Ссылка OCC

Сгенерировано: 2024-12-03 11:16:24 GMT+0000

SAP Коммерция | 2205

Публичный

Оригинальное содержание:https://help.sap.com/docs/SAP_COMMERCE/e5d7cec9064f453b84235dc582b886da?locale=en-US&state=PRODUCTION&version=2205

Предупреждение

Этот документ был создан на основе SAP Help Portal и является неполной версией официальной документации по продукту SAP. Информация, включенная в пользовательскую документацию, может не отражать расположение тем на SAP Help Portal и может не иметь важных аспектов и/или корреляций с другими темами. По этой причине он не предназначен для продуктивного использования.

Для получения более подробной информации посетите сайтhttps://help.sap.com/docs/отказ от ответственности .

Архитектура расширения OCC

Расширения OCC позволяют расширить функциональность OCC без установки дополнений.

Расширения OCC зависят откоммерциявеб-сервисы,и многие существующие надстройки OCC имеют соответствующие расширения OCC с той же бизнес-логикой и функциональностью. Имена расширений OCC заканчиваются на occ, напримерb2bocc,например.

Дополнительные возможности расширений OCC включают следующее:

- Расширения OCC не требуют дополнительных команд для установки. Они импортируются в веб-контекст Spring коммерциявеб-услуги.Нет необходимости копировать туда файлы.
- Расширения OCC имеют иную структуру каталогов, чем дополнения OCC, несмотря на то, что логика у них та же, что и у дополнения OCC.

Структура каталога расширения OCC

Расширение OCC имеет стандартную структуру каталогов расширений, но без веб-части, поскольку это не веб-расширение. Оно содержит те же файлы, что и дополнение OCC, но они расположены по-другому в структуре каталогов. Основные изменения перечислены в следующей таблице:

Структура каталога дополнений OCC (старая)	Структура каталога расширений OCC (новая)	Описание
/acceleratoraddon		Нетуускорительдополнениедиректо каталог в основном переносится на новый / ос /ресурсыкаталог расширения OCC новое имя расширения OCC. Например,x
/acceleratoraddon/web/src	/источник	Классы контроллеров REST и другие связанные расширения OCC.
/acceleratoraddon/web/webroot/WEB-INF/messages	/ресурсы/occ/v2/*<имя_расширения>*/сообщения	Локализованные сообщения расширения OCC
/acceleratoraddon/web/webroot/WEB-INF/lib	/lib	НетуWEB-INF/libкаталог в O main /либ Каталог. /либкаталог /acceleratoraddon/web/webroot
/ресурсы/*<имя_дополнения>*/импорт	/ресурсы/impex	Файлы ImpEx в расширении OCC не загружаются в конфигурацию соглашения. Для получения дополнительной информации Данные проекта .
/ресурсы/*<имя_дополнения>*/веб/весна	/ресурсы/occ/v2/*<имя_расширения>*/веб/весна	Определения весенних бобов расширения OCC /ресурсы/occ/v2/*<расширение_* веб-весна.xmlле импортируются в

Расширения OCC не являются веб-расширениями

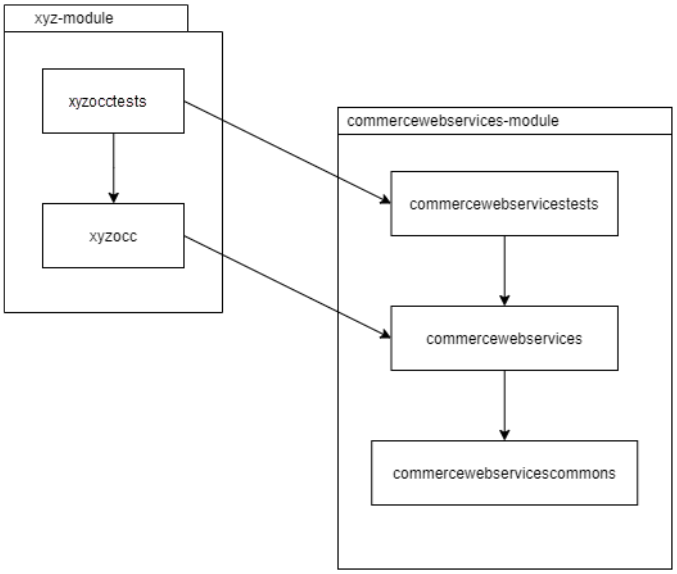
Расширение OCC не является веб-расширением и не имеет веб-контекста. Оно расширяет функциональность OCC, добавляя новые контроллеры REST, бины Spring и локализованные сообщения. Классы контроллеров REST и другие классы расширения OCC находятся в основном /источникВ Spring XML-конфигурации каталога коммерциявеб-услугирасширение, есть оператор импорта, который загружает определения компонентов из всех расширений OCC.

```
<import resource="classpath*:/occ/v2/*occ/web/spring/*-web-spring.xml"/>
```

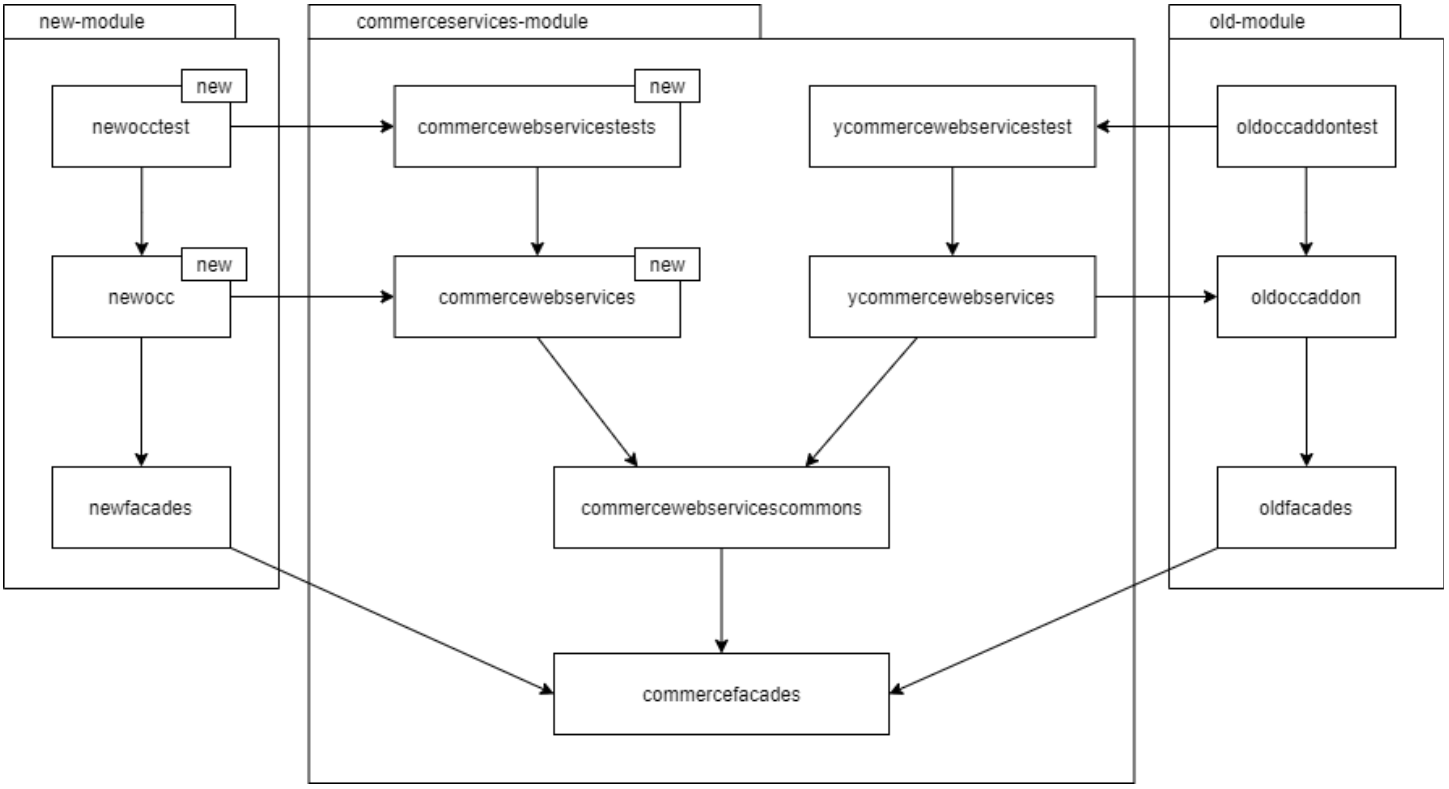
Оператор импорта файла конфигурации XML, содержащий определения компонентов, должен соответствовать следующим требованиям:

