагностировать и устранять.

Исключительные ситуации реализации

В ответ на различные ошибки реализация импорта и экспорта JMS может возвратить исключительные ситуации одного из двух типов:

- исключительная ситуация бизнес-службы: эта исключительная ситуация выбрасывается, если происходит сбой, указанный в бизнес-интерфейсе службы (тип порта WSDL);
- исключительная ситуация рабочей среды: выбрасывается во всех остальных случаях. В большинстве случаев исключительная ситуация cause будет содержать исходную исключительную ситуацию (JMSException).

  Например, импорт ожидает только одно ответное сообщение для каждого сообщения запроса. Если получено несколько ответов или ответ запоздал (время истечения срока ожидания ответа SCA истекло), то выбрасывается исключительная ситуация рабочей среды. Выполняется откат транзакции, и ответное сообщение удаляется из очереди или обрабатывается диспетчером недоставленных событий.

Основные условия сбоя

Основные условия сбоя привязок JMS определяются семантикой транзакций, конфигурацией поставщика JMS или ссылкой на фактическое поведение других компонентов. В число основных условий сбоя входят следующие.

- Сбой установки соединения с поставщиком JMS или целевым расположением.

  Сбой установки соединения с поставщиком JMS для получения сообщений приведет к сбою запуска обработчика событий сообщений. Это событие будет записано в протокол WebSphere Application Server. Постоянные сообщения останутся в целевом расположении до момента их успешного получения (или истечения срока).

  Сбой установки соединения с поставщиком JMS для отправки исходящих сообщений приведет к откату транзакции, которая управляет отправкой.

- Сбой анализа входящего сообщения или создания исходящего сообщения.

  Сбой связывания данных или обработчика данных приводит к откату транзакции, которая управляет работой.
- Сбой отправки исходящего сообщения.

  Сбой отправки исходящего сообщения приводит к откату соответствующей транзакции.
- Получение нескольких ответных сообщений или непредвиденное запаздывание ответного сообщения.

  Импорт ожидает только одно ответное сообщение для каждого сообщения запроса. Кроме того, допустимый период времени получения ответа определяется спецификатором Время истечения срока ответа SCA в запросе. В случае получения ответа или истечения срока будет удалена запись об отношении. Если ответные сообщения приходят непредвиденно или запаздывают, то выбрасывается исключительная ситуация рабочей среды.
- Исключительная ситуация рабочей среды в случае тайм-аута службы, которая выбрасывается по причине опоздания ответа при использовании схемы зависимостей назначений временного динамического ответа.

  Тайм-аут импорта JMS произойдет через промежуток времени, определенный спецификатором времени истечения срока ответа SCA. Если он не задан, то будет использовано значение по умолчанию, равное 60 секундам.

Сообщения SCA на основе JMS, которые не выводятся в диспетчере событий с ошибками

В случае сбоя сообщений SCA, полученных по причине сбоя взаимодействия с JMS, ожидается, что они будут показаны в диспетчере недоставленных сообщений. Если эти сообщения не выводятся в диспетчере недоставленных событий, убедитесь, что максимальное число сбоев доставки, заданное для базового назначения SIB назначения JMS, больше **1**. Если указать значение **2** или большее значение, то взаимодействие с диспетчером недоставленных событий во время вызовов SCA для привязок JMS будет возможным.

Обработка исключительных ситуаций:

В конфигурации привязки настраивается способ обработки исключительных ситуаций, сгенерированных обработчиками данных или привязками данных. На действия системы при возникновении такой исключительной ситуации также влияет тип потока передачи.

В обработчике данных или привязке данных, вызванных из привязки, может возникнуть неполадка. Например, обработчик данных может получить сообщение с поврежденными данными или попытаться прочитать сообщение в неверном формате.

Способ обработки таких исключительных ситуаций в привязке зависит от того, каким образом реализован обработчик данных или привязка данных. Рекомендуется, чтобы привязка данных генерировала исключительную ситуацию **DataBindingException**.

При возникновении любой исключительной ситуации времени выполнения, включая **DataBindingException**, выполняются следующие действия:

- Если поток передачи настроен для выполнения транзакций, то по умолчанию в администраторе событий со сбоем сохраняется сообщение JMS , которое можно обработать или удалить вручную.

  Примечание: В привязке можно настроить такой режим исправления, чтобы сообщение не сохранялось в администраторе событий со сбоем, а вместо этого выполнялся откат.
- Если поток передачи не поддерживает транзакции, то исключительная ситуация регистрируется в протоколе, а сообщение удаляется.

В случае обработчика данных ситуация полностью аналогична. Поскольку обработчик данных вызывается из привязки данных, любая исключительная ситуация обработчика данных инкапсулируется в исключительную ситуацию привязки данных. По этой причине об исключительной ситуации **DataHandlerException** сообщается как о **DataBindingException**.

Базовые привязки JMS

Базовые привязки JMS обеспечивают связь с внешними провайдерами, совместимыми с JMS 1.1. Работа базовых привязок JMS аналогична привязкам JMS.

Служба, предоставляемая через привязку JMS, позволяет модулю архитектуры компонентов служб (SCA) делать вызовы или получать сообщения от внешних систем. Этой системой может быть внешняя система JMS.

Базовая привязка JMS обеспечивает интеграцию с провайдерами JMS, не совместимыми с JCA 1.5, которые поддерживают JMS 1.1 и реализуют необязательный JMS Application Server Facility. Базовая привязка JMS поддерживает таких провайдеров JMS (включая Oracle AQ, TIBCO, SonicMQ, WebMethods и BEA WebLogic), которые не поддерживают JCA 1.5, но поддерживают Application Server Facility спецификации JMS 1.1. Встроенный в WebSphere провайдер JMS (SIBJMS), который является провайдером JCA 1.5 JMS, не поддерживается этой привязкой; в случае использования этого провайдера используйте “Привязки JMS” на стр. 118.

Используйте эту базовую привязку при интеграции с системой на основе JMS, не совместимой с JCA 1.5, в среде SCA. Затем целевые внешние приложения могут получать и отправлять сообщения для интеграции с компонентом SCA.

Базовые привязки JMS – обзор:

Базовые привязки JMS являются привязками JMS не JCA, которые обеспечивают взаимодействие между средой архитектуры компонентов служб (SCA) и системами JMS, которые совместимы с JMS 1.1 и которые реализуют необязательный JMS Application Server Facility.

Общие привязки JMS

Основные аспекты базовых привязок импорта и экспорта JMS включают следующие:

- Порт получателя: позволяет провайдерам JMS не на основе JCA получать и отправлять сообщения управляемому сообщениями EJB (MDB)
- Соединения: охватывают виртуальное соединение между клиентом и приложением провайдера
- Целевые объекты: используются клиентом для определения целевого объекта сообщений, которые он отправляет, или исходного объекта сообщений, которые он получает
- Идентификационные данные: используются для защиты доступа к привязке

Базовые привязки импортов JMS

Базовые привязки импортов JMS позволяют компонентам в пределах модуля SCA связываться со службами, предоставляемыми внешними провайдерами JMS, не совместимыми с JCA 1.5.

Обеспечивающей соединение частью импорта JMS является фабрика соединений. Фабрика соединений, являющаяся объектом, который клиент использует для установления соединения с провайдером, включает набор параметров конфигурации соединений, определяемый администратором. Каждая фабрика соединений является экземпляром интерфейса ConnectionFactory, QueueConnectionFactory или TopicConnectionFactory.

Взаимодействие с внешними системами JMS включает использование целевых объектов для отправки запросов и получения ответов.

Для базовой привязки импорта JMS поддерживаются два типа сценариев использования в зависимости от типа вызываемой операции:

- Односторонний: базовый импорт JMS помещает сообщение на целевой объект отправления, настроенный в привязке импорта. В поле replyTo заголовка JMS не отправляется ничего.

- Двухсторонний (запрос-ответ): Базовый импорт JMS помещает сообщение на целевой объект отправления и затем сохраняет ответ, который он получает из компонента SCA.

  Целевой объект получения задается в свойстве заголовка replyTo исходящего сообщения. Управляемый сообщениями EJB (MDB) развертывается для прослушивания на целевом объекте получения, и по получении ответа MDB передает этот ответ назад компоненту.

  Привязку импорта можно настроить (с помощью поля **Схема зависимости ответа** в Integration Designer) на ожидание ИД зависимости ответного сообщения, скопированного из ИД сообщения-запроса (по умолчанию) или из ИД зависимости сообщения-запроса.

Как для одностороннего, так и двухстороннего сценария могут быть заданы динамические и статические свойства заголовков. Статические свойства можно задать из привязки метода базового импорта JMS. Некоторые из этих свойств имеют специальные значения для выполнения SCA JMS.

Важно отметить, что базовая JMS является асинхронной привязкой. Если вызывающий компонент вызывает базовый импорт JMS синхронно (в двухсторонней операции), то вызывающий компонент блокируется до возвращения ответа службой JMS.

рис. 35 на стр. 128 иллюстрирует, как импорт связывается с внешней службой.

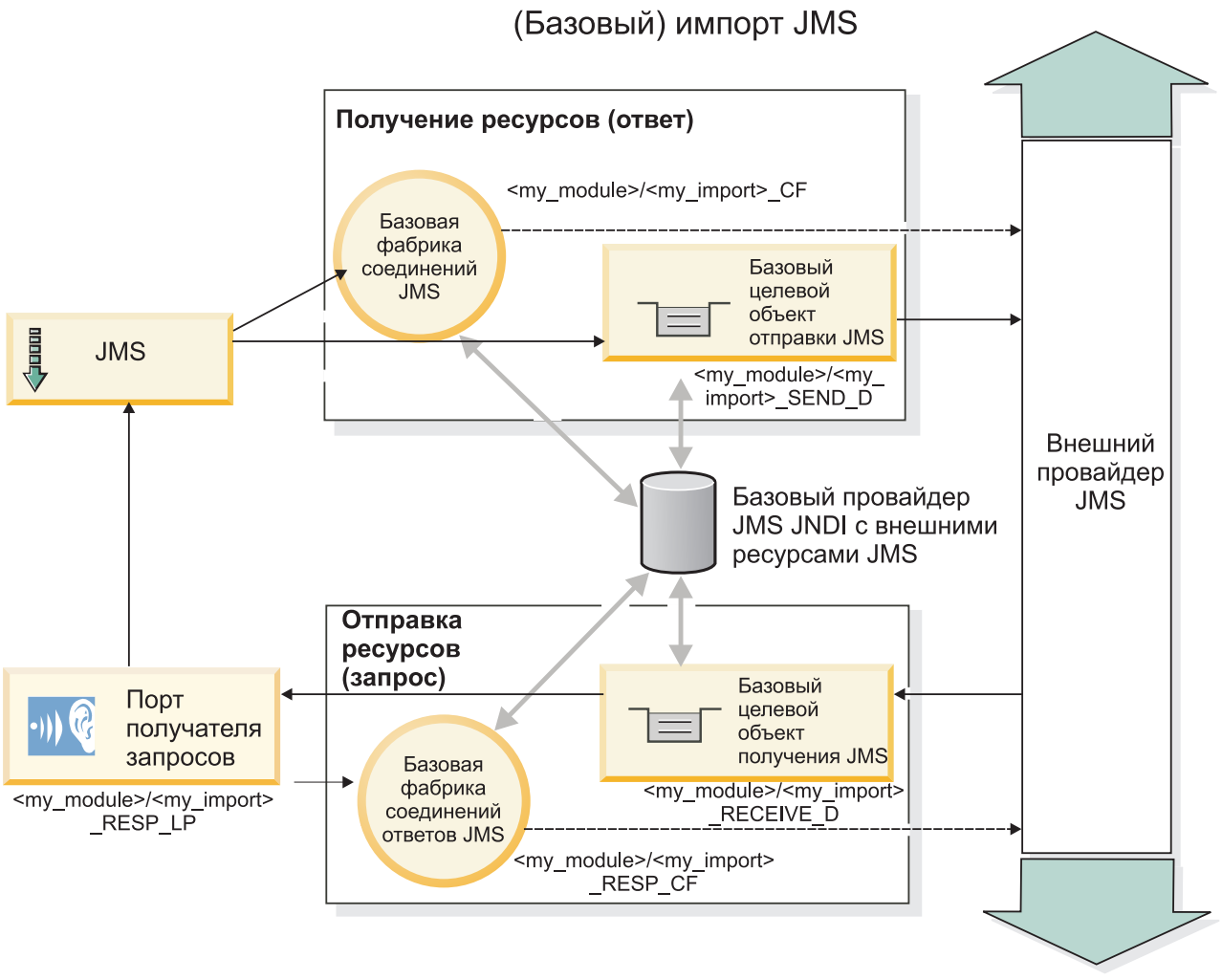


Рисунок 80. Ресурсы привязок базовых импортов JMS

Базовые привязки экспортов JMS

Базовые привязки экспортов JMS обеспечивают средства для модуля SCA по предоставлению служб внешним приложениям JMS.

Обеспечивающей соединение частью экспорта JMS являются ConnectionFactory и ListenerPort.

Общий экспорт JMS имеет целевые объекты отправления и получения.

- Целевой объект получения – это куда должно быть помещено входящее сообщение для целевого компонента.
- Целевой объект отправления – это куда будет отправлен ответ, если входящее сообщение не переопределило его с помощью свойства заголовка replyTo.

Развертывается MDB прослушивания запросов, приходящих на целевой объект получения, указанный в привязке экспорта.

- Целевой объект указывается в поле send и используется для отправки ответа на входящий запрос, если вызванный компонент предоставляет ответ.
- Целевой объект, заданный в поле replyTo входящего сообщения, переопределяет целевой объект, указанный в поле send.
- Для сценариев запрос/ответ привязку импорта можно настроить (с помощью поля **Схема зависимости ответа** в Integration Designer) на ожидание ответа для копирования ИД сообщения-запроса в поле ИД зависимости ответного сообщения (по умолчанию), или ответ может скопировать ИД зависимости запроса в поле ИД зависимости ответного сообщения.

На рисунке рис. 36 на стр. 130 показано, как внешний инициатор запроса связывается с экспортом.

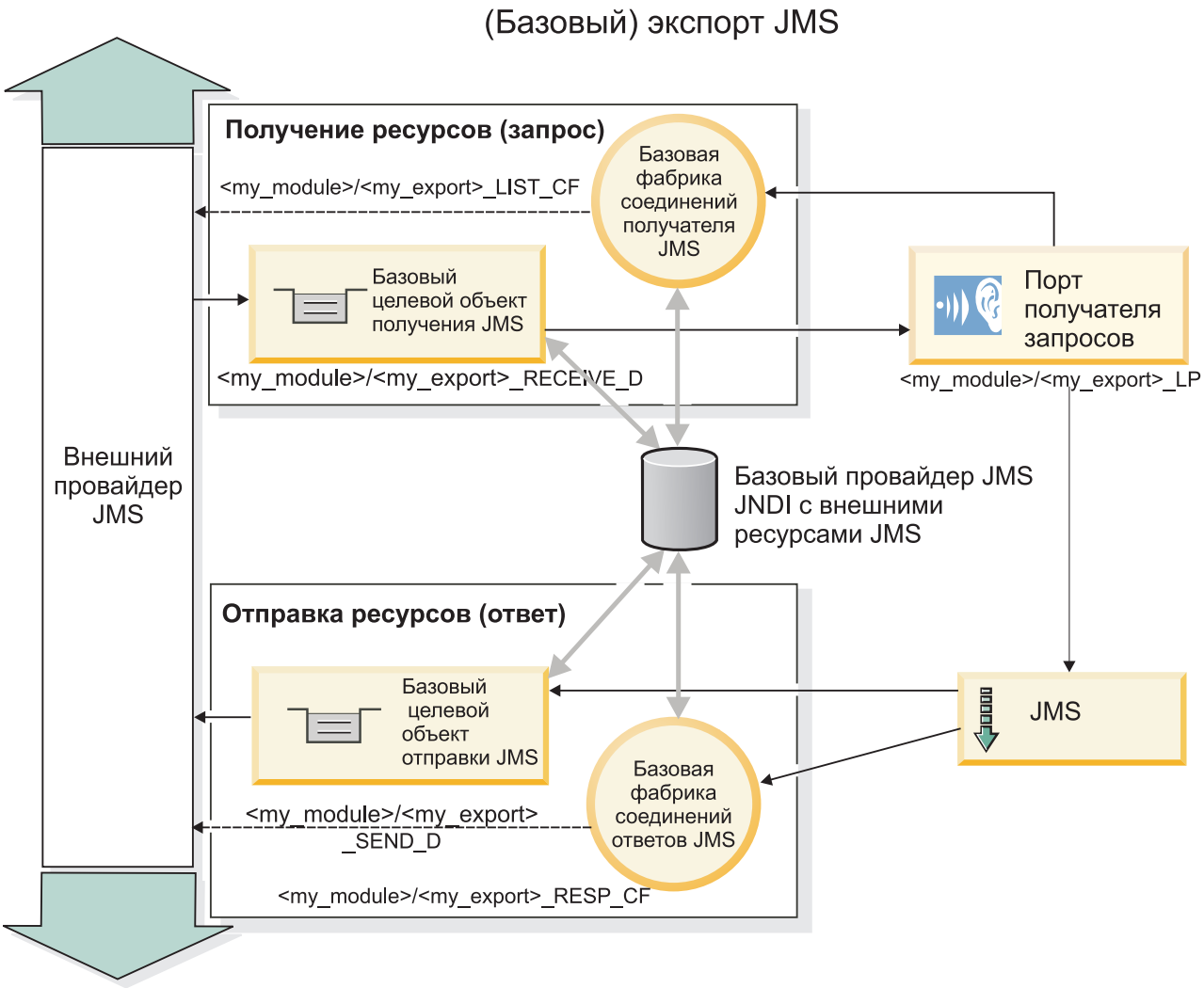


Рисунок 81. Ресурсы базовых привязок экспортов JMS

Основные компоненты привязок Generic JMS:

Компоненты привязки импорта и экспорта Generic JMS соответствуют компонентам встроенных привязок импортов WebSphere JMS и MQ JMS. Основные компоненты включают определения заголовков и доступ к существующим ресурсам Java EE. Однако, ввиду их общего характера, опции соединения с конкретным провайдером JMS отсутствуют, и способность этой привязки создавать ресурсы во время развертывания и установки ограничена.

Базовые импорты

Аналогично приложению импорта MQ JMS, реализация Общая JMS асинхронна и поддерживает три вызова: односторонний, двухсторонний (также называемый запрос-ответ) и обратный вызов.

При развертывании импорта JMS также развертывается управляемый сообщениями EJB (MDB), предоставляемый средой выполнения. MDB прослушивает ответы на сообщение-запрос. MDB связывается (прослушивает) назначением, передаваемым с запросом в поле заголовка replyTo сообщения JMS.

Базовые экспорты

Привязки базовых экспортов JMS отличаются от привязок экспортов EIS обработкой возврата результата. Базовый экспорт JMS явно отправляет ответ в назначение replyTo, указанное во входящем сообщении. Если оно не указано, то используется назначение отправления.

При развертывании базового экспорта JMS также развертывается управляемый сообщениями EJB (MDB, отличающийся от используемого для базовых импортов JMS). Он прослушивает входящие запросы в назначении получения и затем направляет запросы на обработку средой выполнения SCA.

Специальные заголовки

Свойства специальных заголовков используются в базовых импортах и экспортах JMS для информирования целевой привязки о том, как обрабатывать сообщение.

Например, свойство TargetFunctionName используется селектором функций по умолчанию для определения имени операции, вызываемой в интерфейсе экспорта.

Примечание: Привязка импорта может быть настроена на задание имени операции в заголовке TargetFunctionName для каждой операции.

Ресурсы EE

При развертывании привязки JMS в среду Java EE создаются многие ресурсы Java EE.

- Порт получателя для прослушивания назначения получения (ответа) (только при двусторонней связи) для импортов и назначения получения (запроса) для экспортов
- Базовая фабрика соединений JMS для outboundConnection (импорт) и inboundConnection (экспорт)
- Базовое назначение JMS для назначений отправления (импорт) и получения (экспорт, только для двусторонней связи)
- Базовая фабрика соединений JMS для responseConnection (только для двусторонней связи и необязательно; в противном случае outboundConnection используется для импортов, а inboundConnection – для экспортов)
- Базовое назначение JMS для назначения получения (импорт) и отправления (экспорт) (только для двухсторонней связи)
- Назначение JMS обратных вызовов провайдера обмена сообщениями по умолчанию, используемое для доступа к назначению очереди SIB (только для двухсторонней связи)
- Фабрика соединений JMS обратных вызовов провайдера обмена сообщениями по умолчанию, используемая для доступа к назначению JMS обратных вызовов (только для двухсторонней связи)
- Назначение обратных вызовов SIB, используемое для хранения информации о сообщении-запросе для использования во время обработки ответов (только для двухсторонней связи)

Задача установки создает ConnectionFactory, три назначения и ActivationSpec из информации в файлах импорта и экспорта.

Базовые заголовки JMS:

Базовые заголовки JMS - это объекты данных службы (SDO), содержащие все свойства сообщения базового JMS. Это могут быть свойства из входящего сообщения или свойства, применяемые к исходящему сообщению.