- Файл конфигурации должен быть расположен в папке /ресурсы/occ/v2/*<*окк*>*/веб/веснакаталог, где*<*окк*>*указывает на каталог с тем же именем, что и расширение OCC, которое заканчивается на occ, напримерxyzocc.
- Имя файла конфигурации должно заканчиваться на -web-spring.xml.

Зависимости между расширениями



ПримерxyzoccРасширение OCC зависит откоммерциявеб-услуги.Классы расширения OCC находятся в /источниккаталог и не может наследовать от классов в /веб/источниккаталогкоммерциявеб-услуги.По сравнению с OCC-аддоном зависимость обратная. OCC-аддон, напротив, зависит отycommercevеб-сервисыпоскольку исходные файлы веб-дополнения OCC копируются в ycommercevеб-сервисывеб-каталог исходного кода.



Перекрывающиеся пути

При использовании установки надстройки OCC можно переопределить конечные точки контроллера. Используя @Запрос переопределения сопоставления,конечная точка контроллера дополнения может переопределять конечные точки других расширений. Для получения дополнительной информации см.Расширение REST API .

Однако для расширения OCC все конечные точки контроллера должны иметь уникальные пути. Любые конечные точки, которые находятся в конфликте с существующими конечными точками, должны быть изменены, чтобы иметь уникальный путь. Свойство конфигурацииocc.rewrite.overlapping.paths.enabledможет быть использовано в качестве решения. Когда это свойство установлено вистинный,указанное расширение должно предоставлять уникальный путь для конечных точек контроллера, который может конфликтовать с конечной точкой в других расширениях. Например: /*<baseSiteId>*/пользователи/*<userId>*/заказыстановится /*<baseSiteId>*/orgUsers/*<userId>*/заказы

Ограничения канала сайта

Существует ограничение, которое предотвращает несанкционированный доступ с одного сайта к конечным точкам другого сайта. Например, оно может предотвратить вызов конечных точек B2B в контексте сайтов B2C. Вы можете применить ограничение к конечным точкам контроллера, используя аннотацию. Эта аннотация содержит свойство конфигурации, используемое для чтения набора разрешенных каналов сайта для этой конечной точки.

OCC Дополнительный Конвертер

OCC AddOn Converter — это инструмент с интерфейсом командной строки, позволяющий конвертировать OCC AddOns в OCC Extensions.

Архитектура расширений OCC позволяет расширить функциональность OCC без установки каких-либо дополнений OCC. В результате, дополнения OCC необходимо преобразовать в расширения OCC. Для получения дополнительной информации см.Архитектура расширения OCC .

Использование OCC AddOn Converter

Перед запуском скрипта убедитесь, что установлена Java. Для получения дополнительной информации о версии Java см.Прежде чем начать .

Чтобы использовать OCC AddOn Converter, перейдите в раздел <HYBRIS_HOME>/build-tools/occ-addon-converterкаталог. Вы можете использовать скрипт непосредственно из исходного кода или запустить его как упакованный JAR-файл. Смотрите следующие инструкции по запуску скрипта:

Использование OCC AddOn Converter в качестве JAR-файла

Выполните следующие команды:

- Для систем Microsoft Windows:

```
gradlew.bat jar;

java -jar сборка\libs\occ-addon-converter.jar
- - addondir "C:\Projects\xyoccaddon"
- - extdir "C:\Projects\xyzocc"
- - stepsdir "C:\Projects\customSteps" & :: stepsdir необязателен
```

- Для систем на базе Unix (таких как Linux или Mac OS):

```
. /gradlew jar;

java -jar сборка/libs/occ-addon-converter.jar \
- - addondir "/Projects/xyoccaddon" \
- - extdir "/Projects/xyzocc" \
- - stepsdir "/Projects/customSteps" # stepsdir необязателен
```

Использование AddOn Converter из исходного кода

Выполните следующие команды:

- Для систем Microsoft Windows:

```
gradlew.bat запустить --args="--addondir ""C:\Projects\xyoccaddon"" --extdir ""C:\Projects\xyzocc"" --stepsdir "
```

- Для систем на базе Unix (таких как Linux или Mac OS):

```
. /gradlew запустить --args="--addondir \"/Projects/xyoccaddon\" --extdir \"/Projects/xyzocc\" --stepsdir \"/Proj
```

Ниже приводится объяснение параметров:

- ксиоккаддонвид имя OCC AddOn.
- xyzoccspecies имя нового расширения OCC. Имена расширений OCC должны заканчиваться на «occ».
- addondir— параметр, задающий каталог OCC AddOn.
- extdir— параметр, который определяет каталог, в котором находится новое расширение OCC после преобразования. Каталог стирается каждый раз при использовании конвертера.

- stepsdir— необязательный параметр, который наполняет каталог пользовательскими шагами, которые вы можете указать для применения дополнительных изменений.

Архитектура скрипта

OCC AddOn Converter загружает и запускает скрипты Groovy, которые выполняют преобразование. Шаги преобразования включают перемещение, переименование и редактирование файлов. Шаги анализируются без необходимости остановки сервера. Если преобразование данного AddOn требует дополнительных шагов, используйте --stepsdir
ag для указания каталога с вашими пользовательскими шагами. При запуске скрипта регистрируются имена файлов шагов по умолчанию и пользовательских шагов.

Соглашение об именах файлов

OCC AddOn Converter ожидает, что имена файлов шагов начинаются сШаг_.Нумеруйте ваши шаги в соответствии с порядком, в котором вы хотите, чтобы они выполнялись. Независимо от того, находится ли шаг в основном скрипте или в ваших пользовательских шагах, скрипт упорядочивает их по именам файлов, так что
Step_15_Foo.заводнойпроходит между шагамиШаг_10*иШаг_20*.

Написание индивидуального шага

Механизм привязки Groovy в OCC AddOn Converter внедряет следующие две неявно определенные переменные во время выполнения:

- *<муравей>*является примеромAntBuilderобъект. Для получения дополнительной информации об использованииAntBuilder,видетьНаписание сценариев задач Ant .
- *<ctx>*— это контекстный объект с информацией о заданной операции.

The*<ctx>*переменная состоит из следующих элементов:

- *<ctx.ИМЯ_АДДОН>*— это строка с именем дополнения OCC, которое преобразуется.
- *<ctx.ИМЯ_РАСШИРЕНИЯ>*— это строка с именем целевого расширения OCC.
- *<ctx.ADDON_DIRECTORY>*— это строка с каталогом дополнения.
- *<ctx.EXTENSION_DIRECTORY>*— это строка с целевым каталогом нового расширения OCC.

Например, если вы хотите, чтобы ваше новое расширение OCC ссылалось на пакеты из другого модуля, выполните следующий шаг для изменения настроек импорта:

```
ant.echo("Преобразование: $ctx.ADDON_NAME в $ctx.EXTENSION_NAME")
ant.replace(dir: ctx.EXTENSION_DIRECTORY, токен: 'import com.example.old.occ.module', значение: 'import com.exampl
```

Если вы добавите этот код в файл, расположенный в каталоге, на который ссылается --stepsdirag, OCC AddOn Converter заменяетимпорт
com.example.old.occ.modulesимпорт com.example.new.occ.moduleво всех соответствующих файлах модуля.

После преобразования

При создании нового расширения OCC рекомендуется просмотреть вывод и запустить тесты для нового расширения. Расширение должно использоваться с модулем Commerce Web Services. Вы не можете использовать только что созданное расширение OCC ни sucommerceвеб-сервисы расширение или OCC AddOn, из которого было создано расширение.

Архитектура дополнений OCC

OCC — это набор расширений, предоставляющих веб-сервисы RESTful, ориентированные на коммерцию. Дополнения OCC расширяют OCC новыми функциями.

-Осторожность

Эта страница относится к программному обеспечению, которое было устарело как часть Accelerator UI и устаревших расширений шаблонов OCC. Для получения дополнительной информации см. Прекращение поддержки пользовательских интерфейсов Accelerator и старых расширений шаблонов OCC .

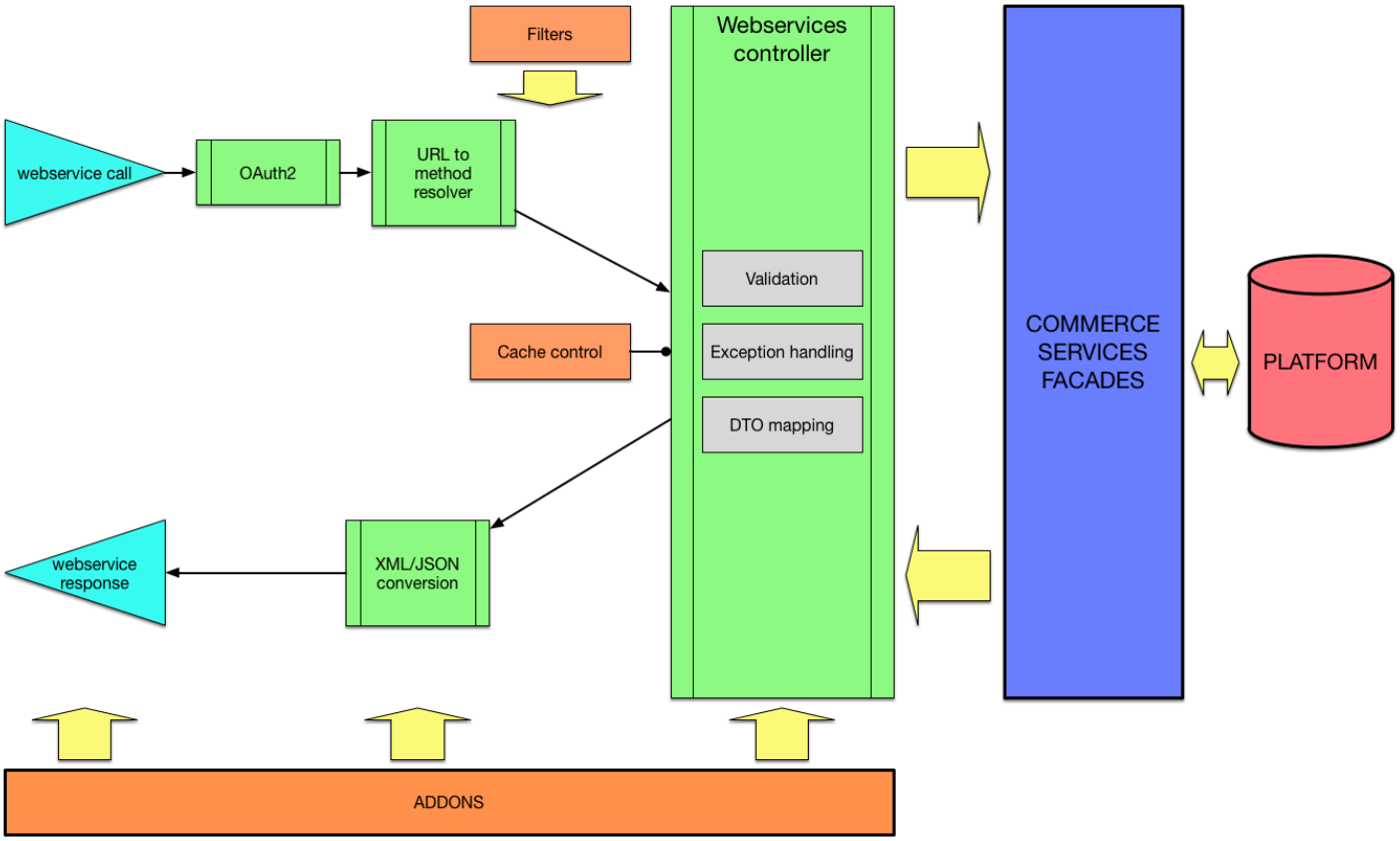
OCC предоставляет веб-сервисам согласованный способ взаимодействия с платформенным уровнем коммерции. Его главная цель — раскрыть существующую функциональность уровня коммерции миру веб-сервисов. Веб-сервисы OCC максимально прозрачны, но содержат некоторые дополнительные элементы, такие как отдельный кэш, механизм обработки исключений, фреймворк аутентификации и сопоставитель атрибутов.

Расширения коммерческих услуг

Модуль Commerce Services содержит следующие расширения:

- коммерцияВеб-сервисыcommonsрасширение
- ycommerceВеб-сервисырасширение
- ycommercewebservicestestрасширение

Некоторые из этих расширений представляют собой шаблоны, которые можно настраивать в соответствии с конкретными требованиями.ycommercewebservicesaddonsпапка содержитускорительВеб-сервисыдополнениеAddOn. Этот конкретный AddOn зависит отуслуги ускорителярасширение и требуется для обеспечения расширенной функциональности платежей с использованием сервисов КИС.



Расширение ycommercewebservices

Это основное расширение шаблона модуля OCC. Самая важная часть этого расширения — расширенное приложение веб-сервисов, построенное на фреймворке Spring MVC. Вызовы к определенным ресурсам выполняются методом с использованием запроса к контроллеру. Стандартный поток выглядит следующим образом:

. Запрос поступает в контроллер веб-сервисов коммерции и в большинстве случаев передается напрямую фасадам коммерции (иногда требуется дополнительная проверка). Он поддерживает следующие типы запросов:

  - GET — запрос данных, который запускает фасадные методы, которые ищут и извлекают нужную информацию.
  - POST, PUT, PATCH - запросы на создание и обновление элементов, которые могут быть отправлены как отдельные параметры URL или с использованием подхода RequestBody. Запросы на создание/обновление обычно дополнительно проверяются в модуле OCC.
  - HEAD — запрос, который можно использовать для получения только информации о количестве запрошенных элементов — это количество указывается в заголовке ответа.

После извлечения объекта данных из коммерческих фасадов он преобразуется в заранее определенные DTO (таким образом веб-сервисы изолируются от изменений объектов данных на коммерческом уровне).

Затем объекты данных отправляют ответ на вызов в формате XML или JSON (при условии отсутствия ошибок или исключений).

Расширение также имеет некоторые дополнительные функции.

Локальный кэш построен на основе Ehcache. Кэш включается на определенных вызовах контроллера - те, которые кэшируются, можно найти в de.hybris.platform.ycommercewebservices.v2.helperупаковываются как вспомогательные классы и помечаются знаком @Кэшируемыйаннотация. Параметры конфигурации кэша можно найти вehcache.xmlle. Где это было возможно, кэш включался непосредственно на контроллере

методы. Кроме того, многие из вызовов контроллера имеют @CacheControlаннотация. Это директива в заголовке ответа, которая используется клиентом или прокси-сервером и управляет тем, как возвращаемые данные должны кэшироваться.