Сообщение JMS содержит заголовки двух типов: системный заголовок JMS и несколько свойств JMS. Заголовки обоих типов можно получить из модуля передачи, расположенного в объекте сообщения службы (SMO), или с помощью API ContextService.

Следующие свойства статически настраиваются в methodBinding:

- JMSType
- JMSCorrelationID

- JMSDeliveryMode
- JMSPriority

Привязка базового JMS поддерживает динамическое изменение заголовков и свойств JMS, как и другие привязки JMS и JMS MQ.

Некоторые поставщики базового JMS ограничивают набор свойств, которые разрешено настраивать в приложении, а также возможные сочетания этих свойств. За дополнительной информацией обратитесь к документации по продукту другого поставщика. В methodBinding было добавлено свойство ignoreInvalidOutboundJMSProperties, позволяющее передавать информацию о любых исключительных ситуациях.

Заголовки и свойства сообщения базового JMS используются только в том случае, если включен базовый коммутатор привязок SCDL-SCA. Если этот коммутатор включен, то передается информация о контексте. По умолчанию этот коммутатор включен. Для того чтобы запретить передачу информации о контексте, измените значение на **false**.

Если передача контекста разрешена, то информацию из заголовков разрешается передавать в сообщение или в целевой компонент. Для включения или выключения передачи информации о контексте укажите значение **true** или **false** в атрибуте contextPropagationEnabled привязки импорта или экспорта. Например:

```
<esbBinding xsi:type="eis:JMSImportBinding" contextProgagationEnabled="true">
```

Значение по умолчанию равно **true**.

Устранение неполадок привязок Generic JMS:

Неполадки привязок Generic JMS можно диагностировать и устранять.

Исключительные ситуации реализации

В ответ на различные ошибки реализация импорта и экспорта Generic JMS может возвратить исключительные ситуации одного из двух типов:

- исключительная ситуация бизнес-службы: эта исключительная ситуация выбрасывается, если происходит сбой, указанный в бизнес-интерфейсе службы (тип порта WSDL);
- исключительная ситуация рабочей среды: выбрасывается во всех остальных случаях. В большинстве случаев исключительная ситуация cause будет содержать исходную исключительную ситуацию (JMSException).

Устранение неполадок истечения срока сообщений Generic JMS

Сообщение запроса от поставщика JMS имеет время истечения срока.

Время истечения срока запроса означает, что поставщик JMS помечает сообщение запроса как устаревшее по прошествии времени, указанного в JMSExpiration сообщения запроса. Как и другие привязки JMS, привязка Generic JMS обрабатывает истечение срока запроса, указав в сообщении обратного вызова, которое отправлено импортом, такое же время истечения срока, как и в исходящем запросе. Уведомление об истечении срока сообщения обратного вызова указывает, что сообщение запроса устарело и клиент должен быть оповещен путем выбрасывания исключительной ситуации бизнес-службы.

Однако если целевое расположение обратного вызова перенесено на поставщик другой фирмы, этот тип истечения срока запроса не поддерживается.

Время истечения срока ответа означает, что поставщик JMS помечает сообщение ответа как устаревшее по прошествии времени, указанного в JMSExpiration сообщения ответа.

Истечение срока ответа для общей привязки JMS не поддерживается, потому что точное поведение поставщика JMS другой фирмы при истечении срока является неопределенным. Однако можно проверить, не истек ли срок ответа, в случае и во время его получения.

Для исходящих сообщений значение JMSExpiration вычисляется по времени ожидания и по значениям requestExpiration, если они заданы в asyncHeader.

Устранение неполадок фабрики соединений Generic JMS

Когда в поставщике Generic JMS определяются определенные типы фабрик соединений, во время попытки запуска приложения может быть выведено сообщение об ошибке. Для того чтобы избежать этой неполадки, можно изменить внешнюю фабрику соединений.

Во время запуска приложения может быть выведено следующее сообщение об ошибке:

```
JMSConnectionFactory порта получения запросов MDB не совпадает
с типом JMSDestination
```

Эта неполадка может возникнуть во время определения внешних фабрик соединения. В частности, исключительная ситуация может быть выброшена в случае создания JMS 1.0.2 Topic Connection Factory вместо JMS 1.1 (unified) Connection Factory (то есть, той, которая способна поддерживать связь как в двухточечном режиме, так и в режиме публикации-подписки).

Для устранения этой неполадки выполните следующие действия:

1. откройте поставщик Generic JMS, который используете.
2. замените указанную JMS 1.0.2 Topic Connection Factory на JMS 1.1 (unified) Connection Factory.

При запуске приложения с указанной JMS 1.1 Connection Factory сообщение об ошибке больше выводиться не будет.

Сообщения SCA на основе Generic JMS, которые не выводятся в диспетчере событий с ошибками

В случае сбоя сообщений SCA, полученных по причине сбоя взаимодействия с общей JMS, ожидается, что они будут показаны в диспетчере недоставленных сообщений. Если эти сообщения не выводятся в диспетчере недоставленных событий, убедитесь, что значение свойства максимального числа попыток базового порта получения запросов больше 1. Если указать значение 1 или большее значение, то взаимодействие с диспетчером недоставленных событий во время вызовов SCA для привязок общей JMS будет возможным.

Обработка исключительных ситуаций:

В конфигурации привязки настраивается способ обработки исключительных ситуаций, сгенерированных обработчиками данных или привязками данных. На действия системы при возникновении такой исключительной ситуации также влияет тип потока передачи.

В обработчике данных или привязке данных, вызванных из привязки, может возникнуть неполадка. Например, обработчик данных может получить сообщение с поврежденными данными или попытаться прочитать сообщение в неверном формате.

Способ обработки таких исключительных ситуаций в привязке зависит от того, каким образом реализован обработчик данных или привязка данных. Рекомендуется, чтобы привязка данных генерировала исключительную ситуацию **DataBindingException**.

В случае обработчика данных ситуация полностью аналогична. Поскольку обработчик данных вызывается из привязки данных, любая исключительная ситуация обработчика данных инкапсулируется в исключительную ситуацию привязки данных. По этой причине об исключительной ситуации **DataHandlerException** сообщается как о **DataBindingException**.

При возникновении любой исключительной ситуации времени выполнения, включая **DataBindingException**, выполняются следующие действия:

- Если поток передачи настроен для выполнения транзакций, то по умолчанию в администраторе событий со сбоем сохраняется сообщение JMS, которое можно обработать или удалить вручную.

  Примечание: В привязке можно настроить такой режим исправления, чтобы сообщение не сохранялось в администраторе событий со сбоем, а вместо этого выполнялся откат.
- Если поток передачи не поддерживает транзакции, то исключительная ситуация регистрируется в протоколе, а сообщение удаляется.

В случае обработчика данных ситуация полностью аналогична. Поскольку обработчик данных вызывается из привязки данных, любая исключительная ситуация обработчика данных инкапсулируется в исключительную ситуацию привязки данных. По этой причине об исключительной ситуации **DataHandlerException** сообщается как о **DataBindingException**.

Привязки WebSphere MQ JMS

Привязка WebSphere MQ JMS обеспечивает интеграцию с внешними приложениями, которые используют провайдера WebSphere MQ на основе JMS.

Привязки экспортов и импортов WebSphere MQ JMS используются непосредственно для интеграции с внешними системами JMS или MQ JMS из вашей серверной среды. Это исключает необходимость в использовании компонента MQ Link или Client Link шины интеграции служб.

Когда компонент взаимодействует со службой WebSphere MQ на основе JMS через импорт, привязка импорта WebSphere MQ JMS использует целевой объект, которому будут отправляться данные, и целевой объект, на котором может быть получен ответ. Преобразование данных в и из сообщения JMS осуществляется через Edge Component привязки обработчика данных или данных JMS.

Когда модуль SCA предоставляет службу клиентам WebSphere MQ JMS, привязка экспорта WebSphere MQ JMS использует целевой объект, на котором может быть получен запрос и которому может быть отправлен ответ. Преобразование данных в и из сообщения JMS осуществляется через привязку обработчика данных или данных JMS.

Селектор функций обеспечивает преобразование в операцию, вызываемую на целевом компоненте.

Привязки WebSphere MQ JMS – обзор:

Привязка WebSphere MQ JMS обеспечивает интеграцию с внешними приложениями, использующими провайдер WebSphere MQ JMS.

Административные задачи WebSphere MQ

Предполагается, что системный администратор WebSphere MQ создаст основной WebSphere MQ Queue Manager, который будут использовать привязки WebSphere MQ JMS, перед выполнением приложения, содержащего эти привязки.

Привязки импортов WebSphere MQ JMS

Импорт WebSphere MQ JMS позволяет компонентам в пределах вашего модуля SCA соединяться со службами, предоставляемыми провайдерами WebSphere MQ на основе JMS. Необходимо использовать поддерживаемую версию WebSphere MQ. Подробные требования к аппаратному и программному обеспечению можно найти на страницах поддержки IBM.

Для привязок импортов WebSphere MQ JMS поддерживаются два типа сценариев использования в зависимости от типа вызываемой операции:

- Односторонний: Импорт WebSphere MQ JMS помещает сообщение на целевой объект отправления, настроенный в привязке импорта. В поле replyTo заголовка JMS не отправляется ничего.
- Двухсторонний (запрос-ответ): Импорт WebSphere MQ JMS помещает сообщение на целевой объект отправления.

  Целевой объект получения задается в поле заголовка replyTo. Управляемый сообщениями EJB (MDB) развертывается для прослушивания на целевом объекте получения, и по получении ответа MDB передает этот ответ назад компоненту.

  Привязку импорта можно настроить (с помощью поля **Схема зависимости ответа** в Integration Designer) на ожидание ИД зависимости ответного сообщения, скопированного из ИД сообщения-запроса (по умолчанию) или из ИД зависимости сообщения-запроса.

Как для одностороннего, так и двухстороннего сценария могут быть заданы динамические и статические свойства заголовков. Статические свойства можно задать из привязки метода импорта JMS. Некоторые из этих свойств имеют специальные значения для выполнения SCA JMS.

Важно отметить, что WebSphere MQ JMS является асинхронной привязкой. Если вызывающий компонент вызывает импорт WebSphere MQ JMS синхронно (в двухсторонней операции), то вызывающий компонент блокируется до возвращения ответа службой JMS.

рис. 37 на стр. 135 иллюстрирует, как импорт связывается с внешней службой.

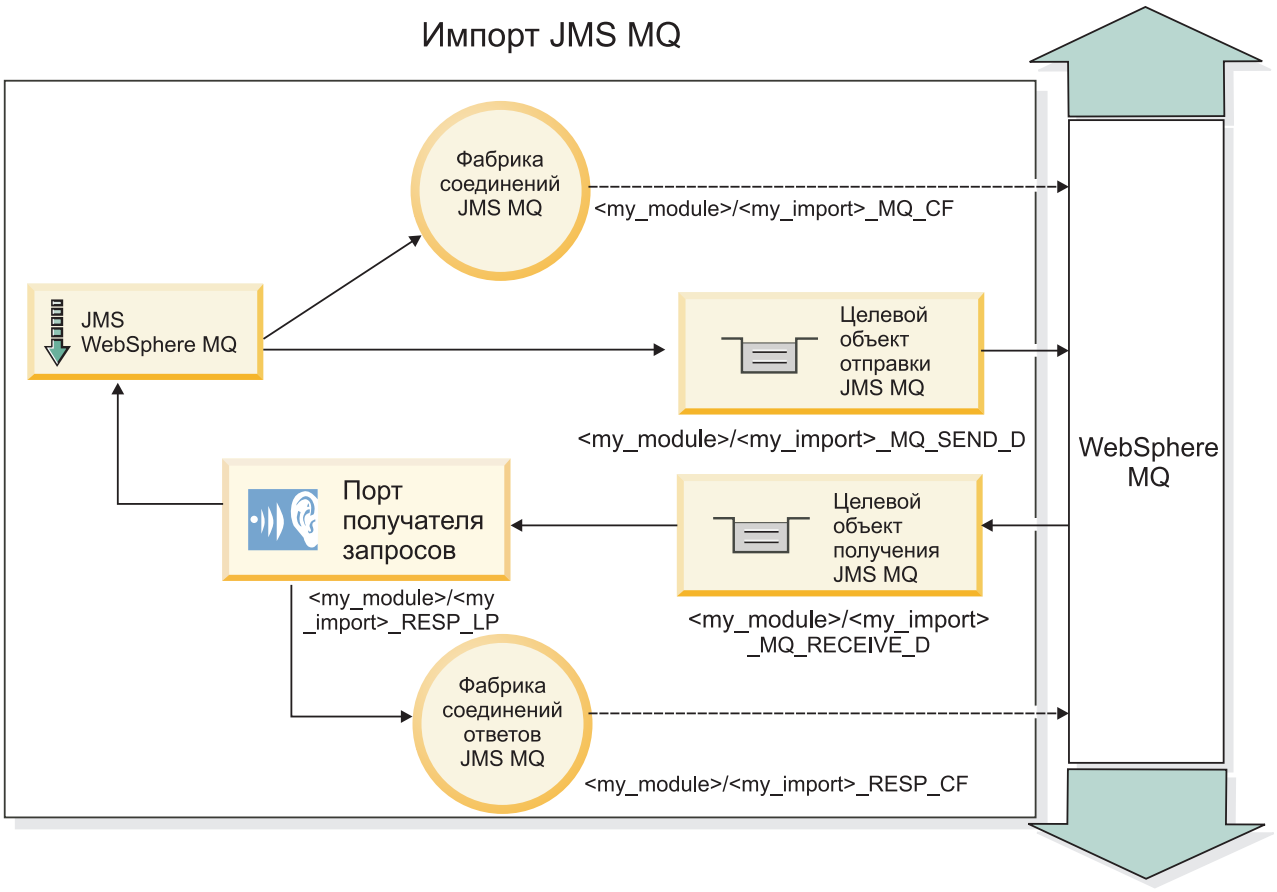


Рисунок 82. Ресурсы привязки импорта WebSphere MQ JMS

Привязки экспортов WebSphere MQ JMS

Привязка экспорта WebSphere MQ JMS обеспечивает средства для модуля SCA по предоставлению служб внешним приложениям JMS на провайдере WebSphere MQ на основе JMS.

Развертывается MDB прослушивания запросов, приходящих на целевой объект получения, указанный в привязке экспорта. Целевой объект, заданный в поле send, используется для отправки ответа на входящий запрос, если вызванный компонент предоставляет ответ. Целевой объект, заданный в поле replyTo ответного сообщения, переопределяет целевой объект, указанный в поле send.

На рисунке рис. 38 на стр. 136 показано, как внешний инициатор запроса связывается с экспортом.

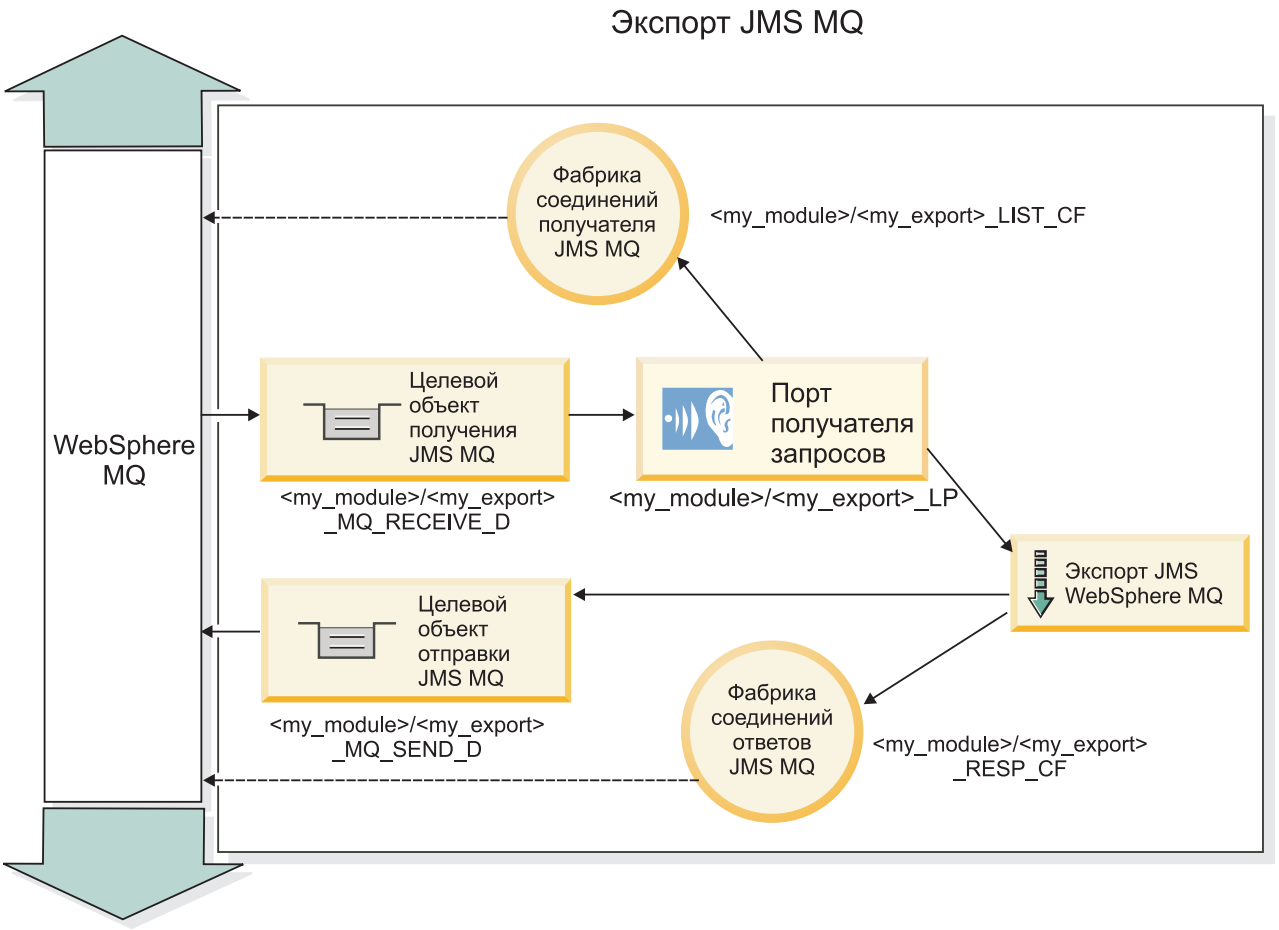


Рисунок 83. Ресурсы привязки экспорта WebSphere MQ JMS

Примечание: рис. 37 на стр. 135 и рис. 38 на стр. 136 иллюстрируют, как приложение из предыдущей версии IBM Business Process Manager связывается с внешней службой. В случае приложений, развернутых для IBM Business Process Manager версии 7.0, используется спецификация активации вместо порта получателя и фабрики соединений.

Основные компоненты привязок WebSphere MQ JMS:

Основные компоненты привязок WebSphere MQ JMS включают заголовки, артефакты Java EE и созданные ресурсы Java EE.

Заголовки

Заголовок сообщения JMS содержит несколько уже заданных полей, содержащих значения, используемые как клиентами, так и провайдерами для идентификации и маршрутизации сообщений. Свойства привязок можно использовать для настройки этих заголовков с использованием постоянных значений, а также заголовки можно определить динамически во время выполнения.

JMSCorrelationID
Указывает на связанное сообщение. Обычно в этом поле задается строка идентификатора сообщения, на которое требуется ответ.

TargetFunctionName
Этот заголовок используется одним из предоставленных селекторов функций для идентификации вызываемой операции. Задание свойства заголовка TargetFunctionName JMS в сообщениях, отправляемых в экспорт JMS, позволяет использовать этот селектор функций. Свойство может быть задано непосредственно в приложениях-клиентах JMS или при соединении импорта с привязкой JMS с таким экспортом. В этом случае привязка импорта JMS должна быть настроена на задание заголовка TargetFunctionName для каждой операции в интерфейсе на имя операции.

Схемы зависимостей

Привязки WebSphere MQ JMS предусматривают разные схемы зависимостей, которые используются для определения того, как сообщения-запросы связаны с ответными сообщениями.

RequestMsgIDToCorrelID
JMSMessageID копируется в поле JMSCorrelationID. Это значение по умолчанию.

RequestCorrelIDToCorrelID
JMSCorrelationID копируется в поле JMSCorrelationID.

Ресурсы EE

При развертывании импорта MQ JMS в среду Java EE создаются многие ресурсы Java EE.

Параметры

Фабрика соединений MQ
Используется клиентами для соединения с провайдером MQ JMS.

Фабрика соединений ответов
Используется средой выполнения SCA MQ JMS, когда назначения отправления и получения относятся к разным администраторам очередей.

Спецификация активации
Спецификация активации MQ JMS связывается с одним или несколькими управляемыми сообщениями EJB и обеспечивает конфигурацию, необходимую им для получения сообщений.

Назначения

- Назначение отправления:
  - Импорты: Куда отправляется запрос или исходящее сообщение.
  - Экспорты: Куда будет отправлено ответное сообщение в отсутствие замещающего поля заголовка JMSReplyTo входящего сообщения.
- Назначение получения:
  - Импорты: Куда должен быть размещен ответ или входящее сообщение.
  - Экспорты: Куда должно быть размещено входящее сообщение или запрос.