Theycommerceвеб-сервисыРасширение использует механизм аутентификации на основе решения OAuth2 framework, реализованного в платформе. Для получения дополнительной информации об OAuth2 см.OAuth 2.0 .

Кроме того, есть несколько фильтров и перехватчиков, которые используются для различных целей, таких как управление кэшем, базовый сайт и проверка клиентов, настройка контекста запроса и атрибутов сеанса. Хотя версия 2коммерциявеб-услугине имеет состояния, в рамках одного запроса все равно создается и используется сеанс.

Расширение commercewebservicescommons

Thекоммерциявеб-сервисыcommonsextension — единственное расширение в модуле OCC, которое не является шаблоном. Это расширение содержит управление кэшем веб-сервисов, сопоставление данных, определения ошибок и некоторые другие элементы, которые не соответствуют расширениям шаблонов. Хотя это расширение не является шаблоном, оно все равно распространяется вместе с исходными кодами. Затем его можно легко расширить или изменить с помощью пользовательского AddOn. Несколько новых типов платформ были определены в расширении commons, и все они связаны с постоянным хранилищем токенов OAuth2, которое было разработано для обеспечения безупречной аутентификации в кластеризованной среде. Кроме того, все определения и сопоставления компонентов DTO между типами моделей платформ и объектами DTO, представленными в веб-сервисах OCC, размещены в этом расширении.

Расширение ycommercewebservicestest

Хотя тесты junit и интеграции хранятся вкоммерциявеб-услугирасширение, расширенный набор тестов коммерциявеб-услугимодуль хранится в отдельном расширении, называемомуcommercewebservicestestрасширение. Это расширение содержит собственный набор тестовых данных, используемый во время выполнения теста. Тесты дляycommerceвеб-сервисыРасширение написано на Groovy. Это модульные тесты, которые выполняют различные операции на тестируемых ресурсах URL и проверяют результаты. Использование тестов Groovy удобно для тестирования веб-сервисов, поскольку тестируются не только отдельные вызовы, но и сложные потоки, такие как процесс оформления заказа.

Существует отдельный тестовый набор для версии 1 и версии 2 веб-сервисов. Последний содержит тестовый набор, написанный на spock, тестовом фреймворке для приложений java и groovy. Тесты Spock имеют немного более высокий уровень абстракции, чем чистые тесты groovy, но их также проще поддерживать, и в конечном итоге они обеспечивают более быстрый способ тестирования приложения веб-сервисов.

Конгурации, действия и волшебники проводятся вбэк-офисрасширение.

ycommercewebservices Дополнения

Thекоммерциявеб-услугиМодуль совместим с AddOns. Идея та же, что и для SAP Commerce Accelerator - с помощью AddOn вы можете добавить новую/изменить существующую функциональностькоммерциявеб-услугирасширение. Преимущество создания AddOn (вместо изменения расширения) заключается в том, что обновление до новой версии более удобно, поскольку не вносятся изменения в коммерциявеб-услугирасширения, не нужно будет переносить никакие настройки. В настоящее время дистрибутив OCC содержит только ускорительвеб-сервисыдополнениеДобавить.

Коммуникация на основе REST

Связь следует модели веб-сервисов на основе REST.ycommerceвеб-сервисыШаблон разработан в двух версиях: v1 и v2. OCC v2 является версией по умолчанию.

-Примечание

Более подробную информацию о разнице между этими версиями см.v1 и v2 в ycommercewebservices .

Создание надстройки для веб-сервисов OCC

Узнайте, как создать дополнение для веб-служб OCC.

-Осторожность

Эта страница относится к программному обеспечению, которое было устарело как часть Accelerator UI и устаревших расширений шаблонов OCC. Для получения дополнительной информации см. Прекращение поддержки пользовательских интерфейсов Accelerator и старых расширений шаблонов OCC .

Обзор

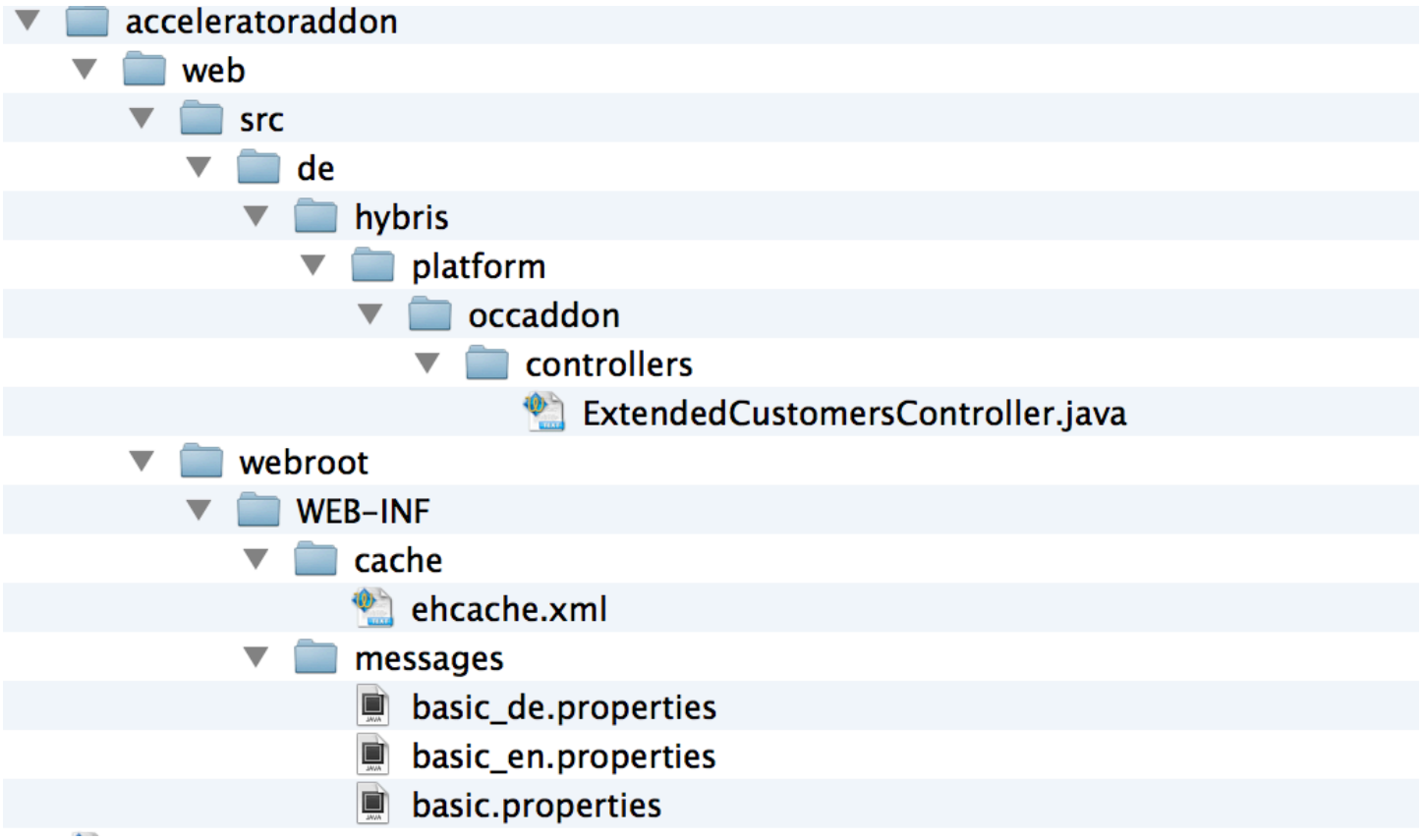
Theycommerceвеб-сервисырасширение раскрывает часть фасадов Commerce как веб-сервисы на основе REST. Поскольку веб-сервисы Commerce основаны на стандартном Spring MVC, вы можете легко настраивать или расширять их, создавая новые дополнения.

Структура дополнения

Структура AddOn почти такая же, как и у любого другого обычного расширения. Есть определенные каталоги и файлы, о которых вам следует знать:

Имя каталога/файла	Замечания
<имя_дополнения>-web-spring.xml	Этот файл должен быть создан в каталоге ресурсов, например ресурс\occaddon\web\spring\occaddon-web-spring.xml. Кроме того, он должен содержать веб-контекст AddOn, который добавляется к веб-контексту веб-служб OCC.
проект.свойства.шаблон	Этот файл должен быть создан, чтобы вы могли использовать скрипт установки AddOn. Это шаблон для файла project.properties, который содержит свойства конфигурации. Эти свойства создаются в процессе установки. -Примечание Подробную информацию об установке AddOn для витрины магазина см.Установка дополнения для определенного магазина
\acceleratoraddonкаталог	Структура папки отражает структуру папок, содержащих веб-компоненты обычного расширения.

Структура AddOn выглядит следующим образом:



На этапе сборки система автоматически копирует файлы из \ускорительдополнениекаталог к целевому расширению, в нашем случае усommerceвеб-сервисырасширение. Файлы копируются в следующие два каталога:

Имя каталога	Описание
\web\addonsrc	Содержит исходный код.

Имя каталога	Описание
\web\webroot\WEB-INF\<resourceDir>\addons	Содержит различные веб-ресурсы, такие как пакет сообщений (когда <resourceDir> = сообщения)или конфигурация кэша (<resourceDir> = кэш).Аналогичные каталоги могут быть созданы и для различные компоненты интерфейса, такие как JSP, HTML, изображения, файлы *.tag, но в случае дополнений для OCC эта часть может не потребоваться.

Структураусommerceсевеб-сервисыс дополнениями выглядит следующим образом:

Создание дополнения

В разделе ниже приведены инструкции по созданию дополнения.

Создание дополнения из шаблона

Чтобы создать AddOn из шаблона, выполните следующие действия:

. Добавьте следующие расширения к вашемуlocalextensions.xmlле:

a.

- поддержка дополнений:Расширение имеетмуравейскрипт, необходимый для корректной сборки дополнений.
- йоккаддон:Базовый шаблон, используемый для создания дополнений для коммерческих веб-сервисов.

. Создайте AddOn, используяекстгенийоккаддоншаблон расширения.

-Примечание

Подробности см.Создание нового расширения иРасширение йоккаддона .

. Добавьте новое расширение кconfig\localextensions.xmlле:

```
<расширения>
    . . .
    <расширение    имя="оккаддон"/>
</расширения>
```

Расширение REST API

Чтобы выявить новые вызовы, вам необходимо определитьКонтроллеркласс с соответствующими методами. АКонтроллердолжны быть созданы в \acceleratoraddon\web\srcкаталог. Если он создан в пакете: <addonPackage>.controllers (например de.hybris.platform.occaddon.контроллерызатем он автоматически добавляется в веб-контекст Spring Commerce Web Services.

-Примечание

Если контроллер не создан в <Пакет дополнений>.controllersпакет, контекст Spring Web должен быть настроен, как описано в СозданиеWebSpringContext раздел.

```
@Контроллер
@RequestMapping(value = "/{baseSiteId}/newResource") публичный
класс NewController
{
    @RequestMapping(метод      =  (МетодЗапроса.GET)
    @ResponseBody
    публичный NewResourceWsDTO {    получитьНовыйРесурс()

        вернуть новый NewResourceWsDTO("newSampleResource");
    }
}
```

После установки дополнения с таким определенным контроллером должна быть возможность вызывать следующие запросы:

- https://localhost:9002/rest/v1/{baseSiteId}/newResourceдля версии 1
- https://localhost:9002/rest/v2/{baseSiteId}/newResourceдля версии 2

-Примечание

Подробную информацию о более сложном сценарии расширения см.Расширение коммерческих веб-сервисов .

Установка надстройки для Commerce Web Services

Theaddoninstallскрипт добавляет запись AddOn врасширениеinfo.xmlCommerce Web Services и генерирует свойства проектале.

-Примечание

Подробную информацию об установке AddOn для витрины магазина см.Установка дополнения для определенного магазина .

Перед запуском установки убедитесь, что:

. поддержка дополнениярасширение указано вlocalextensions.xmlle или находится в каталоге, который загружается автоматически. Они определяются следующей записью: <путь autoload="true" dir= ... />

. ТвойДобавитьиусommerceвеб-сервисырасширения перечислены вlocalextensions.xmlle или находятся в каталоге, который загружается автоматически.

. Вы правильно определилипроект.свойства.шаблонle для дополнения.

Установка дополнения для определенного расширения

. Запуститьмуравейзадача называетсяaddoninstall.

а. Дляоккаддониусommerceвеб-сервисыкомандная строка выглядит следующим образом:

```
ant addoninstall -Daddonnames="occaddon" -DaddonStorefront.ycommercewebservices="ycommercewebservice
```

б. Дляоккаддонитусommercewebservicesкоманда выглядит следующим образом:

```
ant addoninstall -Daddonnames="occaddon" -DaddonStorefront.ycommercewebservices="mycommercewebservic
```

. После успешного завершения работы скрипта перестройте систему.

Сопутствующая информация

Генерация бинов и перечислений

ycommercewebservices Локальное обслуживание СМИ

Платформа оснащена двумя фильтрами для обслуживания местных СМИ:

- МедиаФильтр:настроенный ввеб.xmlфайл веб-приложения mediaweb; любые запросы на незащищенные медиа (/СМИконечная точка) обрабатываются этим фильтром,
- SecureMediaFilterнастраивается отдельно для каждого веб-приложения (его следует добавить вPlatformFilterChain).

Такой подход приводит к несогласованности, поскольку незащищенные медиа обрабатываются приложением mediaweb, тогда как защищенные обрабатываются отдельно каждым веб-приложением, которому они нужны. Решением этого являетсяWebAppMediaFilter,что устраняет несоответствие, обслуживая как незащищенные, так и защищенные носители, однако при условии, что каждое веб-приложение настраивается отдельно.

Обновление URL-адреса изображения

Theycommercewebservices остальные услуги / продуктыЗапрос конечной точки предоставляет ImageData атрибут URL, который был сгенерирован LocalMediaWebURLStrategy (начинается с /СМИ).Для обеспечения совместимости с нашимиWebAppMediaFilters,нам нужно было изменить URL, добавив отдельные решения для V1 и V2.

Конвертер объектов данных V1

Для V1 мы добавили нашу реализацию XStreamSingleValueConverter,который просто добавляет /отдых/v1префикс для существующего атрибута URL ценностьДанные изображения.

Объектный картограф V2 WsDTO

Сопоставление V2 DTO было реализовано с помощью картографов Orika. Для достижения нашей цели мы добавили реализациюAbstractCustomMapper, что добавляет /отдых/v2префикс для значения атрибута URL.

Пример тела ответа GET /продукты/123?поля=код,изображения(url)

```
{
    "код": "123",
    "изображения" : [ {
        "url" : "/rest/v2/medias/?context=..."
    } ]
}
```

Медиа-обслуживание

Обновив URL в теле ответа, нам пришлось настроитьWebAppMediaFilterдляycommerceвеб-сервисыдля обработки медиа-запросов V1 или V2. Чтобы справиться с этим, мы зарегистрировали два отдельныхWebAppMediaFilterвесенние бобы и добавили их в цепочки фильтров для обоих ycommerceвеб-сервисысервлеты (V1 и V2).

OCC вызывает безопасность

Безопасность вызовов OCC обеспечивается гибко настраиваемыми механизмами безопасности Spring.