Заголовки JMS:

Сообщение JMS содержит заголовки двух типов: системный заголовок JMS и несколько свойств JMS. Оба типа заголовков могут быть доступны либо в модуле передачи объекта сообщения службы (SMO), либо с помощью API ContextService.

Системный заголовок JMS

Системный заголовок JMS представляется в SMO элементом JMSHeader, который содержит все поля, типичные для заголовка JMS. Несмотря на то, что они могут изменяться в передаче (или ContextService), некоторые поля системного заголовка JMS, заданные в SMO, не будут передаваться в исходящем сообщении JMS, поскольку они переопределяются системными или статическими значениями.

Основные поля в системном заголовке JMS, которые могут обновляться в передаче (или ContextService), следующие:

- **JMSType** и **JMSCorrelationID** - значения конкретных уже заданных свойств заголовка сообщения
- **JMSDeliveryMode** - значения для режима доставки (постоянные или временные; по умолчанию - постоянные)
- **JMSPriority** - значение приоритета (от 0 до 9; по умолчанию применяется приоритет-JMS-по-умолчанию)

Свойства JMS

Свойства JMS представляются в SMO как записи в списке Свойства. Эти свойства можно добавлять, обновлять или удалять в передаче или с помощью API ContextService.

Свойства также можно задавать статически в привязке JMS. Свойства, которые заданы статически, переопределяют значения (с тем же именем), которые задаются динамически.

Пользовательские свойства, передаваемые из других привязок (например, привязки HTTP), будут выводиться в привязке JMS как свойства JMS.

Параметры распространения заголовков

Распространением системного заголовка и свойств JMS либо из входящего сообщения JMS в последующие компоненты, либо из предыдущих компонентов в исходящее сообщение JMS можно управлять с помощью флага Распространять заголовок протокола на привязке.

Когда флаг Распространять заголовок протокола включен, информация заголовка может передаваться сообщению или целевому компоненту согласно следующему списку:

- Запрос экспорта JMS

  Заголовок JMS, полученный в сообщении, будет распространяться целевым компонентам при помощи контекстной службы. Свойства JMS, полученные в сообщении, будут распространяться целевым компонентам при помощи контекстной службы.
- Ответ экспорта JMS

  Все поля заголовка JMS, заданные в контекстной службе, будут использоваться в исходящем сообщении, если они не переопределяются статическими свойствами, заданными в привязке экспорта JMS. Все свойства, заданные в контекстной службе, будут использоваться в исходящем сообщении, если они не переопределяются статическими свойствами, заданными в привязке экспорта JMS.
- Запрос импорта JMS

  Все поля заголовка JMS, заданные в контекстной службе, будут использоваться в исходящем сообщении, если они не переопределяются статическими свойствами, заданными в привязке импорта JMS. Все свойства, заданные в контекстной службе, будут использоваться в исходящем сообщении, если они не переопределяются статическими свойствами, заданными в привязке импорта JMS.

- Ответ импорта JMS

  Заголовок JMS, полученный в сообщении, будет распространяться целевым компонентам при помощи контекстной службы. Свойства JMS, полученные в сообщении, будут распространяться целевым компонентам при помощи контекстной службы.

Внешние клиенты:

Сервер может отправлять сообщения или принимать сообщения от внешних клиентов с помощью привязок WebSphere MQ JMS.

Внешний клиент (например, веб-портал или информационная система организации) может отправить сообщение компоненту SCA в приложении при помощи экспорта или он может быть вызван компонентом SCA в приложении при помощи импорта.

Привязка экспорта WebSphere MQ JMS развертывает управляемые сообщениями EJB (MDB) для прослушивания запросов, приходящих на целевой объект получения, указанный в привязке экспорта. Целевой объект указывается в поле send и используется для отправки ответа на входящий запрос, если вызванное приложение предоставляет ответ. Таким образом, внешний клиент может вызывать приложения через привязку экспорта.

Импорты WebSphere MQ JMS привязываются к внешним клиентам и могут доставлять им сообщения. Эти сообщения могут требовать или могут не требовать ответа от внешнего клиента.

Дополнительную информацию о взаимодействии с внешними клиентами с помощью WebSphere MQ можно найти в справочной системе WebSphere MQ Information Center.

Устранение неполадок привязок WebSphere MQ JMS:

Неполадки привязок WebSphere MQ JMS можно диагностировать и устранять.

Исключительные ситуации реализации

В ответ на различные ошибки реализация импорта и экспорта MQ JMS может возвратить исключительные ситуации одного из двух типов:

- исключительная ситуация бизнес-службы: эта исключительная ситуация выбрасывается, если происходит сбой, указанный в бизнес-интерфейсе службы (тип порта WSDL);
- исключительная ситуация рабочей среды: выбрасывается во всех остальных случаях. В большинстве случаев исключительная ситуация cause будет содержать исходную исключительную ситуацию (JMSException).

  Например, импорт ожидает только одно ответное сообщение для каждого сообщения запроса. Если получено несколько ответов или ответ запоздал (время истечения срока ожидания ответа SCA истекло), то выбрасывается исключительная ситуация рабочей среды. Выполняется откат транзакции, и ответное сообщение удаляется из очереди или обрабатывается диспетчером недоставленных событий.

Сообщения SCA на основе WebSphere MQ JMS, которые не выводятся в диспетчере событий с ошибками

В случае сбоя сообщений SCA, полученных при взаимодействии с WebSphere MQ JMS, ожидается, что они будут показаны в диспетчере недоставленных сообщений. Если эти сообщения не выводятся в диспетчере недоставленных событий, убедитесь, что значение свойства максимального числа попыток базового порта получения запросов больше **1**. Если указать значение **1** или большее значение, то взаимодействие с диспетчером недоставленных событий во время вызовов SCA для привязок MQ JMS будет возможным.

Сценарии неправильного употребления: сравнение с привязками WebSphere MQ

Привязка WebSphere MQ JMS предназначена для взаимодействия с приложениями JMS, развернутых на WebSphere MQ, которая предоставляет сообщения в соответствии с моделью сообщений JMS. Однако импорт и экспорт WebSphere MQ изначально предназначены для взаимодействия со стандартными приложениями WebSphere MQ и предоставляют передачам доступ ко всему содержимому тела сообщения WebSphere MQ.

Следующие сценарии следует компоновать с использованием привязки WebSphere MQ JMS, а не WebSphere MQ.

- Вызов объекта EJB, управляемого сообщениями JMS (MDB) из модуля SCA, если MDB развернут на поставщике WebSphere MQ JMS. Используйте импорт WebSphere MQ JMS.
- Разрешение вызова модуля SCA из сервлета компонента Java EE или EJB с использованием JMS. Используйте экспорт WebSphere MQ JMS.
- Транзитная передача содержимого JMS MapMessage через WebSphere MQ. Используйте экспорт и импорт WebSphere MQ в комбинации с соответствующим обработчиком или привязкой данных.

Существуют ситуации, в которых ожидается взаимодействие привязки WebSphere MQ с привязкой WebSphere MQ JMS. В частности, в случае стыковки приложений WebSphere MQ, использующих Java EE, с приложениями, не использующими Java EE, используйте экспорт WebSphere MQ и импорт WebSphere MQ JMS (или наоборот) в комбинации с соответствующими привязками данных или модулями передачи как по отдельности, так и совместно.

Обработка исключительных ситуаций:

В конфигурации привязки настраивается способ обработки исключительных ситуаций, сгенерированных обработчиками данных или привязками данных. На действия системы при возникновении такой исключительной ситуации также влияет тип потока передачи.

В обработчике данных или привязке данных, вызванных из привязки, может возникнуть неполадка. Например, обработчик данных может получить сообщение с поврежденными данными или попытаться прочитать сообщение в неверном формате.

Способ обработки таких исключительных ситуаций в привязке зависит от того, каким образом реализован обработчик данных или привязка данных. Рекомендуется, чтобы привязка данных генерировала исключительную ситуацию **DataBindingException**.

В случае обработчика данных ситуация полностью аналогична. Поскольку обработчик данных вызывается из привязки данных, любая исключительная ситуация обработчика данных инкапсулируется в исключительную ситуацию привязки данных. По этой причине об исключительной ситуации **DataHandlerException** сообщается как о **DataBindingException**.

При возникновении любой исключительной ситуации времени выполнения, включая **DataBindingException**, выполняются следующие действия:

- Если поток передачи настроен для выполнения транзакций, то по умолчанию в администраторе событий со сбоем сохраняется сообщение JMS, которое можно обработать или удалить вручную.

  Примечание: В привязке можно настроить такой режим исправления, чтобы сообщение не сохранялось в администраторе событий со сбоем, а вместо этого выполнялся откат.
- Если поток передачи не поддерживает транзакции, то исключительная ситуация регистрируется в протоколе, а сообщение удаляется.

В случае обработчика данных ситуация полностью аналогична. Поскольку обработчик данных вызывается из привязки данных, любая исключительная ситуация обработчика данных инкапсулируется в

исключительную ситуацию привязки данных. По этой причине об исключительной ситуации **DataHandlerException** сообщается как о **DataBindingException**.

Привязки WebSphere MQ

Привязка WebSphere MQ обеспечивает возможность соединения архитектуры компонентов служб (SCA) с приложениями WebSphere MQ.

Привязки экспортов и импортов WebSphere MQ JMS используются непосредственно для интеграции с системой на основе WebSphere MQ из вашей серверной среды. Это исключает необходимость в использовании компонента MQ Link или Client Link шины интеграции служб.

Когда компонент взаимодействует со службой WebSphere MQ через импорт, привязка импорта WebSphere MQ использует очередь, в которую будут отправляться данные, и очередь, из которой может быть получен ответ.

Когда модуль SCA предоставляет службу клиентам WebSphere MQ, привязка экспорта WebSphere MQ использует очередь, из которой может быть получен запрос и в которую может быть отправлен ответ. Селектор функций обеспечивает преобразование в операцию, вызываемую на целевом компоненте.

Преобразование данных полезной нагрузки в и из сообщения MQ выполняется через обработчик данных тела MQ или привязку данных. Преобразование данных заголовка в и из сообщения MQ выполняется через привязку данных заголовка MQ.

Информация о поддерживаемых версиях WebSphere MQ приведена на веб-странице подробных системных требований.

Привязки WebSphere MQ – обзор:

Привязка WebSphere MQ обеспечивает интеграцию со стандартными приложениями на основе MQ.

Административные задачи WebSphere MQ

Предполагается, что системный администратор WebSphere MQ создаст основной WebSphere MQ Queue Manager, который будут использовать привязки WebSphere MQ, перед выполнением приложения, содержащего эти привязки.

Административные задачи WebSphere

Необходимо задать свойство **Путь к стандартной библиотеке** адаптера ресурсов MQ в Websphere для версии WebSphere MQ, поддерживаемой сервером, и перезапустить сервер. Этим обеспечивается использование библиотек поддерживаемой версии WebSphere MQ. Подробные требования к аппаратному обеспечению приведены на страницах поддержки IBM.

Привязки импортов WebSphere MQ

Привязка импорта WebSphere MQ позволяет компонентам в пределах вашего модуля SCA соединяться со службами, предоставляемыми внешними приложениями на основе WebSphere MQ. Необходимо использовать поддерживаемую версию WebSphere MQ. Подробные требования к аппаратному обеспечению приведены на страницах поддержки IBM.

Взаимодействие с внешними системами WebSphere MQ включает использование очередей для отправки запросов и получения ответов.

Для привязки импорта WebSphere MQ поддерживаются два типа сценариев использования в зависимости от типа вызываемой операции:

- Односторонний: Импорт WebSphere MQ помещает сообщение в очередь, настроенную в поле **Очередь целевого объекта отправления** привязки импорта. В поле replyTo заголовка MQMD не отправляется ничего.
- Двухсторонний (запрос-ответ): Импорт WebSphere MQ помещает сообщение в очередь, настроенную в поле **Очередь целевого объекта отправления**

  Очередь получения задается в поле заголовка replyTo MQMD. Управляемый сообщениями EJB (MDB) развертывается для прослушивания очереди получения, и по получении ответа MDB передает этот ответ назад компоненту.

  Привязку импорта можно настроить (с помощью поля **Схема зависимости ответа**) на ожидание ИД зависимости ответного сообщения, скопированного из ИД сообщения-запроса (по умолчанию) или из ИД зависимости сообщения-запроса.

Важно отметить, что WebSphere MQ является асинхронной привязкой. Если вызывающий компонент вызывает импорт WebSphere MQ синхронно (в двухсторонней операции), то вызывающий компонент блокируется до возвращения ответа службой WebSphere MQ.

рис. 39 на стр. 142 иллюстрирует, как импорт связывается с внешней службой.

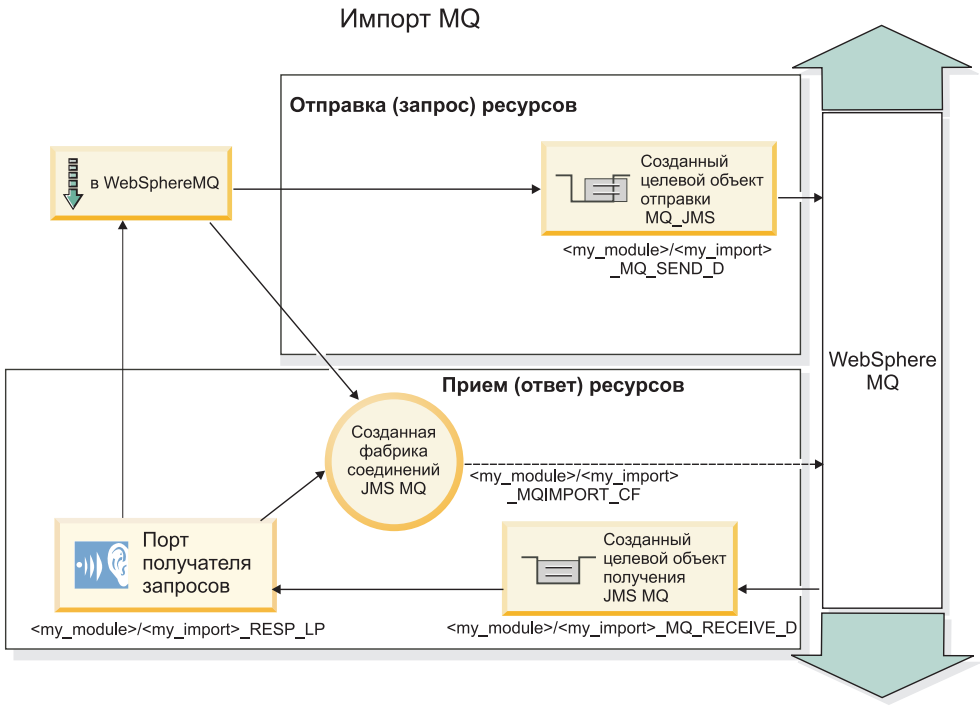


Рисунок 84. Ресурсы привязки импорта WebSphere MQ

Привязки экспортов WebSphere MQ

Привязка экспорта WebSphere MQ обеспечивает средства для модуля SCA по предоставлению служб внешним приложениям на основе WebSphere MQ.

Развертывается MDB прослушивания запросов, приходящих в **очередь целевого объекта получения**, указанную в привязке экспорта. Очередь, заданная в поле **Очередь целевого объекта отправления**, используется для отправки ответа на входящий запрос, если вызванный компонент предоставляет ответ. Очередь, заданная в поле replyTo ответного сообщения, переопределяет очередь, указанную в поле **Очередь целевого объекта отправления**.

На рисунке рис. 40 на стр. 143 показано, как внешний инициатор запроса связывается с экспортом.

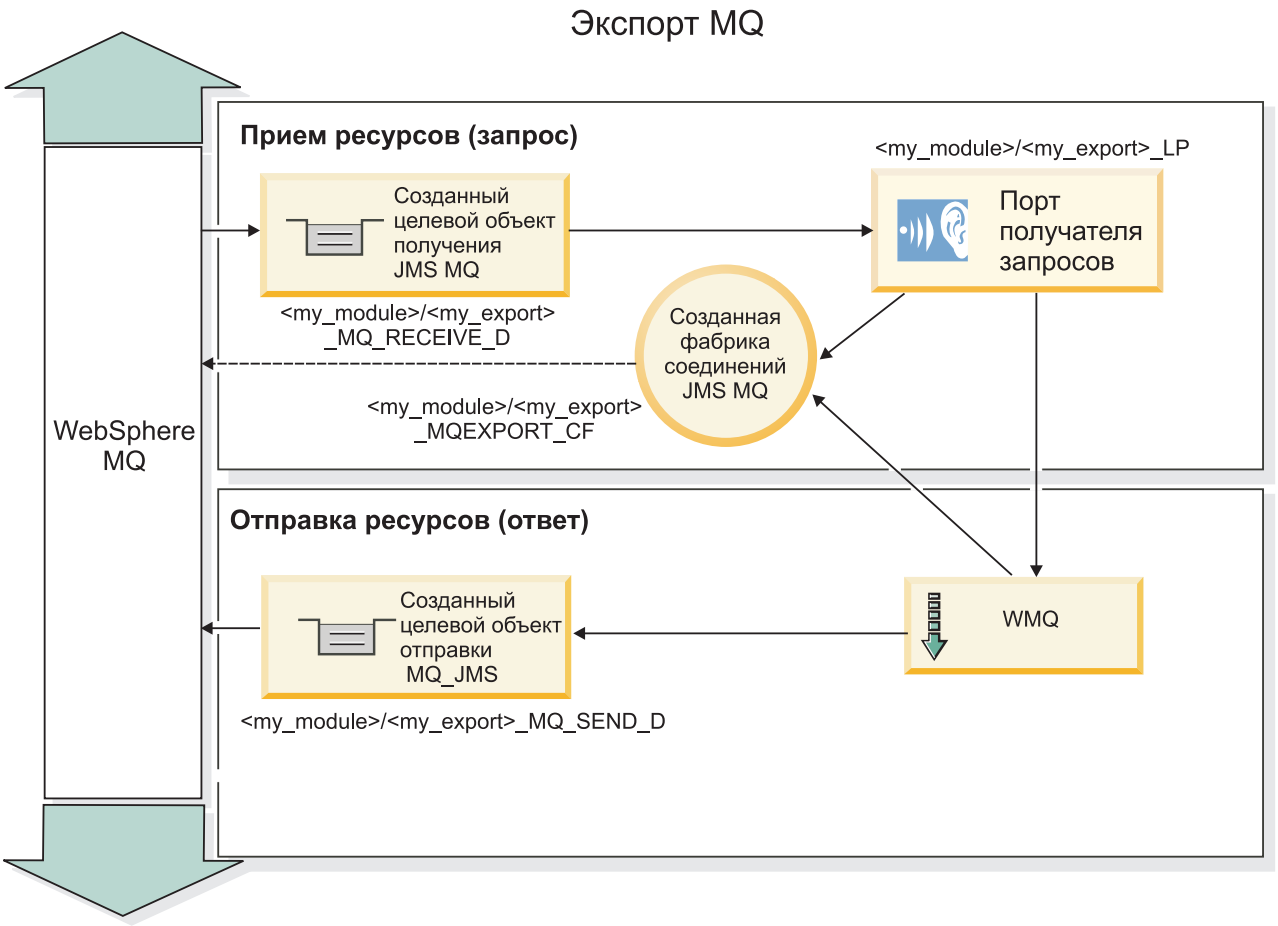


Рисунок 85. Ресурсы привязки экспорта WebSphere MQ

Примечание: рис. 39 на стр. 142 и рис. 40 на стр. 143 иллюстрируют, как приложение из предыдущей версии IBM Business Process Manager связывается с внешней службой. В случае приложений, развернутых для IBM Business Process Manager версии 7.x или более поздней, используется спецификация активации вместо порта получателя и фабрики соединений.

Основные компоненты привязки WebSphere MQ:

Основные компоненты привязки WebSphere MQ включают заголовки, артефакты Java EE и созданные ресурсы Java EE.

Схемы зависимостей

Приложение запроса/ответа WebSphere MQ может использовать один из многих способов связывания ответных сообщений с запросами, создаваемых полями MQMD MessageID и CorrelID. В большинстве случаев инициатор разрешает администратору очередей выбрать MessageID и ожидает копирования отвечающего приложения в CorrelID ответа. В большинстве случаев инициатору и отвечающему приложению косвенно известно, как метод зависимости используется. Иногда отвечающее приложение ставит различные флаги в поле Отчет ответа, которые описывают, как обрабатывать эти поля.

Привязки экспортов для сообщений WebSphere MQ могут настраиваться на следующие опции:

Опции MsgId ответа:

`Создать MsgID`
Позволяет администратору очередей выбрать уникальный MsgId для ответа (по умолчанию).

Копировать из MsgID ответа
Копирует поле MsgId из поля MsgId ответа.

Копировать из сообщения SCA
Задает то MsgId, которое переносится в заголовках WebSphere MQ ответного сообщения SCA или позволяет администратору очередей определить новый Id, если это значение не существует.

Опции согласно отчету
Проверяет поле Отчет MQMD в запросе на указание, как обрабатывать MsgId. Опции MQRO_NEW_MSG_ID и MQRO_PASS_MSG_ID поддерживаются и действуют аналогично опциям Создать MsgId и Копировать из MsgID ответа.

Опции CorrelId ответа:

Копировать из MsgID ответа
Копирует поле CorrelId из поля MsgId ответа (по умолчанию).

Копировать из CorrelID ответа
Копирует поле CorrelId из поля CorrelId ответа.

Копировать из сообщения SCA
Задает то CorrelId, которое переносится в заголовках WebSphere MQ ответного сообщения SCA или оставляет его пустым, если это значение не существует.

Опции согласно отчету
Проверяет поле Отчет MQMD в запросе на указание, как обрабатывать CorrelId. Опции MQRO_COPY_MSG_ID_TO_CORREL_ID и MQRO_PASS_CORREL_ID поддерживаются и действуют аналогично опциям Копировать из MsgID запроса и Копировать из CorrelID ответа.

Привязки импортов для сообщений WebSphere MQ могут настраиваться на следующие опции:

Опции MsgId запроса:

Создать MsgID
Позволяет администратору очередей выбрать уникальный MsgId для запроса (по умолчанию).

Копировать из сообщения SCA
Задает то MsgId, которое переносится в заголовках WebSphere MQ сообщения-запроса SCA или позволяет администратору очередей определить новый Id, если это значение не существует.

Опции зависимости ответа:

CorrelID ответа копируется из MsgId
Ожидается, что ответное сообщение имеет поле CorrelId согласно MsgId запроса (по умолчанию).

MsgID ответа копируется из MsgId
Ожидается, что ответное сообщение имеет поле MsgId согласно MsgId запроса.

CorrelID ответа копируется из CorrelId
Ожидается, что ответное сообщение имеет поле CorrelId согласно CorrelId запроса.

Ресурсы EE

При развертывании привязки WebSphere MQbinding в среду Java EE создаются многие ресурсы Java EE.

Параметры

Фабрика соединений MQ
Используется клиентами для установления соединения с провайдером WebSphere MQ.

Фабрика соединений ответов
Используется средой выполнения SCA MQ, когда назначения отправления и получения относятся к разным администраторам очередей.

Спецификация активации
Спецификация активации MQ JMS связывается с одним или несколькими управляемыми сообщениями EJB и обеспечивает конфигурацию, необходимую им для получения сообщений.

Целевые объекты

- Назначение отправления: Куда отправляется запрос или исходящее сообщение (импорт); куда отправляется ответное сообщение (экспорт) в отсутствие замещающего поля заголовка MQMD ReplyTo во входящем сообщении.
- Назначение получения: Куда должен быть размещен ответ/запрос или входящее сообщение.

Заголовки WebSphere MQ:

В заголовках WebSphere MQ применяются определенные соглашения для преобразования в сообщения архитектуры компонентов служб (SCA).

Сообщения WebSphere MQ включают системный заголовок (MQMD), ноль или более других заголовков MQ (системных или пользовательских) и тело сообщения. При наличии нескольких заголовков сообщения важен их порядок.

Каждый заголовок содержит информацию, описывающую структуру следующего заголовка. MQMD описывает первый заголовок.

Как обрабатываются заголовки MQ

Для обработки заголовков MQ применяется связывание данных заголовка MQ. Следующие заголовки поддерживаются автоматически:

- MQRFH
- MQRFH2
- MQCIH
- MQIIH

Заголовки, начинающиеся с **MQH**, обрабатываются по-другому. Особые поля заголовка не обрабатываются, а остаются в виде набора байтов.

Для обработки прочих заголовков MQ можно подготовить пользовательское связывание данных заголовка MQ.

Как осуществляется доступ к заголовкам MQ

Доступ к заголовкам MQ в продукте может осуществляться следующими способами:

- Посредством объекта сообщения службы (SMO) в посреднике
- Посредством ContextService API

В системе заголовки MQ представляются элементом MQHeader SMO. MQHeader - это контейнер данных заголовка, который расширяет MQControl, но при этом содержит элемент anyType. Он содержит MQMD, MQControl (управляющая информация о теле сообщения MQ), а также список прочих заголовков MQ.

- MQMD представляет содержимое описания сообщения WebSphere MQ, за исключением информации, определяющей структуру и кодировку тела сообщения.
- MQControl содержит информацию, определяющую структуру и кодировку тела сообщения.
- MQHeaders содержит список объектов MQHeader.

Цепочка заголовков MQ не зацеплена, то есть внутри SMO каждый заголовок MQ несет свою собственную управляющую информацию (CCSID, Encoding, Format). Тем самым заголовки можно добавлять и удалять, не затрагивая другие данные заголовков.

Настройка полей в MQMD

Обновить MQMD можно с помощью Context API или посредством объекта сообщения службы (SMO) в посреднике. Следующие поля автоматически передаются в исходящее сообщение MQ:

- Encoding
- CodedCharacterSet
- Format
- Report
- Expiry
- Feedback
- Priority
- Persistence
- CorrelId
- MsgFlags

Настройте связывание MQ для элемента Import или Export, чтобы передавать следующие свойства в исходящее сообщение MQ:

MsgID
В поле **ИД сообщения-запроса** выберите `Скопировать из сообщения SCA`.

MsgType
Выключите переключатель **Установить тип сообщения MQMT_DATAGRAM или MQMT_REQUEST для операции запрос-ответ**.

ReplyToQ
Выключите переключатель **Переопределить ответ в очередь сообщения-запроса**.

ReplyToQMgr
Выключите переключатель **Переопределить ответ в очередь сообщения-запроса**.

Начиная с версии 7.0 поля контекста можно переопределить в пользовательском свойстве определения целевого объекта JNDI. Присвойте пользовательскому свойству MDCTX значение `SET_IDENTITY_CONTEXT` для целевого расположения отправки, чтобы передавать следующие поля в исходящее сообщение MQ:

- UserIdentifier
- AppIdentityData

Присвойте пользовательскому свойству MDCTX значение `SET_ALL_CONTEXT` для целевого расположения отправки, чтобы передавать следующие свойства в исходящее сообщение MQ:

- UserIdentifier
- AppIdentityData
- PutApplType
- PutApplName
- ApplOriginData

Некоторые поля не передаются в исходящее сообщение MQ. Следующие поля переопределяются в ходе отправки сообщения:

- BackoutCount
- AccountingToken

- PutDate
- PutTime
- Offset
- OriginalLength

Статическое добавление MQCIH в привязку WebSphere MQ:

IBM Business Process Manager поддерживает статическое добавление информации заголовка MQCIH без использования модуля передачи.

Существуют разные способы добавления информации заголовка MQCIH в сообщение (например, с использованием примитива передачи Header Setter). Может быть удобнее добавить эту информацию заголовка без использования дополнительного модуля передачи. Статическая информация заголовка, включающая имя программы CICS, ИД транзакции и другие сведения заголовка формата данных, может быть определена или добавлена как часть привязки WebSphere MQ.

WebSphere MQ, MQ CICS Bridge и CICS должны быть настроены на статическое добавление информации заголовка MQCIH.

Integration Designer можно использовать для настройки импорта WebSphere MQ со статическими значениями, которые требуются для информации заголовка MQCIH.

Когда сообщение приходит и обрабатывается импортом WebSphere MQ, выполняется проверка на присутствие информации заголовка MQCIH в сообщении. Если MQCIH присутствует, то статические значения, определяемые в импорте WebSphere MQ, используются для переопределения соответствующих динамических значений в сообщении. Если MQCIH отсутствует, то он создается в сообщении, и добавляются статические значения, определенные в импорте WebSphere MQ.

Статические значения, определенные в импорте WebSphere MQ, зависят от метода. Можно задать другие статические значения MQCIH для разных методов в пределах одного импорта WebSphere MQ.

Это средство не используется для предоставления значений по умолчанию, если MQCIH не содержит конкретной информации заголовка, поскольку статическое значение, определенное в импорте WebSphere MQ, будет переопределено соответствующим значением, предусмотренным во входящем сообщении.

Внешние клиенты:

IBM Business Process Manager может отправлять сообщения или принимать сообщения от внешних клиентов с помощью привязок WebSphere MQ.

Внешний клиент (например, веб-портал или информационная система организации) может отправить сообщение компоненту SCA в приложении при помощи экспорта или он может быть вызван компонентом SCA в приложении при помощи импорта.

Привязка экспорта WebSphere MQ развертывает управляемые сообщениями EJB (MDB) для прослушивания запросов, приходящих на целевой объект получения, указанный в привязке экспорта. Целевой объект указывается в поле send и используется для отправки ответа на входящий запрос, если вызванное приложение предоставляет ответ. Таким образом, внешний клиент может вызывать приложения через привязку экспорта.

Импорты WebSphere MQ привязываются к внешним клиентам и могут доставлять им сообщения. Эти сообщения могут требовать или могут не требовать ответа от внешнего клиента.

Дополнительную информацию о взаимодействии с внешними клиентами с помощью WebSphere MQ можно найти в справочной системе WebSphere MQ Information Center.

Устранение неполадок привязок WebSphere MQ:

Сбои и условия сбоев привязок WebSphere MQ JMS можно диагностировать и устранять.

Основные условия сбоя

Основные условия сбоя привязок WebSphere MQ определяются семантикой транзакций, конфигурацией WebSphere MQ или ссылкой на фактическое поведение других компонентов. В число основных условий сбоя входят следующие.

- Сбой установки соединения с администратором очередей или очередью WebSphere MQ.

  Сбой установки соединения с WebSphere MQ для приема сообщений приведет к сбою запуска порта получения запросов MDB. Это событие будет записано в протокол WebSphere Application Server. Постоянные сообщения останутся в очереди WebSphere MQ до момента их успешного получения (или пока WebSphere MQ не определит, что их срок истек).

  Сбой установки соединения с WebSphere MQ для отправки исходящих сообщений приведет к откату транзакции, которая управляет отправкой.
- Сбой анализа входящего сообщения или создания исходящего сообщения.

  Сбой связывания данных приводит к откату транзакции, которая управляет работой.
- Сбой отправки исходящего сообщения.

  Сбой отправки исходящего сообщения приводит к откату соответствующей транзакции.
- Получение нескольких или непредвиденных ответных сообщений.

  Импорт ожидает только одно ответное сообщение для каждого сообщения запроса. Если получено несколько ответов или ответ запоздал (время истечения срока ожидания ответа SCA истекло), то выбрасывается исключительная ситуация рабочей среды. Выполняется откат транзакции, и ответное сообщение удаляется из очереди или обрабатывается диспетчером недоставленных событий.

Сценарии неправильного употребления: сравнение с привязками WebSphere MQ JMS

Импорт и экспорт WebSphere MQ изначально предназначены для взаимодействия со стандартными приложениями WebSphere MQ и предоставляют передачам доступ ко всему содержимому тела сообщения WebSphere MQ. Однако привязка WebSphere MQ JMS предназначена для взаимодействия с приложениями JMS, развернутых на WebSphere MQ, которая предоставляет сообщения в соответствии с моделью сообщений JMS.

Следующие сценарии следует компоновать с использованием привязки WebSphere MQ JMS, а не WebSphere MQ.

- Вызов объекта EJB, управляемого сообщениями JMS (MDB) из модуля SCA, если MDB развернут на поставщике WebSphere MQ JMS. Используйте импорт WebSphere MQ JMS.
- Разрешение вызова модуля SCA из сервлета компонента Java EE или EJB с использованием JMS. Используйте экспорт WebSphere MQ JMS.
- Транзитная передача содержимого JMS MapMessage через WebSphere MQ. Используйте экспорт и импорт WebSphere MQ в комбинации с соответствующей привязкой данных.

Существуют ситуации, в которых ожидается взаимодействие привязки WebSphere MQ с привязкой WebSphere MQ JMS. В частности, в случае стыковки приложений WebSphere MQ, использующих Java EE, с приложениями, не использующими Java EE, используйте экспорт WebSphere MQ и импорт WebSphere MQ JMS (или наоборот) в комбинации с соответствующими привязками данных или модулями передачи как по отдельности, так и совместно.

Недоставленные сообщения

Если WebSphere MQ не удается доставить сообщение в требуемое целевое расположение (например, по причине ошибок в конфигурации), то она отправляет сообщения в указанную очередь недоставленных сообщений.

В ходе этого процесса она добавляет заголовок недоставленнооого сообщения в начало тела сообщения. Этот заголовок содержит причины сбоя, исходное целевое расположение и другие сведения.

Сообщения SCA на основе MQ, которые не выводятся в диспетчере событий с ошибками

В случае сбоя сообщений SCA, полученных по причине сбоя взаимодействия с WebSphere MQ, ожидается, что они будут показаны в диспетчере недоставленных сообщений. Если эти сообщения не выводятся в диспетчере недоставленных событий, убедитесь, что значение свойства максимального числа сбоя доставок для целевого расположения WebSphere MQ больше 1. Если присвоить этому свойству значение 2 или большее значение, то то взаимодействие с диспетчером недоставленных событий во время вызовов SCA для привязок WebSphere MQ будет возможным.

Недоставленные сообщения MQ повторно отправляются в ошибочный администратор очередей

Когда для исходящих соединений требуется использовать предварительно определенную фабрику соединений, свойства соединения должны совпадать со свойствами, заданными в спецификации активации, используемой для входящих сообщений.

Предварительно определенная фабрика соединений используется для создания соединения во время повторной отправки недоставленного события и поэтому должна быть настроена для использования того же самого администратора очередей, из которого изначально было получено сообщение.

Обработка исключительных ситуаций:

В конфигурации привязки настраивается способ обработки исключительных ситуаций, сгенерированных обработчиками данных или привязками данных. На действия системы при возникновении такой исключительной ситуации также влияет тип потока передачи.

В обработчике данных или привязке данных, вызванных из привязки, может возникнуть неполадка. Например, обработчик данных может получить сообщение с поврежденными данными или попытаться прочитать сообщение в неверном формате.

Способ обработки таких исключительных ситуаций в привязке зависит от того, каким образом реализован обработчик данных или привязка данных. Рекомендуется, чтобы привязка данных генерировала исключительную ситуацию **DataBindingException**.

В случае обработчика данных ситуация полностью аналогична. Поскольку обработчик данных вызывается из привязки данных, любая исключительная ситуация обработчика данных инкапсулируется в исключительную ситуацию привязки данных. По этой причине об исключительной ситуации **DataHandlerException** сообщается как о **DataBindingException**.

При возникновении любой исключительной ситуации времени выполнения, включая **DataBindingException**, выполняются следующие действия:

- Если поток передачи настроен для выполнения транзакций, то по умолчанию в администраторе событий со сбоем сохраняется сообщение JMS, которое можно обработать или удалить вручную.

  Примечание: В привязке можно настроить такой режим исправления, чтобы сообщение не сохранялось в администраторе событий со сбоем, а вместо этого выполнялся откат.
- Если поток передачи не поддерживает транзакции, то исключительная ситуация регистрируется в протоколе, а сообщение удаляется.

В случае обработчика данных ситуация полностью аналогична. Поскольку обработчик данных вызывается из привязки данных, любая исключительная ситуация обработчика данных инкапсулируется в исключительную ситуацию привязки данных. По этой причине об исключительной ситуации **DataHandlerException** сообщается как о **DataBindingException**.

Ограничения привязок

При использовании привязок действуют определенные ограничения, которые описаны в этом разделе.

Ограничения привязки MQ:

Использование привязки MQ имеет несколько ограничений, которые рассматриваются в этом разделе.

Нет рассылки сообщений публикация-подписка

Метод публикация-подписка рассылки сообщений в настоящее время не поддерживается привязкой MQ, хотя сама WMQ поддерживает сообщения публикация-подписка. Однако привязка MQ JMS поддерживает этот метод рассылки.

Общие очереди получения

Многие привязки экспортов и импортов WebSphere MQ предполагают, что все сообщения, присутствующие в настроенных для них очередях получения, предназначены для этого экспорта или импорта. Привязки импортов и экспортов должны настраиваться с учетом следующего:

- Каждый импорт MQ должна иметь свою очередь получения, поскольку привязка импорта MQ предполагает, что все сообщения в очереди получения являются ответами на отправленные ею запросы. Если очередь получения совместно используется более чем одним импортом, то ответы могут быть получены другим импортом и не смогут быть связаны с исходным сообщением-запросом.
- Каждый экспорт MQ должен иметь свою очередь получения, поскольку в противном случае нельзя будет предсказать, какой экспорт получит конкретное сообщение-запрос.
- Импорты и экспорты MQ могут указывать на одну очередь отправления.

Ограничения привязок JMS, MQ JMS и базовых привязок JMS:

Привязки JMS и MQ JMS имеют некоторые ограничения.

Замечания о создании привязок по умолчанию

Ограничения использования привязок JMS, MQ JMS и базовых привязок JMS рассматриваются в следующих разделах:

- Замечания о создании привязок по умолчанию
- Схема зависимостей ответов
- Поддержка двунаправленного текста

При создании привязки несколько полей заполняются по умолчанию, но можно вводить и другие значения. Например, создается имя фабрики соединений. Если известно, что ваше приложение будет размещено на сервере с удаленным доступом к нему с помощью клиента, то во время создания привязки необходимо ввести имена JNDI, а не использовать значения по умолчанию, поскольку, скорее всего, потребуется управлять этими значениями через административную консоль во время выполнения.

Однако, если после принятия значений по умолчанию оказалось, что у вас нет доступа к приложению из удаленного клиента, то можно воспользоваться административной консолью для явного задания значения фабрики соединений. Найдите поле конечных точек провайдера в параметрах фабрики соединений и добавьте значение, например <имя-хоста-сервера>:7276 (если используется номер порта по умолчанию).

Схема зависимостей ответов

При использовании схемы зависимостей ответов ИД зависимости - ИД зависимости для связывания сообщений в операции запрос-ответ необходимо иметь динамический ИД зависимости в сообщении.

Для создания динамического ИД зависимости в модуле передачи в редакторе потока передачи перед импортом с помощью привязки JMS добавьте примитив передачи Mapping. Откройте редактор преобразований. Известные заголовки архитектуры компонентов служб будут доступны в целевом сообщении. Перенесите поле, содержащее уникальный ИД, в исходное сообщение на ИД зависимости в заголовке JMS целевого сообщения.

Поддержка двунаправленного текста

Во время выполнения для имен Java Naming and Directory Interface (JNDI) поддерживаются только символы ASCII.

Общие очереди получения

Многие привязки экспортов и импортов предполагают, что все сообщения, присутствующие в настроенных для них очередях получения, предназначены для этого экспорта или импорта. Привязки импортов и экспортов должны настраиваться с учетом следующего:

- Каждая привязка импорта должна иметь свою очередь получения, поскольку привязка импорта предполагает, что все сообщения в очереди получения являются ответами на отправленные ею запросы. Если очередь получения совместно используется более чем одним импортом, то ответы могут быть получены другим импортом и не смогут быть связаны с исходным сообщением-запросом.
- Каждый экспорт должен иметь свою очередь получения, поскольку в противном случае нельзя будет предсказать, какой экспорт получит конкретное сообщение-запрос.
- Импорты и экспорты могут указывать на одну очередь отправления.

Бизнес-объекты

В отрасли разработки программного обеспечения для компьютеров разработано несколько моделей и сред программирования, в которых *бизнес-объекты* предоставляют обычное представление бизнес-данных для обработки приложения.

Обычно эти бизнес-объекты имеют следующие характеристики:

- определяются с использованием отраслевых стандартов;
- прозрачно связывают данные с таблицами баз данных или с информационными системами предприятия;
- поддерживают протоколы удаленного вызова;
- предоставляют основу модели программирования приложений.

Process Designer и Integration Designer предоставляют разработчикам одну из таких моделей бизнес-объектов для представления различных видов бизнес-сущностей из различных доменов.

В Process Designer бизнес-объекты концентрируются на представлении типа данных. В наборах инструментов доступны базовые бизнес-объекты (типы переменных). Кроме того, можно создавать пользовательские типы переменных (пользовательские бизнес-объекты). См. раздел Бизнес-объекты и переменные.

В Integration Designer, который доступен только в IBM Business Process Manager Advanced, бизнес-объекты могут представлять более сложные конструкции XSD. В Integration Designer бизнес-объекты имеют тесную привязку со схемами XML. Дополнительная информация об интеграции бизнес-объектов Integration Designer с бизнес-объектами Process Designer приведена в разделах Зеркальное копирование библиотеки и Конструктивные элементы XML не поддерживаются.

Во время разработки в Integration Designer модель бизнес-объектов позволяет разработчикам определять бизнес-объекты в виде определений схемы XML. Во время выполнения бизнес-данные, определенные в определениях схемы XML, представлены в виде бизнес-объектов Java. В этой модели бизнес-объекты основаны на ранних проектах спецификации объектов данных служб (SDO) и предоставляют полный набор интерфейсов приложений программной модели, требуемых для управления бизнес-данными. В следующих подразделах приведена дополнительная информация о применении бизнес-объектов в Integration Designer.

Определение бизнес-объектов

Определения бизнес-объектов создаются с помощью редактора бизнес-объектов в Integration Designer. Редактор бизнес-объектов хранит бизнес-объекты в виде определений схем XML.

Использование схемы XML для определения бизнес-объектов имеет некоторые преимущества.