Общая информация

Вызовы OCC защищены стандартными, высоконастраиваемыми механизмами безопасности Spring. Пользователь, который получает доступ к приложению, называется принципалом. Это не обязательно должен быть реальный пользователь, это может быть внешняя система, например, бэкэнд- или фронтэнд-приложение, или мобильное приложение. Важно различать аутентификацию и авторизацию:

- Аутентификация означает проверку предоставленных учетных данных. Если учетные данные действительны, то принципалу назначаются соответствующие роли.
- Авторизация означает принятие решения о том, может ли принципал выполнить данное действие. Это определяется на основе назначенных ролей принципала, а также других ограничений, например, защищенного канала связи.

Для упрощения аутентификации и авторизации OCC использует стандартный протокол OAuth2. Основная цель — обеспечить долгосрочный доступ к принципалу и дифференцировать правила безопасности в зависимости от типа клиентского приложения.

Процесс авторизации происходит отдельно в два этапа:

- HTTP-уровень
- Уровень обслуживания (бизнеса)

Каждый уровень применяет свой собственный набор правил и ограничений.

Роли пользователей OCC

Безопасность вызовов OCC основана в основном на ролях пользователей. Эти роли назначаются принципалу в зависимости от типа аутентификации:

Тип аутентификации	Возможные роли
Анонимный	Неаутентифицированному субъекту по умолчанию назначается встроенная роль АНОНИМНЫЙ.
Клиенты	Каждому клиентскому приложению, которое было аутентифицировано с использованием токена OAuth2 в потоке учетных данных клиента, назначается определенная роль в зависимости от определения клиента. При определении клиентов не забудьте назначить им либо ROLE_CLIENT, либо ROLE_TRUSTED_CLIENT, поскольку эти роли разрешают клиенту доступ кycommerceвеб-сервисырасширение. Подробности см.:Настройка клиентов OAuth .
Клиенты	Пользователям, которые были аутентифицированы с помощью токена OAuth2 в потоке паролей, назначается список ролей, которые получаются из сервисного уровня таким же образом, как это работает во всем приложении. По умолчанию используются роли CUSTOMERGROUP и CUSTOMERMANAGERGROUP.
Гости	Анонимные пользователи, которые указали свой адрес электронной почты. Это можно сделать звонок /клиенты/текущий/гостевой входв v1 или /users/anonymous/carts/{guid}/email в v2. Для таких пользователей встроенный в роли ГОСТЬ назначена.

Конфигурация пружины безопасности

Конфигурация сервера ресурсов OAuth2 и другие аспекты безопасности определены в /ycommercewebservices/web/webroot/WEB-INF/config/common/security-spring.xmlФайл и файлы конфигурации, специфичные для версии веб-сервисов: /ycommercewebservices/web/webroot/WEB-INF/config/v2/security-v2-spring.xml, /ycommercewebservices/web/webroot/WEB-INF/config/v1/security-v1-spring.xml.

Конфигурация OAuth хранится в платформе. Вы можете настроить параметры сервера в проекте. свойства файлаoauth2расширение. Подробности см.OAuth2 .

Чтобы Spring Security заработал, вам нужно добавитьspringSecurityFilterChainк цепочке фильтров веб-сервисов, как показано ниже в конфигурации для веб-сервисов v1 и v2.

версия 1 (/ycommercewebservices/web/webroot/WEB-INF/config/v1/filter-config-v1-spring.xml)

```
. . .
    <bean id="commerceWebServicesFilterChainV1"          class="de.hybris.platform.servicelayer.web.PlatformFilterChain
        <конструктор-аргумент>
            <ref bean="commerceWebServicesFilterChainListV1" </          />
        constructor-arg>
```

```
        </боб>
        <alias name="defaultCommerceWebServicesFilterChainListV1" alias="commerceWebServicesFilterChainListV1" /> <util:list
        id="defaultCommerceWebServicesFilterChainListV1">
            <!-- универсальные фильтры платформы -->
            <ref    bean="log4jFilter" />
            <реф  bean="tenantActivationFilter" />
            <реф  bean="sessionFilter" />
            <!-- фильтры commerceWebservices --> <ref
                bean="commerceWebServicesBaseSiteFilterV1" />
            <реф  bean="commerceWebServicesSessionAttributesFilterV1" />
            <реф  bean="baseSiteCheckFilterV1" />
            <!--    Безопасность -->
            <реф  bean="springSecurityFilterChain" />
            <реф  bean="guestRoleFilterV1" />
        </util:list>
    . . .
```

версия 2 (/ycommercewebservices/web/webroot/WEB-INF/config/v2/filter-config-v2-spring.xml)

```
    . . .
            <bean id="commerceWebServicesFilterChainV2"                class="de.hybris.platform.servicelayer.web.PlatformFilterCha
            <constructor-arg>
                <ref bean="commerceWebServicesFilterChainListV2" </                />
            constructor-arg>
        </боб>
        <alias name="defaultCommerceWebServicesFilterChainListV2" alias="commerceWebServicesFilterChainListV2" /> <util:list
        id="defaultCommerceWebServicesFilterChainListV2">
            <!-- фильтр, который перехватывает и разрешает исключения, выдаваемые другими фильтрами -->
            <ref bean="exceptionTranslationFilter"                    />
            <!-- универсальные фильтры платформы -->
            <ref    bean="log4jFilter" />
            <реф  bean="restSessionFilterV2" />
            <!-- фильтры commerceWebservices --> <ref
                bean="baseSiteMatchingFilter" />
            <реф  bean="commerceWebServicesSessionAttributesFilterV2"                    />
            <!--    Безопасность -->
            <реф  bean="springSecurityFilterChain"                />
            <реф  bean="userMatchingFilter"            />
            <!-- Соответствующие фильтры -->
            <ref    bean="cartMatchingFilter"            />
            <реф  bean="guestRoleFilterV2"            />
        </util:list>
    . . .
```

Сопутствующая информация

Управление пользователями и группами пользователей

Управление и проверка прав доступа

OAuth 2.0

Реализация API OCC

Получите общее представление о том, как реализован API SAP Commerce OCC, и узнайте, что вам нужно знать, чтобы расширить его с помощью собственной реализации API.

Реализация RESTful

Реализация RESTful в OCC предоставляет пользователю подход к URL-адресам и контролю доступа.

Кэширование

Обзор кэширования в OCC, а также рекомендации по его использованию.

Ссылка на звонки

Образцы вызовов OCC и сценарии покупок клиентов дают вам набор примеров возможного использования API OCC. Сохранить корзину в OCC

Функция сохранения корзины позволяет сохранять и восстанавливать сохраненные корзины позднее.

Картирование DTO и конфигурация ответа

Механизм сопоставления данных упрощает сопоставление данных между исходными и целевыми объектами.

Ответы об ошибках OCC

Рекомендации по использованию ответов об ошибках OCC.

Преобразователи HTTP-сообщений

API OCC позволяет преобразовывать объекты DTO в их текстовое представление (JSON или XML), используемое в вызовах REST, или обратно. Концепция WsDTO

WsDTO — это уровень данных, используемый REST API в OCC.

Оплата в OCC

Описание потока вызовов процесса оплаты, включая определение платежных реквизитов и авторизацию карты.

Включение и использование очереди статусов заказов

Очередь обновления статуса заказа включена и доступна по умолчанию вusommercевеб-сервисырасширение. Однако оно не привязано ни к одному бизнес-процессу, отвечающему за обработку заказов, поскольку обработка заказов — сложный механизм, имеющий множество настраиваемых точек соприкосновения. Поэтому клиент должен указать их все и охватить все возможные изменения статуса заказа.

Включение резервного языка с помощью OCC

Обеспечьте правильный языковой резерв в случаях, когда определенный язык недоступен для запрашиваемых данных.

Реализация RESTful

Реализация RESTful в OCC предоставляет пользователю подход к URL-адресам и контролю доступа.

Без гражданства

OCC не использует сеансы. Это означает, что куки JSESSIONID можно (и нужно) игнорировать. Чтобы получить доступ к ресурсам от конкретного пользователя, вы можете следовать соглашению URL, описанному ниже (или реализовать свое собственное).

Пользователи

Пользовательские ресурсы доступны по следующему пути:

```
https://localhost:9002/rest/v2/{baseSiteID}/users/{userID}/...
```

Допустимые значения userID

- анонимный

  Анонимный пользователь.

- ${ИдентификаторКлиента}

  Идентификатор клиента зарегистрированного пользователя.

Пример:

```
ПОЛУЧИТЬ https://localhost:9002/rest/v2/wsTest/users/2036bc69-d1ee-4cf4-9205-210c2f936970/addresses
```

Вам необходимо иметь соответствующие права, чтобы видеть ресурсы указанного пользователя.

Тележки

Ресурс корзины доступен по следующему пути:

https://localhost:9002/rest/v2/{baseSiteID}/users/{userID}/carts/{cartID}

Допустимые значения cartID

- текущий

  Представляет последнюю измененную корзину указанного пользователя.

- ${guid}

  GUID анонимной корзины. Работает только с анонимным пользователем.

- ${код}

  Код неанонимной корзины. Работает только для зарегистрированных пользователей.

  Все вызовы, относящиеся к конкретной корзине, имеют одинаковую структуру, которая также содержит владельца корзины:

  ОТПРАВИТЬ https://localhost:9002/rest/v2/wsTest/users/2036bc69-d1ee-4cf4-9205-210c2f936970/carts/0000012/entrie

  Таким образом, вы сможете получить доступ только к корзинам, принадлежащим указанному пользователю, и для этого вам необходимо иметь соответствующие права.

Заказы

Доступ к заказам может быть предоставлен отдельному пользователю или как к глобальному ресурсу для всех пользователей, но для просмотра ресурсов вам необходимо иметь соответствующие права.

Ресурс заказов пользователей доступен по следующему пути:

https://localhost:9002/rest/v2/{baseSiteID}/users/{userID}/orders/{orderID}

Допустимые значения orderID

- ${код}

  Код заказа. Работает только для зарегистрированных пользователей.

  Все вызовы, относящиеся к конкретному заказу, имеют одинаковую структуру, которая также содержит владельца заказа:

  ОТПРАВИТЬ https://localhost:9002/rest/v2/wsTest/users/2036bc69-d1ee-4cf4-9205-210c2f936970/orders/testOrder1

  Таким образом, вы сможете получить доступ только к заказам, принадлежащим указанному пользователю. Для этого у вас должны быть соответствующие права.

  Глобальный ресурс заказов доступен по следующему пути:

  https://localhost:9002/rest/v2/{baseSiteID}/orders/{orderID}

Допустимые значения кода

- ${код}

  Код заказа.

- ${guid}

  Глобальный идентификатор Ордена.

Все вызовы, относящиеся к конкретному заказу, имеют одинаковую структуру, которая также содержит владельца заказа:

ОТПРАВИТЬ https://localhost:9002/rest/v2/wsTest/orders/1beb1e9f5043ef28aa5f821ada8aeee5a7a40ac4

Таким образом, вы сможете получить доступ ко всем заказам пользователя, но для этого у вас должны быть соответствующие права.

Контроль доступа

Доступ зависит от ролей, предоставленных OAuth2.

Роль	Описание	Права
РОЛЬ_КЛИЕНТА	Клиентское приложение (т.е. мобильное приложение)	Возможен доступ только к анонимным пользовательским ресурсам.
ROLE_CUSTOMERGROUP	Пользователь аутентифицирован клиентским приложением	Может получить доступ только к своим собственным ресурсам
ROLE_TRUSTEDCLIENT	Доверенное клиентское приложение (например, Adobe CQ5)	Может получить доступ ко всем пользователям и их ресурсам
ROLE_CUSTOMERMANAGERGROUP	Пользователь-менеджер аутентифицируется клиентским приложением (например, POS-терминалом)	Может получить доступ ко всем пользователям и их ресурсам

Глаголы в RESTful API

В версии OCC API v2 некоторые HTTP-методы были переработаны для соответствия более современному стандарту RESTful.

ПОСТ: Создать ресурс

POST используется для создания подчиненного ресурса, который не существует. В результате возвращается созданная сущность. Примеры: создание заказа или адреса.

PUT: обновить весь ресурс

PUT используется для обновления полной сущности путем отправки всей сущности на URL, который указывает на определенный ресурс. Все отсутствующие поля будут установлены в NULL или значение по умолчанию. Примеры: обновление пользователя или адреса.

PATCH: Частичное обновление ресурса

PATCH используется для частичного обновления. Например, когда вам нужно обновить только одно поле ресурса, используется PATCH. PUTting или POSTing полного представления ресурса создает дополнительные накладные расходы и использует больше пропускной способности. Примеры: обновление фамилии пользователя или обновление названия улицы в адресе.

УДАЛИТЬ: Удалить ресурс

DELETE используется для удаления ресурса. Примеры: удаление пользователя или адреса.

Кэширование

Обзор кэширования в OCC, а также рекомендации по его использованию.

Обзор

Кэш располагается между одним или несколькими веб-серверами (также известными как исходные серверы) и клиентом или многими клиентами и отслеживает поступающие запросы, сохраняя копии ответов, таких как HTML-страницы, изображения и файлы. Затем, если поступает еще один запрос на тот же URL, он может использовать один из собранных ответов, вместо того чтобы снова запрашивать его у исходного сервера. В этом документе описывается, как использовать кэширование в OCC. Он содержит подробную информацию как о кэшировании на стороне клиента, так и о кэшировании на стороне сервера.

Кэширование на стороне клиента

Theвеб-сервисыcommonspасширение определяет @CacheControlаннотация, которая может быть использована для генерации заголовка Cache-Control в ответе. Если вы хотите включить кэширование на стороне клиента для определенного метода или всего контроллера, просто аннотируйте его с помощью @CacheControl и указать соответствующие директивы. Пример использования аннотации можно увидеть вПродуктыController.javacорт:

```
. . .
    @RequestMapping(значение = "/{productCode}", метод = RequestMethod.GET)
    @CacheControl(директива = CacheControlDirective.PRIVATE, maxAge = 120)
```

```
    @ResponseBody
    public ProductWsDTO getProductByCode(@PathVariable Final String ProductCode,
            @RequestParam(defaultValue = DEFAULT_FIELD_SET) конечные строковые поля)
    {
      . . .
    }
. . .
```

-Примечание

Поскольку аннотация Cache-Control применяется только к методам GET и HEAD, она не повлияет на другие методы запроса.

-Примечание

@CacheControlаннотация работает только еслиCacheControlHandlerInterceptorдобавлен в перехватчики mvc.

```
. . .
    <mvc:перехватчики>
        <bean class="de.hybris.platform.webservicescommons.interceptors.CacheControlHandlerInterceptor"/> </mvc:interceptors>

. . .
```

Кэширование на стороне сервера

Конфигурация кэша Spring

Конфигурацию кэша Spring можно найти в следующем файле:ycommercewebservices/web/webroot/WEB-INF/config/cache-config-spring.xml.

Необходимо включить функцию кэширования. Это можно сделать с помощью <кэш:управляемый аннотациями>элемент. Этот элемент также позволяет определить генератор ключей по умолчанию и менеджер кэша, который будет использоваться для кэширования.