- Схема XML предоставляет модель определения данных, основанную на стандартах, и базу для взаимодействия между несовместимыми разнородными системами и приложениями. Схемы XML используются совместно с языком описания веб-служб (WSDL) для предоставления стандартных контрактов интерфейсов между компонентами, приложениями и системами.
- Схема XML предоставляет расширенную модель определения данных для представления бизнес-данных. Эта модель, в частности, включает в себя сложные типы, простые типы, пользовательские типы, наследование типов и количество значений.
- Бизнес-объекты могут быть определены в организациях отраслевых стандартов или в других системах и приложениях с помощью бизнес-интерфейсов и данных, определенных на языке описания веб-служб, а также с помощью схемы XML. Integration Designer может непосредственно импортировать эти бизнес-объекты.

Кроме того, Integration Designer обеспечивает поддержку обнаружения бизнес-данных в базах данных и информационных системах предприятия и последующего создания определения бизнес-объекта в виде стандартизированной схемы XML для этих бизнес-данных. Бизнес-объекты, созданные таким образом, часто называются *бизнес-объектами приложений*, поскольку их структура копирует структуру бизнес-данных, определенных в информационной системе предприятия.

Когда процесс манипулирует данными из многих различных информационных систем, очень полезным может оказаться преобразование несовместимого представления бизнес-данных (например, CustomerEIS1 и CustomerEIS2 или OrderEIS1 и OrderEIS2) в единое каноническое представление (например, Customer или Order). Каноническое представление часто называют *обобщенным бизнес-объектом*.

Определения бизнес-объектов, в частности обобщенных, часто используются несколькими приложениями. Для поддержки повторного использования Integration Designer позволяет создавать бизнес-объекты в библиотеках, которые впоследствии можно связывать с несколькими модулями приложения.

Язык описания веб-служб (WSDL) определяет контракты для служб, предоставляемые и принимаемые модулем приложения SCA, а также контракты, используемые для создания компонентов в модуле приложения. На WSDL могут быть описаны как операции, так и бизнес-объекты контракта (определенные схемой XML для представления бизнес-данных).

Работа с бизнес-объектами

Архитектура служебных компонентов (SCA) предоставляет среду определения модуля приложения, служб, которые оно предоставляет, служб, которыми оно пользуется, и состав компонентов, которые обеспечивают бизнес-логику модуля приложения. Бизнес-объекты играют важную роль в приложении, определяя бизнес-данные, которые используются для описания контрактов служб и компонентов и бизнес-данных, которыми манипулируют компоненты.

На следующей диаграмме показан модуль приложения SCA и проиллюстрированы многие точки, в которых разработчик работает с бизнес-объектами.

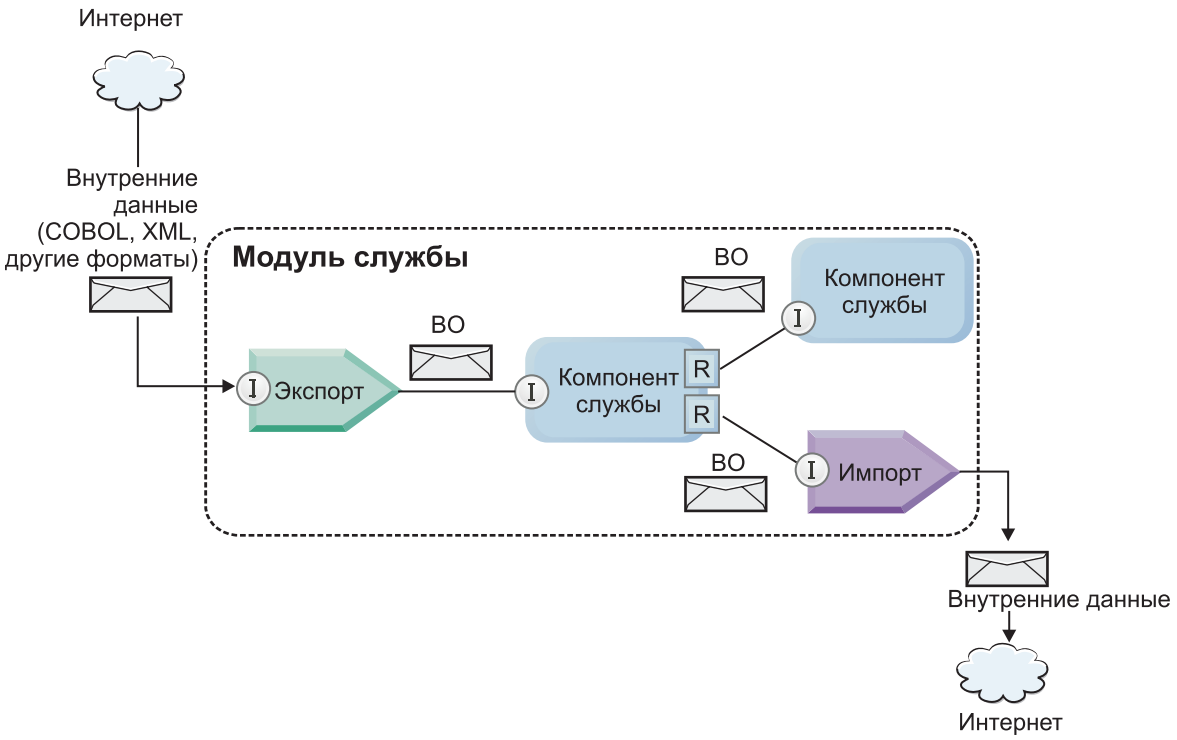


Рисунок 86. Бизнес-объекты представляют собой данные, которыми обмениваются собой службы приложения

Примечание: В этом разделе описано использования бизнес-объектов модулями приложения SCA. Если используются интерфейсы Java, то модули приложения SCA также могут обрабатывать объекты Java.

Программная модель бизнес-объектов

Программная модель бизнес-объектов состоит из набора интерфейсов Java, которые представляют собой:

- определение бизнес-объекта и данные экземпляра;
- набор служб, которые поддерживают операции над бизнес-объектами.

Определения типов бизнес-объектов представлены интерфейсами commonj.sdo.Type и commonj.sdo.Property. Программная модель бизнес-объектов предоставляет набор правил преобразования информации сложных типов схемы XML в интерфейс Type, а также каждого элемента определения сложного типа в интерфейс Property.

Экземпляры бизнес-объектов представлены интерфейсом commonj.sdo.DataObject. Программная модель бизнес-объектов является нетипизированной. Это означает, что один и тот же интерфейс commonj.sdo.DataObject можно использовать для представления различных определений бизнес-объектов, например Customer и Order. Определение свойств, значения которых можно задавать для бизнес-объекта и получать из него, зависят от типа информации, указанной в схеме XML, которая связана с каждым бизнес-объектом.

Поведение программной модели бизнес-объектов основано на спецификации объектов данных служб 2.1. Дополнительные сведения приведены в спецификации SDO 2.1 для Java, учебниках и javadocs в Интернете: http://www.oasis-opencsa.org/.

Службы бизнес-объектов поддерживают различные операции жизненного цикла (например, создание, равенство, анализ и сериализацию) бизнес-объектов.

Специфика программной модели бизнес-объектов описана в разделах Программирование с использованием служб бизнес-объектов и Генерируемая документация по API и SPI для бизнес-объектов.

Привязки, привязки данных и обработчики данных

Как показано в рис. 41 на стр. 153, бизнес-данные, используемые для вызова служб, предоставляемых модулями приложения SCA, преобразуются в бизнес-объекты таким образом, чтобы компоненты SCA могли манипулировать бизнес-данными. Подобным образом бизнес-объекты, которыми манипулируют компоненты SCA, преобразуются в формат данных, требуемый для внешних служб.

В некоторых случаях, например, в случае привязки веб-службы, привязка, используемая для экспорта и импорта служб, автоматически преобразует данные в соответствующий формат. В других случаях, например, в случае привязки JMS, разработчики могут предоставить привязку данных или обработчик данных, который преобразует нестандартные форматы в бизнес-объекты, представленные интерфейсом DataObject.

Дополнительные сведения о разработке привязок данных и обработчиков данных приведены в разделах “Обработчики данных” на стр. 59 и “Привязки данных” на стр. 61.

Компоненты

Компоненты SCA определяют свои контракты служб для передачи и приема с помощью комбинации языка описания веб-служб и схемы XML. Бизнес-данные, передаваемые SCA между компонентами, представлены в виде бизнес-объектов, использующих интерфейс DataObject. SCA проверяет совместимость типов этих бизнес-объектов с контрактом интерфейса, определенного в компоненте, который требуется вызвать.

Обобщения программной модели для манипулирования бизнес-объектами отличаются в различных компонентах. Компонент POJO и примитив Custom компонента потока передачи обеспечивают непосредственное управление бизнес-объектами, включая непосредственное программирование на Java с помощью интерфейсов и служб программирования бизнес-объектов. Большинство компонентов предоставляют высокоуровневые абстракции для манипулирования бизнес-объектами, а также фрагменты кода Java для определения пользовательского поведения в интерфейсах и службах бизнес-объектов.

Бизнес-объекты можно преобразовать с помощью любой комбинации компонентов Interface Flow Mediation и Business Object Map или компонента потока передачи и его примитива карты связей XML. Эти возможности преобразования бизнес-объектов полезны для преобразования бизнес-объектов, специфических для приложения, в обобщенные бизнес-объекты и обратно.

Специальные бизнес-объекты

Объекты служебных сообщений и бизнес-графиков - два специализированных типа бизнес-объектов, которые используются для особых целей приложения.

Объект служебного сообщения

Объект служебного сообщения (SMO) - это специализированный бизнес-объект, который используется компонентами потока передачи для представления набора данных, связанных с вызовом службы.

SMO имеет фиксированную структуру верхнего уровня, состоящую из заголовков, контекста, тела и вложений (если таковые имеются).

- Заголовки содержат информацию, связанную с вызовом службы с использованием отдельного протокола или привязки. Примерами являются заголовки SOAP и заголовки JMS.
- Данные контекста содержат дополнительную логическую информацию, связанную с вызовом во время его обработки компонентом потока передачи. Эта информация обычно не является частью данных приложения, отправляемых или получаемых клиентами.
- Тело SMO содержит бизнес-данные полезной нагрузки, которые представляют базовое сообщение приложения или данных вызова в форме стандартного бизнес-объекта.

Кроме того, SMO может содержать данные вложения для вызовов веб-службы с использованием SOAP со вложениями.

Потоки передачи выполняют такие задачи как маршрутизация запросов и преобразование данных, а SMO предоставляет объединенное представление заголовка и содержимого полезной нагрузки в единой унифицированной структуре.

Бизнес-график

Бизнес-график является специальным бизнес-объектом, который используется для обеспечения поддержки синхронизации данных в сценариях интеграции.

Рассмотрим пример, в котором две информационные системы предприятия имеют представление определенного заказа. При изменении заказа в одной системе можно отправить другой системе сообщение относительно синхронизации данных. Бизнес-графики поддерживают отправку части измененного заказа другой системе с комментарием, содержащем краткие сведения об изменении, определяющие тип изменения.

В данном примере бизнес-график Order сообщит другой системе, что одна из позиций в заказе удалена и что свойство предполагаемой даты отправки заказа обновлено.

Бизнес-графики можно без затруднения добавлять к существующим бизнес-объектам в Integration Designer. Чаще всего они находятся в сценариях, в которых используются адаптеры WebSphere, и для поддержки приложений WebSphere InterChange Server.

Режим анализа бизнес-объектов

В Integration Designer существует свойство модулей и библиотек, которое можно использовать для переключения активного и отложенного режимов анализа XML для бизнес-объектов.

- Если этой опции присвоено значение *активный*, то потоки байтов XML во время создания бизнес-объекта анализируются активно.
- Если опции присвоено значение *отложенный*, то бизнес-объект создается обычным способом, а фактическая обработка потока байтов XML откладывается и частичный анализ проводится только во время обращения к свойствам бизнес-объекта.

В любом режиме анализа XML при создании бизнес-объекта данные, имеющие иной формат, чем XML, всегда анализируются в активном режиме.

Замечания при выборе режима анализа бизнес-объекта

Режим анализа бизнес-объекта определяет, как анализируются данные XML во время выполнения. Режим анализа бизнес-объекта определяется на модуле или библиотеке во время их создания. Можно изменить режим анализа модуля или библиотеки, однако необходимо знать о последствиях.

Режим анализа бизнес-объектов задается на уровне модулей и библиотек. Модули, которые созданы в IBM Integration Designer до версии 7, будут выполняться в режиме активного анализа без необходимости каких-либо изменений. По умолчанию модули и библиотеки, создаваемые в IBM Integration Designer версии 7 и выше пользуются наиболее подходящим режимом анализа в зависимости от числа факторов, в частности от режима анализа существующих проектов в рабочей области или от режима анализа зависимых проектов или других проектов в том же решении и так далее. Режим анализа бизнес-объектов модуля или библиотеки можно изменить в соответствии с требованиями существующей реализации, однако необходимо знать о следующем.

Замечания

- В режиме отложенного анализа бизнес-объектов данные XML обрабатываются быстрее. Однако существуют различия в совместимости между активным режимом и отложенным режимом, о которых необходимо знать перед изменением конфигурации модуля или библиотеки. Эти различия повлияют на поведение модулей во время выполнения. Дополнительные сведения о том, какой режим анализа является

оптимальным для приложения, приведены в разделе "Преимущества использования режима отложенного анализа по сравнению с режимом активного анализа" по связанным ссылкам.

- Модуль можно настроить для выполнения только в одном из режимов анализа. Библиотеки можно настроить для поддержки одного из режимов анализа или обоих режимов анализа. На библиотеку, настроенную для поддержки обоих режимов анализа, может ссылаться как модуль, использующий режим активного анализа, так и модуль, использующий режим отложенного анализа. Во время выполнения режим анализа библиотеки будет определяться модулями, которые на нее ссылаются. Во время выполнения модуль объявляет свой режим анализа, и этот режим анализа используется модулем и всеми библиотеками, которые он использует.
- Модули и библиотеки, настроенные для различных режимов анализа, совместимы в следующих случаях:
  - модули и библиотеки, настроенные для режима отложенного анализа, совместимы с библиотеками, которые используют режим отложенного анализа или как активного, так и отложенного анализа;
  - модули и библиотеки, настроенные для режима активного анализа, совместимы с библиотеками, которые используют режим активного анализа или как активного, так и отложенного анализа;
  - библиотеки, настроенные для режимов отложенного и активного анализа, совместимы только с библиотеками, использующими режим как отложенного, так и активного анализа;
- используйте один и тот же режим анализа для взаимодействия модулей, которые обмениваются сообщениями с использованием привязки SCA. Если модули обмениваются сообщениями с помощью других режимов анализа, то это может привести к недостаточной производительности.

Понятия, связанные с данным:

“Преимущества использования режимов отложенного и активного анализа” на стр. 156
В некоторых приложениях выгодно использовать режим отложенного анализа XML, а в некоторых производительность повышается при использовании режима активного анализа. Рекомендуется протестировать приложение в обоих режимах анализа для того чтобы определить, какой из них больше подходит для конкретных характеристик приложения.

Преимущества использования режимов отложенного и активного анализа

В некоторых приложениях выгодно использовать режим отложенного анализа XML, а в некоторых производительность повышается при использовании режима активного анализа. Рекомендуется протестировать приложение в обоих режимах анализа для того чтобы определить, какой из них больше подходит для конкретных характеристик приложения.

Производительность приложений, анализирующих большие потоки данных XML, повышается в случае использования режима отложенного анализа XML. Производительность повышается с ростом потока байтов XML и уменьшения объема данных в потоке байтов, к которому обращается приложение.

Производительность следующих приложений может повышаться в режиме активного анализа:

- приложения, анализирующие потоки данных не в формате XML;
- приложения с сообщениями, созданные с помощью службы BOFactory;
- приложения, анализирующие очень короткие сообщения в формате XML.

Ссылки, связанные с данной:

“Замечания при выборе режима анализа бизнес-объекта” на стр. 155
Режим анализа бизнес-объекта определяет, как анализируются данные XML во время выполнения. Режим анализа бизнес-объекта определяется на модуле или библиотеке во время их создания. Можно изменить режим анализа модуля или библиотеки, однако необходимо знать о последствиях.

Замечания по миграции приложений и разработке

Если требуется перенастроить приложение, которое изначально разработано для использования режима активного анализа, на использование режима отложенного анализа или если планируется предусмотреть возможность переключения приложения между режимами отложенного и активного анализа, необходимо знать о различиях между режимами и об их переключении.

Обработка ошибок

Если в анализируемом потоке байтов XML присутствуют ошибки формата, то выбрасываются исключительные ситуации анализа.

- В режиме активного анализа XML эти исключительные ситуации происходят в момент выделения бизнес-объекта из входного потока XML.
- Если включен режим отложенного анализа XML, то исключительные ситуации анализа происходят скрыто при доступе к свойствам бизнес-объектов и анализе фрагмента XML с ошибкой формата.

Для работы с XML, в котором присутствует ошибка формата, выберите одну из следующих опций:

- развернуть корпоративную сервисную шину на ребрах для проверки входного XML;
- создать код отложенного распознавания ошибок в точке, в которой происходит обращение к свойствам бизнес-объектов.

Стеки и сообщения об исключительных ситуациях

Поскольку в основе режимов активного и отложенного анализа XML лежат разные реализации, трассировка стека, выбрасываемая интерфейсами программирования бизнес-объектов и службами, содержит одно и то же имя исключительной ситуации, но может содержать иное сообщение об исключительной ситуации или классы исключительных ситуаций, специфических для реализации, в оболочке.

Формат сериализации XML

Режим отложенного анализа XML обеспечивает оптимизацию производительности, при которой во время сериализации производится попытка копирования кода XML из входного потока байтов в выходной без изменений. В результате производительность растет, но формат сериализации выходного потока байтов XML может отличаться в случае обновления всего бизнес-объекта в режиме отложенного анализа и в случае его работы в режиме активного анализа XML.

Несмотря на то, что формат сериализации XML может быть не полностью эквивалентным синтаксически, семантическое значение, представляемое бизнес-объектом, будет эквивалентным независимо от режима анализа, и благодаря семантической эквивалентности приложения могут беспрепятственно обмениваться кодом XML, работая в различных режимах анализа.

Агент проверки экземпляра бизнес-объекта

Агент проверки экземпляра режима отложенного анализа XML бизнес-объекта обеспечивает высококачественную проверку бизнес-объектов, в частности проверку фасетов значений свойств. Благодаря этим усовершенствованиям, агент проверки экземпляра режима отложенного анализа перехватывает дополнительные неполадки, которые не перехватываются в режиме активного анализа, и предоставляет более подробные сообщения об ошибках.

Карты связей XML версии 602

Потоки передачи, изначально разработанные перед WebSphere Integration Developer версии 6.1, могли содержать примитивы Mapping, использующие карту связей таблицы стилей, которая не может непосредственно выполняться в режиме отложенного анализа XML. Во время миграции приложения для использования в режиме отложенного анализа XML файлы карты связей, связанные с примитивами Mapping, могут автоматически обновляться мастером миграции для выполнения в новом режиме. Однако если примитив Mapping относится непосредственно к таблице стилей, которую невозможно отредактировать вручную, то миграция таблицы стилей не происходит, и таблица стилей не может выполняться в режиме отложенного анализа XML.

Частные непубликуемые API

Если в приложении используются непубликуемые частные и специфические для реализации интерфейсы программирования бизнес-объектов, то может произойти сбой компиляции приложения при смене режима анализа. В режиме активного анализа эти частные интерфейсы обычно являются классами реализации бизнес-объекта, определенными в Eclipse Modeling Framework (EMF).

Во всех случаях настоятельно рекомендуется удалять частные API из приложения.

API EMF для объектов служебного сообщения

Компонент отображения в IBM Integration Designer предоставляет возможность манипулирования содержимым сообщения с помощью классов и интерфейсов Java из пакета com.ibm.websphere.sibx.smobo. В режиме отложенного анализа XML интерфейсы Java из пакета com.ibm.websphere.sibx.smobo также могут использоваться, но методы, которые относятся непосредственно к классам и интерфейсам Eclipse Modeling Framework (EMF) или наследуются от интерфейсов EMF, могут дать сбой.

ServiceMessageObject и его содержимое невозможно преобразовать в объекты EMF в режиме отложенного анализа XML.

Служба BOMode

Служба BOMode используется для того чтобы определить, является ли текущий режим анализа активным или отложенным.

Миграция

Все приложения до версии 7.0.0.0 работают в режиме активного анализа XML. При их миграции во время выполнения с помощью инструментов BPM для миграции во времени выполнения они продолжают работать в режиме активного анализа XML.

Для того чтобы приложение до версии 7.0.0.0 можно было использовать в режиме отложенного анализа XML, сначала требуется использовать Integration Designer для миграции артефактов приложения. После миграции можно настроить приложение для использования режима отложенного анализа XML.

Сведения о миграции артефактов в Integration Designer приведены в разделе Миграция исходных артефактов, а сведения о назначении режима анализа - в разделе Настройка режима анализа бизнес-объектов в модулях и библиотеках.

Взаимосвязи

Взаимосвязь определяет связь между двумя или более элементами данных, как правило - бизнес-объектами. В IBM Business Process Manager Advanced взаимосвязь может применяться для преобразования данных, которые одинаковы во всех бизнес-объектах или имеют различное представление, либо для определения взаимосвязей между различными объектами, расположенными в разных приложениях. Взаимосвязи могут совместно использоваться внутри приложения, в нескольких решениях и даже в нескольких продуктах.

Служба взаимосвязей в IBM Business Process Manager Advanced предоставляет инфраструктуру и операции для управления взаимосвязями. Поскольку она позволяет работать с любыми бизнес-объектами вне зависимости от их расположения, с ее помощью можно получить единую картину, описывающую все приложения предприятия. Кроме того, ее можно использовать как конструктивный блок в решениях BPM. Расширяемые и управляемые взаимосвязи могут использоваться в сложных решениях интеграции.

Что такое взаимосвязи?

Взаимосвязь определяет связь между бизнес-объектами. Каждый из таких бизнес-объектов называется *участником* взаимосвязи. Участники взаимосвязи отличаются друг от друга по выполняемой ими функции, или *роли*. Взаимосвязь содержит список возможных ролей.

Определение взаимосвязи содержит описание каждой роли и их связей между собой. Кроме того, оно описывает общее представление взаимосвязи. Например, одна роль может иметь только одного участника, а другая - произвольное число участников. Например, можно определить взаимосвязь *автомобиль-владелец*, в которой одному владельцу может принадлежать несколько автомобилей. Один из экземпляров этой взаимосвязи может иметь следующих участников в каждой из ролей:

- Автомобиль (Ferrari)
- Владелец (John)

Определение взаимосвязи задает шаблон *экземпляра* взаимосвязи. Экземпляр - это динамическое представление взаимосвязи. В примере взаимосвязи *автомобиль-владелец* экземпляр может описывать любое из следующих отношений:

- John владеет Ferrari
- Sara владеет Mazda
- Bob владеет Ferrari

Использование взаимосвязей освобождает от необходимости создания пользовательского кода для определения и отслеживания взаимосвязей в бизнес-логике. В ряде случаев служба взаимосвязей может делать все автоматически. Ознакомьтесь с примером, описанным в разделе Взаимосвязи идентификаторов.

Сценарии

Здесь описана типичная ситуация, в которой решению интеграции могут потребоваться взаимосвязи. Крупная компания покупает несколько компаний, или бизнес-подразделений. Каждое бизнес-подразделение использует свое программное обеспечение для отслеживания сотрудников и ноутбуков. Компании требуется средство для отслеживания всех сотрудников и их ноутбуков. Это решение должно предоставлять возможности для выполнения следующих задач:

- Просмотр всех сотрудников из разных бизнес-подразделений так, как если бы информация о них хранилась в одной базе данных
- Единое представление всех портативных ноутбуков
- Возможность для сотрудника войти в систему и купить ноутбук
- Объединение различных прикладных систем предприятия из разных бизнес-подразделений

Для решения этих задач компании, в частности, необходимо средство для того, чтобы пользователь John Smith из одного приложения и пользователь John A. Smith из другого приложения был показан как один сотрудник. То есть необходим способ получения консолидированного представления на основе данных из нескольких приложений.

Более сложные сценарии использования взаимосвязей включают в себя создание процессов BPEL, которые строят взаимосвязи между объектами из разных приложений. В еще более сложном сценарии бизнес-объекты расположены в решении интеграции, а не в приложениях. Служба взаимосвязей предоставляет платформу для постоянного управления взаимосвязями. При отсутствии такой службы вам бы потребовалось создать собственную службу хранения объектов. Ниже приведены два примера сложных сценариев использования взаимосвязей:

- В приложении SAP есть бизнес-объект **car** (автомобиль) с номером VIN, и необходимо узнать, не принадлежит ли данный автомобиль кому-то еще. Однако взаимосвязь владения установлена с пользователем, определенным в приложении PeopleSoft. В данном случае есть два решения, между которыми необходимо создать мост.

- Крупная компания, занимающаяся розничной торговлей, хочет получить возможность отслеживать возвращенные товары, деньги за которые были возвращены наличными или на кредитную карту. В этом должны участвовать два разных приложения: система управления заказами (OMS) с информацией о покупках и система управления возвратами (RMS) с информацией о возврате товара. Бизнес-объекты расположены в нескольких приложениях, и необходимо каким-то образом показать существующие между ними взаимосвязи.

Стандартные шаблоны

Чаще всего используются такие шаблоны взаимосвязей, как шаблоны *эквивалентности*. Они основаны на перекрестных ссылках или соотношении. Существует два типа взаимосвязей с таким шаблоном: *неидентифицирующие* и *идентифицирующие*.

- **Неидентифицирующие взаимосвязи** определяют связь между бизнес-объектами и другими данными по схеме один-ко-многим или многие-ко-многим. В каждом экземпляре взаимосвязи может быть один или несколько экземпляров каждого участника. Один из примеров неидентифицирующей взаимосвязи - это статическая взаимосвязь поиска. Например, она может связывать **CA** в приложении SAP со значением **California** в приложении Siebel.
- **Идентифицирующая взаимосвязь** устанавливает связь между бизнес-объектами и другими данными по схеме один-к-одному. В каждом экземпляре взаимосвязи может быть только один экземпляр каждого участника. Идентифицирующие взаимосвязи отражают перекрестные ссылки между семантически эквивалентными бизнес-объектами, которые по-разному идентифицируются в разных приложениях. Каждый участник взаимосвязи представляет бизнес-объект, у которого есть одно или несколько значений, которые однозначно идентифицируют этот объект. Идентифицирующие взаимосвязи обычно служат для преобразования ключевых атрибутов бизнес-объектов, таких как ИД или код продукта.

  Например, если в приложениях SAP, PeopleSoft и Siebel есть бизнес-объект **car**, то для их синхронизации в обычной ситуации вам потребовалось бы вручную создать логику синхронизации взаимосвязей, состоящую из шести связей:

  SAP -> универсальный объект

  универсальный объект -> SAP

  PeopleSoft-> универсальный объект

  универсальный объект-> PeopleSoft

  Siebel-> универсальный объект

  универсальный объект-> Siebel

  Однако служба взаимосвязей предоставляет готовые реализации шаблонов, содержащие все указанные экземпляры взаимосвязей.

Инструменты для работы с взаимосвязями

Редактор взаимосвязей из Integration Designer позволяет моделировать роли и взаимосвязи бизнес-интеграции. Подробная информация и инструкции по созданию взаимосвязей и использованию редактора взаимосвязей приведены в разделе Создание взаимосвязей.

Служба взаимосвязей - это инфраструктурная служба из состава IBM Business Process Manager, сохраняющая взаимосвязи и роли в системе и предоставляющая операции для управления ролями и взаимосвязями.

Администратор взаимосвязей - это административный интерфейс для управления взаимосвязями. С ним можно работать на страницах Администратор взаимосвязей в административной консоли.

Взаимосвязи можно вызывать внутри программ с помощью API службы взаимосвязей.

Служба взаимосвязей

Служба взаимосвязей хранит данные взаимосвязей в таблицах взаимосвязей, в которых содержатся значения, заданные в различных приложениях и решениях. Служба взаимосвязей предоставляет операции для управления ролями и взаимосвязями.

Принципы работы взаимосвязей

Взаимосвязи и роли можно определить в графическом интерфейсе редактора взаимосвязей, который доступен в Integration Designer. Служба взаимосвязей хранит данные о соответствии в таблицах базы данных взаимосвязей, расположенной в источнике данных, заданном при настройке службы взаимосвязей. Отдельная таблица (иногда именуемая таблицей участников) содержит информацию о каждом участнике взаимосвязи. Служба взаимосвязей использует таблицы взаимосвязей для отслеживания значений, заданных в разных приложениях, и передачи обновленной информации во все решения.

Взаимосвязи, представляющие собой бизнес-артефакты, можно развернуть в проекте или в общей библиотеке. При первом развертывании служба взаимосвязей добавляет данные.

Когда во время выполнения связям или другим компонентам IBM Business Process Manager требуется экземпляр взаимосвязи, экземпляры взаимосвязи обновляются или извлекаются, в зависимости от сценария.

С данными экземпляров взаимосвязи и роли можно работать тремя способами:

- Через вызовы API службы взаимосвязи во фрагменте кода на Java в компоненте IBM Business Process Manager
- С помощью преобразования взаимосвязей в службе преобразования бизнес-объектов IBM Business Process Manager
- Используя администратор взаимосвязей

Подробную информацию и пошаговые инструкции по созданию взаимосвязей, определению типов взаимосвязей и использованию редактора взаимосвязей можно найти в разделе Создание взаимосвязей.

Администратор взаимосвязей

Администратор взаимосвязей - это административный интерфейс для управления взаимосвязями. С ним можно работать на страницах Администратор взаимосвязей в административной консоли.

Администратор взаимосвязей предоставляет графический пользовательский интерфейс для создания и изменения данных о взаимосвязях и ролях во время выполнения. Элементами взаимосвязей можно управлять на всех уровнях: на уровне экземпляра взаимосвязи, экземпляра роли, данных атрибутов и данных свойств. С помощью администратора взаимосвязей можно выполнить следующие действия:

- Просмотреть список взаимосвязей в системе и подробную информацию о каждой взаимосвязи
- Управлять экземплярами взаимосвязей:
  - Запросить данные взаимосвязи для просмотра подмножеств данных экземпляра
  - Запросить данные взаимосвязи для просмотра подмножеств данных экземпляра с помощью представлений базы данных
  - Просмотреть список экземпляров взаимосвязей, соответствующих запросу, а также подробную информацию о каждом экземпляре
  - Изменить значения свойств экземпляра взаимосвязи
  - Создать или удалить экземпляр взаимосвязи
- Управлять ролями и экземплярами ролей:
  - Просмотреть сведения о роли или ее экземпляре
  - Изменить свойства экземпляра роли
  - Создать или удалить экземпляр роли для взаимосвязи

  - Выполнить откат данных экземпляра взаимосвязи до того момента времени, когда данные еще были правильными
- Импортировать данные из статической взаимосвязи в свою систему или экспортировать данные статической взаимосвязи в файл RI ли CSV
- Удалить схему и данные взаимосвязи из хранилища при удалении приложения, использовавшего эту взаимосвязь

Взаимосвязи в среде сетевого развертывания

Для использования взаимосвязей в среде сетевого развертывания (ND) не требуется выполнять никакую дополнительную настройку.

В средах сетевого развертывания (ND) взаимосвязи устанавливаются в кластере приложений. Эти взаимосвязи доступны во всем кластере, то есть любой сервер кластера может использовать данные экземпляра, хранящиеся в базе данных взаимосвязей. Возможность запуска службы взаимосвязей в среде ND повышает ее масштабируемость и готовность.

Администратор взаимосвязей позволяет управлять взаимосвязями из разных кластеров, используя единый административный интерфейс. Для подключения администратора взаимосвязей к серверу кластера необходимо выбрать его MBean взаимосвязи.

API службы взаимосвязей

Взаимосвязи можно вызывать внутри программ с помощью API службы взаимосвязей, используя их внутри связей бизнес-объектов или за их пределами.

Доступны три типа API:

- API для работы с экземплярами взаимосвязей (в том числе создания, обновления и удаления данных экземпляра напрямую)
- API для поддержки шаблонов взаимосвязей (в том числе correlate(), correlateforeignKeyLookup)
- Шаблоны поиска взаимосвязей (API поиска)

Шина служб предприятия в IBM Business Process Manager

IBM Business Process Manager поддерживает интеграцию служб приложений, в том числе весь набор функций продукта WebSphere Enterprise Service Bus.

Подключение служб через Enterprise Service Bus

С помощью Enterprise Service Bus (ESB) можно значительно повысить гибкость SOA. Участники взаимодействия служб соединяются с ESB, а не непосредственно друг с другом.

Когда клиент соединяется с ESB, ESB принимает на себя ответственность за доставку этих запросов, с помощью сообщений, провайдеру служб, предоставляющему требуемые функцию и Quality of Service. ESB обеспечивает взаимодействия клиента и провайдера и адресует несоответствующие протоколы, шаблоны взаимодействия или возможности служб. ESB также обеспечивает или расширяет мониторинг и управление. ESB предоставляет средства виртуализации и управления, которые реализуют и расширяют базовые возможности SOA.

ESB абстрагирует следующее:

Расположение и идентификация

Участникам не нужно знать расположение или идентификатор других участников. Например, клиентам не требуется знать, каким из нескольких провайдеров обслуживается запрос; провайдеры служб могут добавляться и удаляться без прерывания.

Протокол взаимодействия

Участникам не требуется совместно использовать один и тот же протокол связи или стиль взаимодействия. Например, запрос, выраженный как SOAP через HTTP, может обслуживаться провайдером, который использует только SOAP через Java Message Service (JMS).

Интерфейс

Клиентам и провайдерам не нужно согласовывать общий интерфейс. ESB согласовывает различия преобразованием запроса и ответных сообщений в форму, ожидаемую провайдером.

Качества службы (взаимодействия)

Участники или системные администраторы объявляют свои требования к Quality of Service, включая идентификацию запросов, шифрование и дешифрование содержания сообщений, автоматический контроль взаимодействия служб и то, как их запросы должны маршрутизироваться (например, с оптимизацией скорости или стоимости).

Вставка ESB между участниками позволяет модулировать их взаимодействие через логическую конструкцию, называемую *передачей*. Передачи оперируют сообщениями на их пути между клиентами и провайдерами. Например, передачи могут использоваться для поиска служб с конкретными характеристиками, которые запрашивает клиент, или для разрешения различий в интерфейсах между клиентами и провайдерами. Для сложных взаимодействий передачи могут соединяться последовательно.

С помощью передач Enterprise Service Bus выполняет следующие действия между клиентом и службой:

- *Маршрутизация* сообщений между службами. Enterprise Service Bus предоставляет общую инфраструктуру связей, которую можно использовать для соединений служб, и тем самым бизнес-функций, которые они представляют, без необходимости написания и обслуживания сложной логики связей программистами.
- *Преобразование* протоколов передачи между клиентом и службой. Enterprise Service Bus обеспечивает основанный на стандартах совместимый способ интеграции бизнес-функций, которые используют разные стандарты ИТ. Он предусматривает интеграцию бизнес-функций, которые обычно несовместимы, например, соединение приложений в подразделенческих отделениях или предоставление приложений другим компаниям для участия в служебных взаимодействиях.
- *Преобразование* форматов сообщений между клиентом и службой. Enterprise Service Bus позволяет бизнес-функциям изменять информацию в разных форматах с помощью шины, обеспечивающей, чтобы информация, доставляемая бизнес-функции, была в формате, требуемом этим приложением.
- *Обработка* бизнес-событий из разных источников. Enterprise Service Bus поддерживает основанные на событиях взаимодействия, кроме обмена сообщениями для обработки запросов служб.

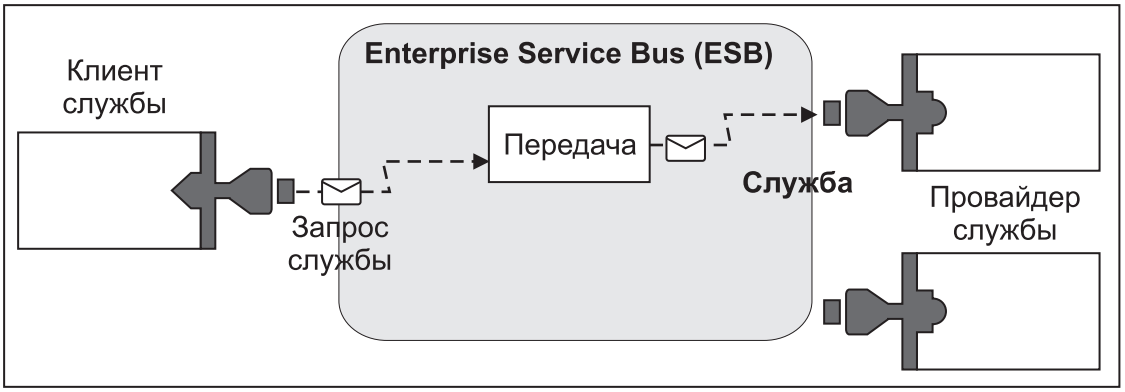


Рисунок 87. Enterprise Service Bus. Enterprise Service Bus направляет сообщения между приложениями, которыми являются клиенты или провайдеры служб. Эта шина преобразует протоколы передачи и форматы сообщений между клиентами и провайдерами. На следующем рисунке каждое приложение использует свой протокол (показанный разными геометрическими формами их соединителей) и свой формат сообщений.

Использование Enterprise Service Bus позволяет сосредоточиться на основной деятельности и не отвлекаться на свои компьютерные системы. При необходимости можно изменять и добавлять службы. Например, в

ответ на изменения бизнес-требования можно предоставить дополнительный ресурс службы или добавить новые возможности. Необходимые изменения можно вносить перенастройкой шины без влияния или с небольшим влиянием на существующие службы и приложения, использующие шину.

Инфраструктура обмена сообщениями шины служб предприятия

IBM Business Process Manager предоставляет функции шины служб предприятия. IBM Business Process Manager поддерживает интеграцию технологий, основанных на службах, сообщениях, а также технологий, управляемых событиями, для предоставления инфраструктуры обмена сообщениями, основанной на стандартах, в интегрированной шине служб предприятия.

Функции служб предприятия, которые можно использовать в своем приложении предприятия, предоставляют не только транспортный уровень, но и поддержку средств передачи данных для обеспечения взаимодействия служб. Шина служб предприятия основана на открытых стандартах и архитектуре на основе служб (SOA). Она использует надежную инфраструктуру Java EE и связанные с ней службы платформы, предоставляемые IBM WebSphere Application Server Network Deployment.

IBM Business Process Manager включает в себя аналогичную технологию, предоставляемую IBM WebSphere Enterprise Service Bus. Эта функция входит в состав IBM Business Process Manager, поэтому для ее применения не требуется отдельная лицензия на WebSphere Enterprise Service Bus.

Однако в среде предприятия можно установить дополнительные автономные лицензии на продукт WebSphere Enterprise Service Bus для расширения набора средств связи в решениях интеграции процессов, основанных на IBM Business Process Manager. Например, WebSphere Enterprise Service Bus можно установить рядом с приложением SAP для обслуживания IBM WebSphere Adapter for SAP и преобразования сообщений SAP перед их отправкой по сети в бизнес-процессы, созданные с помощью IBM Business Process Manager.

Целевые хосты сообщений и очередей

Целевые хосты сообщений или очередей обеспечивают функцию обмена сообщениями на сервере. Для того чтобы сервер стал целевым хостом сообщений, настройте его в качестве адресата сообщений.

На сервере работает модуль обмена сообщениями. Модуль обмена сообщениями предоставляет функции обмена сообщениями и точку подключения для приложений, которые подключаются к шине. Асинхронная связь SCA, импорт и экспорт JMS, а также асинхронная внутренняя обработка используют очереди модуля обмена сообщениями.

Среда развертывания подключает адресанта к адресату сообщений через шину при развертывании модулей приложений. Если известен адресант и адресат сообщений, можно определить необходимый тип среды развертывания.

Приложения могут хранить постоянные данные в хранилище данных, которое представляет собой набор таблиц в базе данных/схеме или в хранилище файлов. Модуль обмена сообщениями использует экземпляр источника данных JDBC для взаимодействия с этой базой данных.

Настройте целевой хост сообщений при определении среды развертывания, выбрав пункт **Сервер** в административной консоли или назначив сервер целевым хостом при установке ПО.

Хранилища данных:

Каждый модуль обмена сообщениями может использовать хранилище данных, представляющее собой набор таблиц базы данных или схему, хранящую постоянные данные.

Все таблицы хранилища данных расположены в одной схеме таблиц базы данных. Каждое хранилище данных можно создать в отдельной базе данных. Вместо этого можно создать несколько хранилищ данных в одной базе данных так, чтобы каждое хранилище использовало свою схему.

Модуль обмена сообщениями использует экземпляр источника данных JDBC для взаимодействия с базой данных, содержащей хранилище данных этого модуля.

Комплексы связи JDBC

Комплексы связи JDBC можно использовать для взаимодействия приложений с реляционными базами данных.

Приложения используют комплексы связи JDBC для взаимодействия с реляционными базами данных. Комплекс связи JDBC сообщает определенный класс реализации драйвера JDBC для доступа к определенному типу баз данных. Для того чтобы создать пул подключений к этой базе данных, свяжите источник данных с комплексом связи JDBC. Комплекс связи JDBC и объекты источника данных функционируют вместе подобно фабрике соединений Java EE Connector Architecture (JCA), обеспечивающей соединение с нереляционными базами данных.

Примеры установки типовой автономной среды и типовой среды развертывания приведены в предыдущем разделе.

Дополнительная информация о комплексах связи JDBC приведена в разделе “Комплексы связи JDBC” справочной системы Information center WebSphere Application Server.

Шина интеграции служб для IBM Business Process Manager

Шина интеграции служб - управляемый механизм связи, поддерживающий интеграцию служб путем синхронного и асинхронного обмена сообщениями. Шина состоит из взаимосвязанных модулей обмена сообщениями, которые управляют ресурсами шины. Это одна из технологий WebSphere Application Server, на которой основывается IBM Business Process Manager.

Одиночная шина интеграции служб, как и одиночный модуль обмена сообщениями, использует ту же схему базы данных, которая применяется в базе данных продукта по умолчанию. У каждой среды развертывания имеется собственная шина. Одиночная шина имеет имя **BPM.*имя-среды-развертывания*.Bus**.

Целевой объект шины - это логический адрес, к которому приложение может обратиться как источник и/или приемник данных. Целевой объект очереди - целевое объект шины, используемый для двухточечного обмена сообщениями.

Каждая шина может включать один или несколько элементов, каждый из которых является сервером или кластером.

Топология шины - физическое размещение серверов приложений, служб обмена сообщениями и администраторов очередей WebSphere MQ, а также шаблонов соединений шины и ссылок между ними, образующих шину служб предприятия.

Служебные приложения и служебные модули

Служебный модуль - это модуль SCA, предоставляющий службы в среде выполнения. При развертывании служебного модуля в IBM Business Process Manager выполняется компоновка связанного служебного приложения, которое упаковывается в файл EAR.

Служебные модули являются базовыми блоками развертывания и могут содержать компоненты, библиотеки и промежуточные модули, используемые связанным служебным приложением. Служебные модули имеют экспорты и могут иметь импорты для определения взаимосвязей между модулями и клиентами и провайдерами служб. WebSphere Process Server поддерживает модули для бизнес-служб и модули передачи. Как модули, так и модули передачи являются типами модулей SCA. Модуль передачи обеспечивает связь между приложениями путем преобразования вызова службы в формат, понятный целевому объекту, передачи запроса целевому объекту и возврата результата отправителю. Модуль для бизнес-службы реализует логику бизнес-процесса. Однако модуль может также включать ту же логику передачи, которая может быть упакована в модуле передачи.

Развертывание служебного приложения

Процесс развертывания файла EAR, содержащего служебное приложение, ничем не отличается от процесса развертывания любого файла EAR. Во время развертывания можно изменить значения параметров передачи. После развертывания файла EAR, содержащего модуль SCA, можно просмотреть сведения о служебном приложении и связанном с ним модулем. Можно посмотреть, как служебный модуль соединяется с клиентами служб (через экспорты) и провайдерами служб (через импорты).

Просмотр сведений о модуле SCA

Сведения о служебном модуле, доступные для просмотра, зависят от модуля SCA. В них предусмотрены следующие атрибуты.

- Имя модуля SCA
- Описание модуля SCA
- Имя связанного приложения
- Информация о версии модуля SCA, если модуль имеет разные версии
- Импорты модуля SCA:
  - Интерфейсы импорта являются абстрактными определениями, которые описывают, как модуль SCA обращается к службе.
  - Привязки импортов являются конкретными определениями, которые описывают физический механизм, с помощью которого модуль SCA обращается к службе. Например, с помощью SOAP/HTTP.
- Экспорты модуля SCA:
  - Интерфейсы экспортов являются абстрактными определениями, которые описывают, как клиенты служб обращаются к модулю SCA.
  - Привязки экспортов являются конкретными определениями, которые описывают физический механизм, с помощью которого клиент службы обращается к модулю SCA и, косвенно, к службе.
- Свойства модуля SCA

Импорт и привязка импорта

Импорт определяет взаимодействие между модулями архитектуры компонентов служб (SCA) и провайдерами служб. Модули SCA используют импорты для обеспечения доступа компонентов к внешним службам (за пределами модуля SCA) с использованием локального представления. Привязки импортов определяют конкретный способ обращения к внешней службе.

Если модулям SCA не требуется обращение к внешним службам, то им не нужны и импорты. Модули передачи обычно имеют один или несколько импортов, которые используются для передачи сообщений или запросов назначенным целевым объектам.

Интерфейсы и привязки

Импорту модуля SCA требуется по крайней мере один интерфейс, и импорт модуля SCA имеет одну привязку.

- Интерфейсы импортов являются абстрактными определениями, обозначающими некоторый набор операций, использующих язык описания веб-служб (WSDL), язык XML для описания веб-служб. Модуль SCA может иметь много интерфейсов импортов.
- Привязки импортов являются конкретными определениями, которые обозначают физический механизм, который модули SCA используют для доступа к внешней службе.

Поддерживаемые привязи импортов

IBM Business Process Manager поддерживает следующие привязки импортов:

- Привязки SCA связывают модули SCA с другими модулями SCA. Привязки SCA также указываются как привязки по умолчанию.

- Привязки веб-служб позволяют компонентам вызывать веб-службы. Поддерживаются протоколы SOAP1.1/HTTP, SOAP1.2/HTTP и SOAP1.1/JMS.

  Можно использовать привязку SOAP1.1/HTTP или SOAP1.2/HTTP, основанную на Java API for XML Web Services (JAX-WS), которая предусматривает взаимодействие со службами с использованием документа или литеральных привязок RPC и которая использует обработчики JAX-WS для настройки вызовов. Отдельная привязка SOAP1.1/HTTP предусматривается для обеспечения взаимодействия со службами, которые используют привязку в коде RPC или где существует требование использования обработчиков JAX-RPC для настройки вызовов.
- Привязки HTTP допускают обращение к приложениям с помощью протокола HTTP.
- Привязки импортов Enterprise JavaBeans (EJB) позволяют компонентам SCA вызывать службы, предоставляемые бизнес-логикой Java EE, выполняемой на сервере Java EE.
- Привязки информационной системы предприятия (EIS) обеспечивают связь между компонентами SCA и внешней EIS. Эта связь обеспечивается использованием адаптеров ресурсов.
- Привязки Java Message Service (JMS) 1.1 допускают взаимодействие с провайдером обмена сообщениями WebSphere Application Server по умолчанию. JMS может использовать разные типы передачи, включая TCP/IP и HTTP или HTTPS. Класс сообщений JMS и его пять подтипов (Текст, Байты, Объект, Поток и Преобразование) автоматически поддерживаются.
- Generic JMS bindings permit interoperability with third-party JMS providers that integrate with the WebSphere Application Server using the JMS Application Server Facility (ASF).
- Привязки WebSphere MQ JMS допускают взаимодействие с провайдерами JMS на основе WebSphere MQ. Класс сообщений JMS и его пять подтипов (Текст, Байты, Объект, Поток и Преобразование) автоматически поддерживаются. Для использования WebSphere MQ в качестве провайдера JMS используйте привязки WebSphere MQ JMS.
- Привязки WebSphere MQ допускают взаимодействие с WebSphere MQ. Привязки WebSphere MQ можно использовать только с удаленными администраторами очередей через соединение с клиентом WebSphere MQ; они не могут использоваться с локальными администраторами очередей. Для соединения с внутренними приложениями WebSphere MQ используйте привязки WebSphere MQ.

Динамический вызов служб

Службы могут через любую поддерживаемую привязку импорта. Служба обычно находится в конечной точке, указанной в импорте. Эта конечная точка вызывается статической конечной точкой. Другую службу можно вызвать переопределением статической конечной точки. Динамическое переопределение статических конечных точек позволяет вызвать службу в другой конечной точке через любую поддерживаемую привязку импорта. Динамический вызов служб позволяет вызвать службу, где поддерживаемая привязка импорта не имеет статической конечной точки.

Импорт со связанной привязкой используется для задания протокола и его конфигурации для динамического вызова. Импорт, используемый для динамического вызова, может быть соединен с вызывающим компонентом или динамически выбран во время выполнения.

Для веб-службы и вызовов SCA также возможно сделать динамический вызов без импорта с помощью протокола и конфигурации, извлеченных из URL конечной точки. Тип целевого объекта вызова определяется из URL конечной точки. Если используется импорт, то URL должен быть совместим с протоколом привязки импорта.

- URL SCA указывает вызов другого модуля SCA.
- URL HTTP или JMS по умолчанию указывает вызов веб-службы; для этих URL можно предоставить дополнительное значение типа привязки, которое указывает, что URL представляет вызов через привязку HTTP или JMS.
- Для URL HTTP веб-службы по умолчанию используется SOAP 1.1 и может быть задано значение типа привязки, которое указывает использование SOAP 1.2.

Экспорт и привязка экспорта

Экспорт определяет взаимодействие между модулями архитектуры компонентов служб (SCA) и клиентами служб. Модули SCA используют экспорты для предоставления служб другим. Привязки экспортов определяют конкретный способ обращения клиентов служб к модулю SCA.

Интерфейсы и привязки

Экспорту модуля SCA требуется по крайней мере один интерфейс.

- Интерфейсы экспортов являются абстрактными определениями, обозначающими некоторый набор операций, использующих язык описания веб-служб (WSDL), язык XML для описания веб-служб. Модуль SCA может иметь много интерфейсов экспортов.
- Привязки экспортов являются конкретными определениями, которые обозначают физический механизм, который клиенты служб используют для доступа к службе. Обычно экспорт модуля SCA имеет одну заданную привязку. Экспорт без заданной привязки интерпретируется средой выполнения как экспорт с привязкой SCA.

Поддерживаемые привязи экспортов

IBM Business Process Manager поддерживает следующие привязки экспортов:

- Привязки SCA связывают модули SCA с другими модулями SCA. Привязки SCA также указываются как привязки по умолчанию.
- Привязки веб-служб допускают вызов экспортов как веб-служб. Поддерживаются протоколы SOAP1.1/HTTP, SOAP1.2/HTTP и SOAP1.1/JMS.

  Можно использовать привязку SOAP1.1/HTTP или SOAP1.2/HTTP, основанную на Java API for XML Web Services (JAX-WS), которая предусматривает взаимодействие со службами с использованием документа или литеральных привязок RPC и которая использует обработчики JAX-WS для настройки вызовов. Отдельная привязка SOAP1.1/HTTP предусматривается для обеспечения взаимодействия со службами, которые используют привязку в коде RPC или где существует требование использования обработчиков JAX-RPC для настройки вызовов.
- Привязки HTTP допускают обращение к экспортам с помощью протокола HTTP.
- Привязки экспортов Enterprise JavaBeans (EJB) допускают представление компонентов SCA как EJB, чтобы бизнес-логика Java EE могла вызывать компоненты SCA, которые в противном случае недоступны ей.
- Привязки информационной системы предприятия (EIS) обеспечивают связь между компонентами SCA и внешней EIS. Эта связь обеспечивается использованием адаптеров ресурсов.
- Привязки Java Message Service (JMS) 1.1 допускают взаимодействие с провайдером обмена сообщениями WebSphere Application Server по умолчанию. JMS может использовать разные типы передачи, включая TCP/IP и HTTP или HTTPS. Класс сообщений JMS и его пять подтипов (Текст, Байты, Объект, Поток и Преобразование) автоматически поддерживаются.
- Базовые привязки JMS допускают взаимодействие с внешними провайдерами, которые интегрируются с WebSphere Application Server с помощью JMS Application Server Facility (ASF).
- Привязки WebSphere MQ JMS допускают взаимодействие с провайдерами JMS на основе WebSphere MQ. Класс сообщений JMS и его пять подтипов (Текст, Байты, Объект, Поток и Преобразование) автоматически поддерживаются. Для использования WebSphere MQ в качестве провайдера JMS используйте привязки WebSphere MQ JMS.
- Привязки WebSphere MQ допускают взаимодействие с WebSphere MQ. Для соединения с администратором очередей MQ в удаленной системе используется удаленное соединение (или клиент). Локальное соединение (или привязки) являются прямым соединением с WebSphere MQ. Оно может использоваться для соединения с администратором очередей MQ только в той же системе. WebSphere MQ предусматривает оба типа соединений, но привязки MQ поддерживают только "удаленное" (или "клиентское") соединение.

Модули передачи

Модули передачи являются модулями архитектуры компонентов служб (SCA), которые могут изменять формат, содержимое или целевой объект запросов служб.

Модули передачи работают с сообщениями, которые находятся на пути между клиентами служб и провайдерами служб. Можно направлять сообщения другим провайдерам служб и изменять содержимое и формат сообщений. Модули передачи могут предоставлять такие функции, как ведение протокола и обработка ошибок, которые отвечают вашим требования.

С помощью административной консоли можно изменять определенные аспекты модулей передачи без необходимости повторного развертывания модуля.

Компоненты модулей передачи

Модули передачи содержат следующие элементы:

- Импорты, которые определяют взаимодействия между модулями SCA и провайдерами служб. Они позволяют модулям SCA вызывать внешние службы так, как если бы они были локальными. Можно просматривать импорты модуля передачи и изменять привязку.
- Экспорты, которые определяют взаимодействия между модулями SCA и клиентами служб. Они позволяют модулю SCA предлагать службу и определять внешние интерфейсы (точки доступа) модуля SCA. Экспорты модуля передачи можно просматривать.
- Компоненты SCA, которые создают блоки для модулей SCA, таких как модули передачи. Модули и компоненты SCA можно создавать и настраивать графически с помощью Integration Designer. После развертывания с помощью административной консоли можно настраивать определенные аспекты модуля передачи без необходимости повторного развертывания модуля.

  Обычно модули передачи содержат специальный тип компонента SCA, называемого *компонентом потока передачи*. Компоненты потока передачи определяют потоки передачи.

  Компонент потока передачи может не содержать или содержать один или несколько примитивов передачи. IBM Business Process Manager поддерживает поставляемый набор примитивов передачи, который предоставляет функциональные возможности для маршрутизации и преобразования сообщений. Для получения дополнительных функциональных возможностей примитивов передачи можно применить пользовательский примитив передачи для вызова пользовательской логики.

  Назначением модуля передачи, который не содержит компонента потока передачи, является преобразование запросов служб из одного протокола в другой. Например, запрос службы может быть сделан с помощью SOAP/JMS, но может требовать преобразования в SOAP/HTTP перед отправкой.

  Примечание: Модули передачи можно просматривать и вносить определенные изменения в них с помощью IBM Business Process Manager. Однако IBM Business Process Manager не позволяет просматривать или изменять компоненты SCA в пределах модуля. Для настройки компонентов SCA используйте Integration Designer.

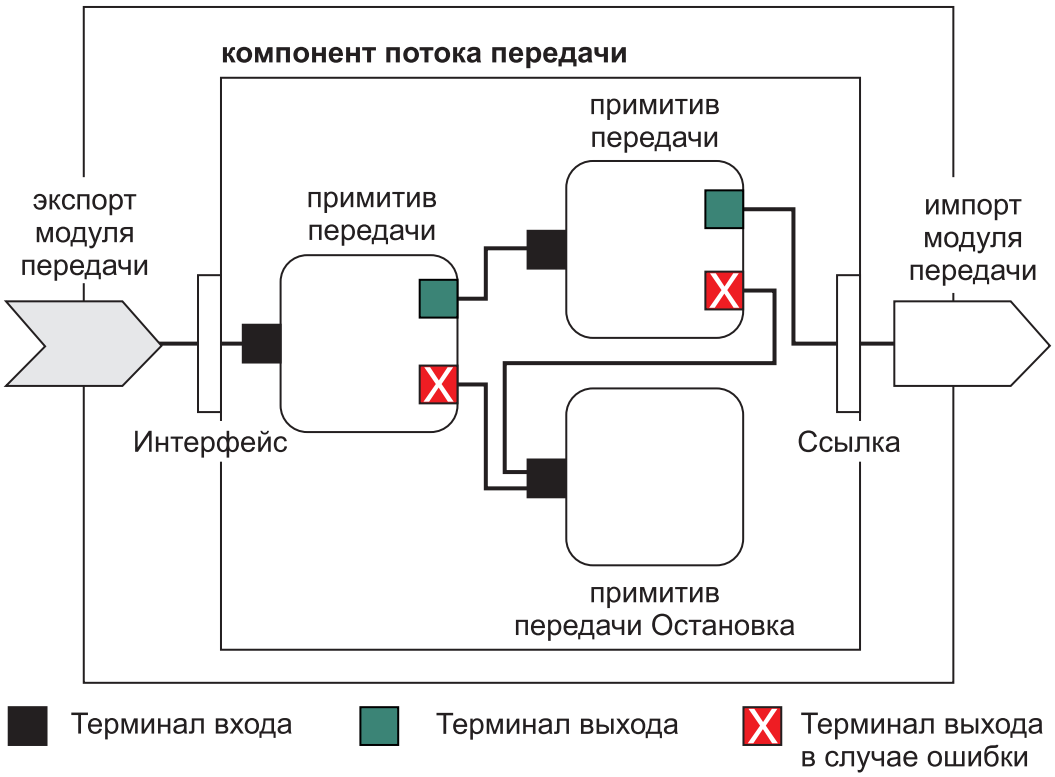


Рисунок 88. Упрощенный пример модуля передачи. Модуль передачи содержит один компонент потока передачи, который содержит примитивы передачи.

- Свойства

  Примитивы передачи имеют свойства, некоторые из которых могут отображаться в административной консоли как дополнительные свойства модуля SCA.

  В случае свойств примитива передачи в административной консоли IBM Business Process Manager разработчик интеграции должен открыть эти свойства. Некоторые свойства сами предоставляют себя для административной настройки, и Integration Designer описывает эти открываемые свойства, поскольку они могут быть открыты из цикла интеграции в административном цикле. Другие свойства не готовы к административной настройке, поскольку их изменение может повлиять на поток передачи таким образом, что модуль передачи потребует повторного развертывания. Integration Designer перечисляет свойства, которые можно выбрать для открытия в открытых свойствах примитива передачи.

  Административную консоль IBM Business Process Manager можно использовать для изменения значения открытых свойств без необходимости повторного развертывания модуля передачи или перезапуска сервера или модуля.

  Обычно потоки передачи используют изменения свойств сразу же. Однако если изменения свойств делаются в ячейке администратора развертывания, то они могут влиять на каждый узел. поскольку данный узел синхронизируется. Кроме того, потоки передачи, которые находятся в процессе, продолжают использовать предыдущие значения.

  Примечание: Из административной консоли можно изменять только значения свойств, но не группы, имена или типы свойств. При необходимости изменения групп, имен или типов свойств используйте Integration Designer.
- Модуль передачи или зависимая библиотека может также определять подпотоки. Подпоток представляет собой повторно используемую единицу логики интеграции и состоит из соединенных друг с другом примитивов передачи. Примитив можно добавлять в поток передачи для вызова подпотока.

Развертывание модулей передачи

Модули передачи создаются с помощью Integration Designer и обычно развертываются в IBM Business Process Manager внутри файла EAR.

Значения открытых свойств можно изменить во время развертывания.

Можно экспортировать модуль из Integration Designer и вызвать Integration Designer для упаковки модуля передачи в файл архива Java (JAR) и файл JAR внутри файла EAR. Затем можно развернуть файл EAR, установив новое приложение из административной консоли.

Модули передачи можно рассматривать как одну сущность. Однако модули SCA определяются несколькими файлами XML, хранящимися в файле JAR.

Пример файла EAR, содержащего модуль передачи

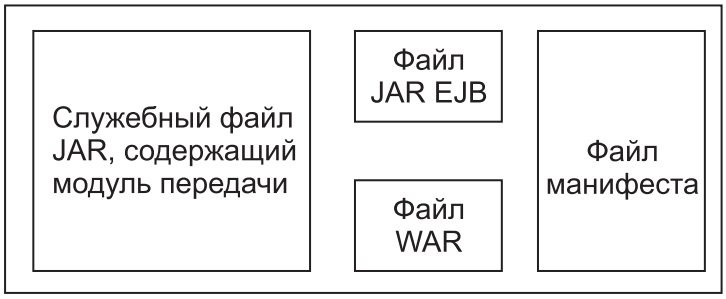


Рисунок 89. Упрощенный пример файла EAR, содержащего модуль передачи. Файл EAR содержит файлы JAR. Файл JAR утилит содержит модуль передачи.

Примитивы передачи

Компоненты потока передачи работают на потоках сообщений между компонентами служб. Возможности компонента передачи реализуются *примитивами передачи*, которые реализуют стандартные типы реализаций служб.

Компонент потока передачи имеет один или несколько потоков. Например, один для запроса и один для ответа.

IBM Business Process Manager поддерживает поставляемый набор примитивов передачи, которые реализуют стандартные возможности передачи для модулей передачи или модулей, развернутых в IBM Business Process Manager. Когда требуются специальные возможности передачи, можно развернуть свои пользовательские примитивы передачи.

Примитив передачи определяет “входящую” операцию, обрабатывающую сообщения, которые представляются объектами сообщений служб (SMO). Примитив передачи также может определять “исходящую” операцию, которая отправляет сообщения другому компоненту или модулю.

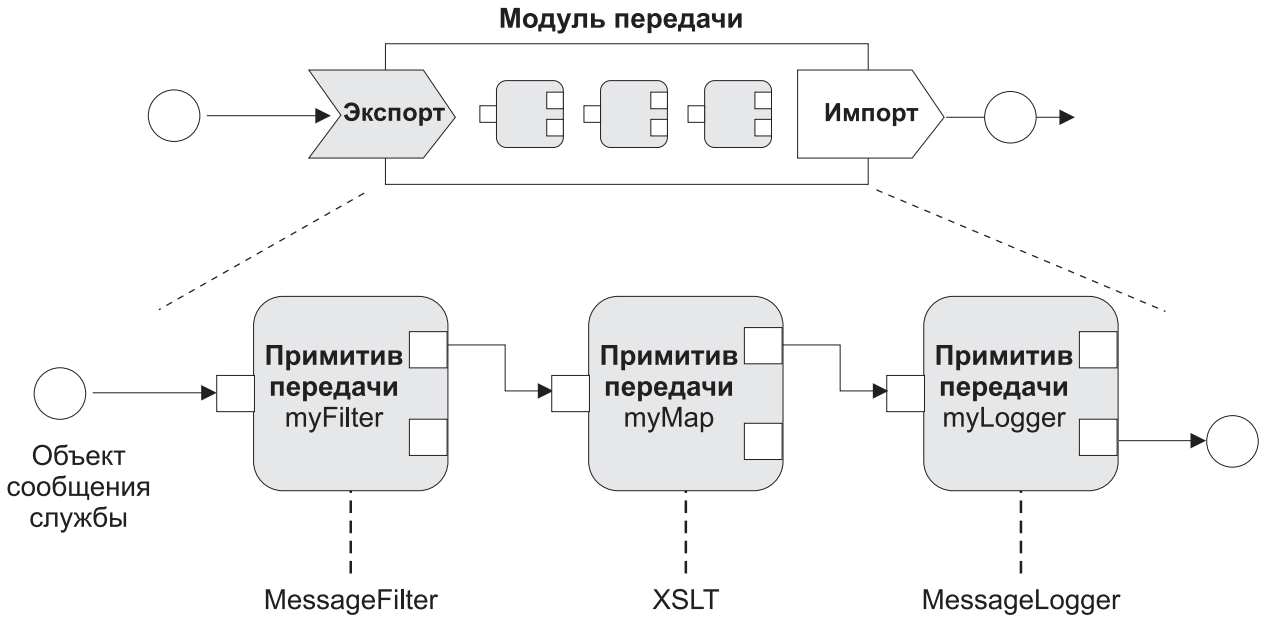


Рисунок 90. Модуль передачи, содержащий три примитива передачи

Integration Designer можно использовать для настройки примитивов передачи и задания их свойств. Некоторые из этих свойств можно сделать видимыми администратору среды выполнения открытием их. Любое свойство примитива передачи, которое может быть открыто, также может быть динамическим свойством. Динамическое свойство можно переопределять во время выполнения с помощью файла стратегий.

Integration Designer также позволяет графически моделировать и производить сборку компонентов потока передачи из примитивов передачи, а также собирать модули передачи или модули из компонентов потока передачи. Административная консоль ссылается на модули передачи и модули как на модули SCA.

Integration Designer также предусматривает определение подпотоков в модулях или их зависимых библиотеках. Подпоток может содержать любой примитив передачи за исключением примитива передачи обработки стратегий. Подпоток вызывается из потока запроса или ответа или из другого подпотока с помощью примитива передачи подпотока. Свойства, открытые из примитивов передачи в подпотоке, представляются как свойства на примитивах передачи подпотока. Они затем могут открываться снова, пока они не достигнут уровня модуля, на котором они затем могут быть изменены администратором среды выполнения.

Поддерживаемые примитивы передачи

IBM Business Process Manager поддерживает следующий набор примитивов передачи:

Карта бизнес-объектов

Преобразует сообщения.

- Определяет преобразования сообщений с помощью карты бизнес-объектов, которая может использоваться многократно.
- Позволяет определять преобразования сообщений графически с помощью редактора карты бизнес-объекта.
- Может изменять содержимое сообщения.
- Может преобразовать тип входящего сообщения в другой тип исходящего сообщения.

Пользовательская передача

Позволяет реализовать пользовательскую логику передачи в коде Java. Пользовательский примитив передачи объединяет гибкость определенного пользователем примитива передачи с простотой готового примитива передачи. Можно создавать сложные преобразования и шаблоны маршрутизации путем:

- Создания кода Java.
- Создания собственных свойств.
- Добавления новых терминалов.

Службу можно вызвать из пользовательского примитива передачи, а примитив передачи вызовов служб предназначен для вызова служб и предоставляет дополнительную возможность, такую как повторение.

Обработчик данных

Позволяет преобразовать часть сообщения. Он используется для преобразования некоторого элемента сообщения из физического формата в логическую структуру или из логической структуры в физический формат. Основным применением примитива является преобразование физического формата, такого как текстовая строка, в пределах объекта текстового сообщения JMS в логическую структуру бизнес-объекта и обратно. Эта передача обычно используется для следующего:

- Преобразование некоторого раздела входящего сообщения из определенной структуры в другую. Например, когда SMO включает строковое значение, отделенное запятой, и требуется проанализировать его в конкретном бизнес-объекте.
- Изменение типа сообщения. Для примера можно привести ситуацию, когда экспорт JMS настроен на использование привязки данных базового типа JMS, но в пределах модуля передачи разработчик интеграции решает, что это содержимое должно быть включено в конкретную структуру BO.

Поиск в базе данных

Изменяет сообщения, используя информацию из предоставленной пользователем базы данных.

- Необходимо настроить базу данных, источник данных и все параметры идентификации сервера для используемого примитива передачи Поиск в базе данных. В этом может помочь административная консоль.
- Примитив передачи Поиск в базе данных может выполнить чтение только одной таблицы.
- Указанный ключевой столбец должен содержать уникальное значение.
- Данные в столбцах значений должны быть либо простого типа схемы XML, либо типа схемы XML, расширенного простым типом схемы XML.

Поиск конечных точек

Предусматривает динамическую маршрутизацию запросов поиском конечных точек служб в хранилище.

- Информация о конечных точках служб извлекается из WebSphere Service Registry and Repository (WSRR). Реестр WSRR может быть локальным или удаленным.
- Изменения реестра делаются через административную консоль WSRR.
- IBM Business Process Manager необходимо знать, какой реестр используется, поэтому необходимо создать определения доступа WSRR с помощью административной консоли IBM Business Process Manager.

Отправитель событий

Расширяет мониторинг, позволяя отправлять события изнутри компонента потока передачи.

- Можно остановить действие передачи, выключив переключатель.
- Можно просмотреть события обработчика событий с помощью браузера событий общего формата IBM Business Process Manager.
- События необходимо отправлять только в значимую точку потока передачи из соображений производительности.

- Можно указать части сообщения, которые содержит событие.
- События отправляются в формате событий общего формата и отправляются на сервер инфраструктуры событий общего формата.
- Для полного использования информации отправителя событий получателям событий необходимо понимать структуру событий общего формата. События общего формата имеют общую схему, но это не модель конкретных данных приложения, которая содержится в расширенных элементах данных. Для моделирования расширенных элементов данных инструменты Integration Designer создают файл определений каталога событий инфраструктуры событий общего формата для каждого из настроенных примитивов передачи Отправитель событий. Файлы определений каталога событий являются артефактами экспортов, которые предоставляются в помощь; они не используются Integration Designer или средой выполнения IBM Business Process Manager. Файлы определений каталога событий должны указываться при создании приложений для использования событий отправителя событий.
- Можно указать другой мониторинг из IBM Business Process Manager. Например, можно отслеживать события, отправляемые из импортов и экспортов.

Ошибка

Останавливает некоторый конкретный путь в потоке и создает исключительную ситуацию.

Объединение

Позволяет объединять (комбинировать) сообщения.

- Может использоваться только с примитивом передачи Разъединение.
- Вместе примитивы передачи Разъединение и Объединение позволяют объединять данные в одном исходящем сообщении.
- Примитив передачи Объединение получает сообщения до точки принятия решения, затем отправляется одно сообщение.
- Для удержания объединенных данных должен использоваться общий контекст.

Разъединение

Позволяет разделять и объединять (комбинировать) сообщения.

- Вместе примитивы передачи Разъединение и Объединение позволяют объединять данные в одном исходящем сообщении.
- В режиме повторения примитив передачи Разъединение позволяет повторять одно входящее сообщение, которое содержит повторяющийся элемент. При каждом появлении повторяющегося элемента отправляется сообщение.
- Для удержания объединенных данных должен использоваться общий контекст.

Метод set заголовков HTTP

Предоставляет механизм управления заголовками в сообщениях HTTP.

- Может создавать, задавать, копировать или удалять заголовки сообщений HTTP.
- Может задавать несколько действий для изменения нескольких заголовков HTTP.

Преобразование

Преобразует сообщения.

- Позволяет выполнять преобразования расширяемого языка таблицы стилей (XSL) или преобразования по карте связей бизнес-объектов.
- Сообщения преобразуются с использованием преобразования XSLT 1.0, XSLT 2.0 или преобразования по карте связей бизнес-объектов. Преобразование XSL применяется к сообщению, сериализованному в формат XML; преобразование по карте связей бизнес-объектов применяется к объектам данных службы (SDO).

Метод set элементов сообщения

Предоставляет простой механизм задания содержимого сообщений.

- Может изменять, добавлять или удалять элементы сообщения.
- Не изменяет тип сообщения.

- Данные в столбцах значений должны быть либо простого типа схемы XML, либо типа схемы XML, расширенного простым типом схемы XML.

Фильтр сообщений

Направляет сообщения на различные пути на основании содержания сообщения.

- Можно остановить действие передачи, выключив переключатель.

Регистратор сообщений

Регистрирует сообщения в реляционной базе данных или через пользовательский регистратор. Сообщения сохраняются как XML, поэтому данные впоследствии могут обрабатываться приложениями XML.

- Можно остановить действие передачи, выключив переключатель.
- Схема реляционной базы данных (структура таблиц) определяется IBM.
- По умолчанию примитив передачи Регистратор сообщений использует общую базу данных. Среда выполнения преобразует источник данных в **jdbc/mediation/messageLog** в общую базу данных.
- Для настройки работы пользовательского обработчика можно настроить классы реализации Обработчика. Также для настройки работы пользовательского регистратора можно предоставить классы реализации программы форматирования, классы реализации фильтра или того и другого.

Метод set заголовков MQ

Предоставляет механизм управления заголовками в сообщениях MQ.

- Может создавать, задавать, копировать или удалять заголовки сообщений MQ.
- Может задавать несколько действий для изменения нескольких заголовков MQ.

Обработка стратегий

Обеспечивает динамическую настройку запросов путем поиска конечных точек служб и связанных файлов стратегий в хранилище.

- Файл стратегий можно использовать для динамического переопределения открытых свойств других примитивов передачи.
- Информация о конечных точках служб и информация о стратегиях извлекается из WebSphere Service Registry and Repository (WSRR). Реестр WSRR может быть локальным или удаленным.
- Изменения реестра делаются через административную консоль WSRR.
- IBM Business Process Manager необходимо знать, какой реестр используется, поэтому необходимо создать определения доступа WSRR с помощью административной консоли IBM Business Process Manager.

Вызов службы

Вызывает службу изнутри потока передачи вместо ожидания конца потока передачи и использования механизма вызова.

- Если служба возвращает ошибку, можно повторить вызов той же службы или вызвать другую службу.
- Примитив передачи Вызов службы является мощным примитивом передачи, который может использоваться сам по себе для простых вызовов служб или в комбинации с другими примитивами передачи для сложных передач.

Задание типа сообщения

Во время разработки интеграции позволяет рассматривать слаботипизированные поля сообщения как строготипизированные. Поле является слаботипизированным, если оно может содержать более одного типа данных. Поле является строготипизированным, если его тип и внутренняя структура известны.

- Во время выполнения примитив передачи Задание типа сообщения позволяет проверить, соответствует ли содержимое сообщения ожидаемым типам данных.

Метод set заголовков SOAP

Предоставляет механизм управления заголовками в сообщениях SOAP.

- Может создавать, задавать, копировать или удалять заголовки сообщений SOAP.
- Может задавать несколько действий для изменения нескольких заголовков SOAP.

Остановка
Останавливает некоторый конкретный путь в потоке без создания исключительной ситуации.

Фильтр типов
Позволяет направить сообщения по другому пути потока на основании их типа.

WebSphere eXtreme Scale - Получить
Можно получить информацию из среды кэша сервера eXtreme Scale.

- Можно найти значения в кэше и хранить их как элементы в сообщении с использованием ключа.
- Сочетая примитивы передачи Хранение и Получение eXtreme Scale, можно кэшировать ответы из системы базового сервера каталогов. Для будущих запросов не потребуется получение прав доступа к этой системе базового сервера каталогов.
- Необходимо создать определения eXtreme Scale с помощью административной консоли WebSphere ESB, чтобы можно было задать сервер eXtreme Scale для использования.

WebSphere eXtreme Scale - Сохранить
Можно сохранить информацию в среде кэша сервера eXtreme Scale.

- Можно сохранить информацию в кэше eXtreme Scale с помощью ключа и объекта.
- Сочетая примитивы передачи Сохранить и Получить eXtreme Scale, можно использовать примитив передачи Сохранить для сохранения данных в кэше, а примитив передачи Получить применять для получения данных, сохраненных ранее в кэше.
- Необходимо создать определения eXtreme Scale с помощью административной консоли WebSphere ESB, чтобы можно было задать сервер eXtreme Scale для использования.

Динамическая маршрутизация

Можно направлять сообщения различными путями, используя конечные точки, определенные во время интеграции, или конечные точки определенные динамически во время выполнения.

К динамической маршрутизации относятся два случая маршрутизации сообщений:

- Динамическая маршрутизация, в которой поток динамический, но все возможные конечные точки предопределены в модуле архитектуры компонентов служб (SCA).
- Динамическая маршрутизация, в которой поток динамический и выбор конечных точек также динамический. Конечные точки служб выбираются из внешнего источника во время выполнения

Динамический выбор конечной точки

Среда выполнения имеет возможность направлять сообщение-запрос и ответное сообщение в адрес конечной точки, определенный элементом заголовка сообщения. Этот элемент заголовка сообщения может обновляться примитивами передачи в потоке передачи. Адрес конечной точки может обновляться информацией из реестра, базы данных или информацией из самого сообщения. Маршрутизация ответных сообщений применяется, только когда ответ отправляется экспортом JAX-WS веб-службы.

Для того чтобы среда выполнения реализовывала динамическую маршрутизацию запроса или ответа, модуль SCA должен иметь заданное свойство `Использовать динамическую конечную точку при задании в заголовке сообщения`. Разработчики интеграции могут задать свойство `Использовать динамическую конечную точку при задании в заголовке сообщения` или могут открыть его (сделать видимым в среде выполнения), чтобы администратор мог задать его. Свойства модуля можно просмотреть в окне Свойства модуля. Для вызова окна выберите **Приложения > Модули SCA > Свойства модулей**. Разработчик интеграции дает имена-псевдонимы открытым свойствам, и эти имена показываются в административной консоли.

Реестр

IBM WebSphere Service Registry and Repository (WSRR) можно использовать для хранения информации о конечных точках служб и затем создавать модули SCA для извлечения конечных точек из реестра WSRR.

При разработке модулей SCA используйте примитив передачи Поиск конечных точек для предоставления возможности потоку передачи запроса реестра WSRR для поиска конечной точки службы или набора конечных точек служб. Если модуль SCA извлекает набор конечных точек, то он должен использовать другой примитив передачи для выбора предпочитаемой конечной точки.

Управление стратегией передачи запросов служб

С помощью стратегий передачи можно управлять потоками передачи между клиентами служб и поставщиками служб.

Для управления потоками передачи можно использовать стратегии передачи, хранящиеся в IBM WebSphere Service Registry and Repository (WSRR). Реализация управления стратегией служб в WSRR основана на Web Services Policy Framework (WS-Policy).

Для управления запросами служб с помощью стратегий передачи в реестре WSRR должны содержаться необходимые модули SCA и документы стратегии передачи.

Как прикрепить стратегию передачи к запросу службы

При разработке модуля SCA, который будет использовать стратегию передачи, в поток передачи необходимо включить примитив передачи Policy Resolution. Во время выполнения примитив передачи Policy Resolution получает информацию о стратегии передачи из реестра. Следовательно, для управления стратегией передачи запросов служб модуль SCA должен содержать компонент потока передачи.

В реестре стратегии передачи можно прикрепить к модулю SCA или целевой службе, применяемой модулем SCA. Прикрепленные стратегии передачи можно использовать для всех сообщений службы, обрабатываемых модулем SCA. К стратегии передачи можно прикрепить условия. Условия стратегий передачи определяют, в каком контексте следует применять ту или иную стратегию. Кроме того, с помощью классификации стратегий передачи можно задать состояние управления.

WebSphere Service Registry and Repository

Продукт WebSphere Service Registry and Repository (WSRR) позволяет хранить, использовать и управлять информацией о конечных точках служб и стратегиях передачи. WSRR может помочь сделать служебные приложения более динамичными и адаптируемыми к изменяющимся бизнес-условиям.

Введение

Потоки передачи могут использовать WSRR как динамичный механизм поиска, предоставляющий информацию о конечных точках служб или стратегиях передачи.

Для настройки доступа к WSRR необходимо создать документы определения WSRR с помощью административной консоли. Кроме этого, можно использовать административные команды WSRR из клиента сценариев wsadmin. Определения WSRR и его свойства соединения представляют собой механизм, предназначенный для подключения к экземпляру реестра и получения конечной точки службы или стратегии передачи.

Конечные точки служб

Можно использовать WSRR для хранения информации о службах, с которыми вы уже работаете или планируете работать, либо о которых должны быть осведомлены. Эти службы могут находиться в ваших или в других системах. Например, с помощью WSRR приложение может выполнять поиск наиболее подходящей службы для удовлетворения требований к функциональности и производительности.

При разработке модуля SCA, которому требуется доступ к конечным точкам служб из WSRR, необходимо включить в поток передачи примитив передачи Поиск конечных точек. В среде выполнения примитив-посредник Поиска конечной точки получает конечные точки служб из реестра.

Стратегии передачи

WSRR также можно использовать для хранения информации и стратегиях передачи. Стратегии передачи могут помочь управлять запросами служб путем динамического переопределения свойств модуля. Если WSRR содержит стратегии передачи, прикрепленные к некоторому объекту, представляющему ваш модуль SCA или вашу целевую службу, то стратегии передачи могут переопределять свойства модуля. Если требуется применять стратегии передачи в разных контекстах, то можно создать условия стратегий передачи.

Примечание: Стратегии передачи отвечают за управление потоками отображения и не имеют отношения к защите.

При разработке модуля SCA, который будет использовать стратегию передачи, необходимо включить в поток передачи примитив передачи Обработка стратегий. В среде выполнения примитив передачи Обработка стратегий получает информацию о стратегиях передачи из реестра.

WebSphere eXtreme Scale

С помощью продукта WebSphere eXtreme Scale (eXtreme Scale) можно получить систему кэширования, интегрируемую с приложением IBM Business Process Manager. С помощью eXtreme Scale с IBM Business Process Manager может улучшить значения времени ответа службы и ее надежность, а также предоставить дополнительные функции интеграции.

eXtreme Scale действует как эластичная масштабируемая сетка данных в памяти. Таблица данных динамически кэширует, делит на разделы, копирует данные приложений и бизнес-логику, а также управляет ими на нескольких серверах. С помощью eXtreme Scale можно также получить показатели QoS, такие как целостность транзакции, высокая готовность и предсказуемые значения времени ответа.

С помощью потоков передачи можно обратиться к функции кэширования eXtreme Scale, путем включения примитивов передачи WebSphere eXtreme Scale в свой поток. При разработке модуля архитектуры компонентов служб (SCA), которому требуется сохранять информацию в кэше eXtreme Scale, необходимо включить в поток передачи примитив передачи Сохранить WebSphere eXtreme Scale. Если требуется получить информацию из кэша eXtreme Scale, необходимо включить примитив передачи Получить WebSphere eXtreme Scale. Применяя сочетание этих двух примитивов передачи в потоке передачи, можно кэшировать ответ из системы базового сервера каталогов, с тем чтобы дальнейшие запросы могли получать ответ из кэша.

Для настройки доступа к eXtreme Scale необходимо создать определение WebSphere eXtreme Scale с помощью административной консоли. Вместо этого можно также использовать административные команды WebSphere eXtreme Scale из клиента сценариев wsadmin. Определение eXtreme Scale является тем механизмом, который примитивы передачи Получить и Сохранить WebSphere eXtreme Scale используют для соединения с сервером eXtreme Scale.

Клиенты службы сообщений

Клиенты службы сообщений доступны C/C++ и .NET для разрешения приложениям не Java соединения с Enterprise Service Bus.

Message Service Clients for C/C++ and .NET предоставляет API с именем XMS, который имеет тот же набор интерфейсов, что и API Java Message Service (JMS). Message Service Client for C/C++ содержит две реализации XMS: одну для использования приложениями C и другую для использования приложениями C++. Message Service Client for .NET содержит полностью управляемую реализацию XMS, которая может использоваться любым языком, совместимым с .NET.

Message Service Clients for .NET можно получить на http://www-01.ibm.com/support/docview.wss?rs=0&q1=IA9H&uid=swg24011756&loc=en_US&cs=utf-8&cc=us&lang=en

Message Service Clients for C/C++ можно получить на http://www-01.ibm.com/support/docview.wss?rs=0&q1=ia94&uid=swg24007092&loc=en_US&cs=utf-8&cc=us&lang=en.

Также можно установить и использовать поддержку клиента Java EE на WebSphere Application Server Network Deployment, включая клиент веб-служб, клиент EJB и клиент JMS.