Конфигурация ниже используетcommerceCacheKeyGeneratorописано в разделе ниже: <Генератор ключей кэша>.Менеджер кэша определен с использованием SpringCompositeCacheManagerсделать механизм кэширования доступным также для дополнений. Единый менеджер кэширования для OCC используетTenantAwareEhCacheManagerFactoryBeanкласс, который определен ввеб-сервисыcommonsрасширение.

```
<бобы    xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance xmlns:cache="http://
        www.springframework.org/schema/cache"
        xmlns:context="http://www.springframework.org/schema/context"
        xsi:schemaLocation="http://www.springframework.org/schema/beans http://
          www.springframework.org/schema/beans/spring-beans.xsd http://
          www.springframework.org/schema/cache
            http://www.springframework.org/schema/cache/spring-cache.xsd">

    <cache:annotation-driven cache-manager="compositeWsCacheManager" key-generator="commerceCacheKeyGenerator"/

    <псевдоним имя="defaultWsCacheManagerList" псевдоним="wsCacheManagerList"/>
    <utils:list id="defaultWsCacheManagerList">
        <ref bean="defaultWSCacheManager"/> </
        utils:list>

    <!-- Композитный менеджер кэша используется для того, чтобы позволить дополнениям добавлять свои собственные менеджеры кэша путем изменения wsCacheMa

    <псевдоним    имя="defaultCompositeWSCacheManager" псевдоним="compositeWsCacheManager"/>
    <боб    идентификатор="defaultCompositeWSCacheManager" класс="org.springframework.cache.support.CompositeCacheManager">
        <свойство    имя="менеджеры кэша">
            <реф  bean="wsCacheManagerList"/>
```

```
        </свойство>
    </боб>

    <!-- Менеджер кэша по умолчанию для OCC: -->

    <псевдоним имя="defaultWSCacheManager" alias="wsCacheManager"/>
    <боб идентификатор="defaultWSCacheManager" class="org.springframework.cache.ehcache.EhCacheCacheManager">
        <свойство имя="менеджер кэша" ref="wsEhcache"/>
    </боб>

    <псевдоним имя="defaultWSEhcache" псевдоним="wsEhcache"/>
    <боб id="defaultWSEhcache" class="de.hybris.platform.commercewebservicescommons.cache.TenantAwareEhCacheMa
        <свойство имя="configLocation" значение="/WEB-INF/cache/ehcache.xml"/>
    </боб>
</бобов>
```

Конфигурация Ehcache

Ehcache — это кэш с открытым исходным кодом, основанный на стандартах, который используется для повышения производительности, разгрузки базы данных и упрощения масштабируемости. Подробности

Конфигурацию кэша можно найти в следующем файле:ycommercewebservices/web/webroot/WEB-INF/ehcache.xml.

```
<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="../config/ehcache.xsd" updateCheck="false" мониторинг="автоопределение"
        dynamicConfig="true">

    <!--
    см. ehcache-core-*.jar/ehcache-failsafe.xml для описания элементов
     - - >

    <diskStore path="java.io.tmpdir/occ_cache"/> <defaultCache

            maxElementsInMemory="100000"
            вечный="ложь"
            timeToIdleSeconds="360"
            timeToLiveSeconds="360"
            overflowToDisk="true"
            diskPersistent="ложь"
            maxEntriesLocalDisk="10"
            diskExpiryThreadIntervalSeconds="360"
            memoryStoreEvictionPolicy="FIFO"
            />

    <кэш имя="fieldSetCache"
        maxElementsInMemory="1000"
        вечный="истина"
        переполнениеНаДиске="истина"
        diskPersistent="ложь"
        maxEntriesLocalDisk="2000"
        memoryStoreEvictionPolicy="LRU"/>

    <кэш имя="productSearchCache"
        maxElementsInMemory="1000"
        вечный="false"
        переполнениеНаДиске="истина"
        timeToLiveSeconds="150"
        diskPersistent="false"
        maxEntriesLocalDisk="2000"
        memoryStoreEvictionPolicy="LRU"/>

. . .
. . .
```

```
. . .
</ehcache>
```

Кэшированные методы

Методы, которые необходимо кэшировать, должны быть аннотированы символом @.Кэшируемыйаннотация. В простейшем формате аннотация требует только имя кэша, связанного с этим методом. Например, @Кэшируемый (defaultCache).В таком случае ключ для кэша генерируется с помощью генератора ключей, определенного в <кэш:annotation-drive >тег. Дляусommerceсевеб-сервисырасширение это commerceCacheKeyGenerator.Если вы хотите использовать другой генератор ключей или определить ключ с помощью SpEL, то используйте атрибут ключа аннотации. Например, @Кэшируемый (значение = defaultCache, ключ =*<"{#параметр1,#параметр2,#параметр3}>*).

Генератор ключей кэша

Генератор ключей по умолчанию, настроенный дляусommerceсевеб-сервисырасширение естьcommerceCacheKeyGenerator.Это определено в коммерциявебсервисыcommons-spring.xmlле.

```
. . .
        <bean id="commerceCacheKeyGenerator"            class="de.hybris.platform.commercewebservicescommons.cache.CommerceCa
            <имя_свойства="baseSiteService" </bean>    ref="baseSiteService"/>

. . .
```

Этот генератор ключей создает ключ на основе параметров метода и некоторых дополнительных атрибутов, таких как:*<базовый сайт,> <язык>,<пользователь>*, *<валюта>*. Базовый идентификатор сайта и коды ISO языка всегда добавляются к сгенерированному ключу. Идентификатор пользователя и код ISO валюты не добавляются по умолчанию, но их можно легко добавить, как показано в следующем примере:

```
@Cacheable(value = "productCache", key =
"T(de.hybris.platform.commercewebservicescommons.cache.CommerceCacheKeyGenerator).generateKey(true,true,#produc
```

Первый параметрсгенерироватьКлючМетод определяет, следует ли добавлять идентификатор пользователя к ключу кэша. Второй параметр определяет, следует ли добавлять изокод валюты к ключу кэша.

Ссылка на звонки

Образцы вызовов OCC и сценарии покупок клиентов дают вам набор примеров возможного использования API OCC.

Прежде чем работать с примерами вызовов, ознакомьтесь со следующими ключевыми функциями API OCC:

Без гражданства

OCC не использует сеансы. Чтобы получить доступ к ресурсам для конкретного пользователя, вы можете использовать следующие соглашения URL:

- Ресурсы пользователя:https://localhost:9002/rest/v2/{baseSiteID}/users/{userID}/...
- Ресурсы корзины:https://localhost:9002/rest/v2/{baseSiteID}/users/{userID}/carts/{cartID}...
- Заказать ресурсы:https://localhost:9002/rest/v2/{baseSiteID}/users/{userID}/orders/{orderID}...

где {ID пользователя}может иметь следующие значения:

- ${customerId} -Уникальный идентификатор клиента зарегистрированного пользователя.
- анонимный -Анонимный пользователь.
- текущий -Пользователь представлен токеном OAuth.

где {cartID}может иметь следующие значения:

- ${guid} -Глобальный уникальный идентификатор (GUID) анонимной корзины. Работает только для анонимного пользователя.
- ${код} -Код неанонимной корзины. Работает только для зарегистрированных пользователей.
- текущий -Представляет последнюю измененную корзину указанного пользователя.

Параметры запроса локализации

Каждый из вызовов может содержать дополнительные параметры URL, которые изменяют локализацию возвращаемых объектов. Общие параметры:

- язык:Изменяет язык локализованных значений в ответе. Укажите код языка ISO в качестве значения. Если параметр lang не указан, то ответ локализуется на языке по умолчанию вашего базового хранилища.
- валюта:Изменяет валюту вашего вызова веб-сервиса. Это означает, что все вычисления выполняются в запрошенной валюте, и все значения цен представлены с использованием запрошенной валюты. Укажите код валюты ISO в качестве значения. Если нет*<curr>*Если указан параметр, то используется валюта по умолчанию вашего базового магазина.

Вы можете использовать эти параметры с каждым запрошенным ресурсом. Обработка параметров изолирована в специализированном фильтре HTTP-запросов. Проверьте следующие примеры:

- https://localhost:9002/rest/v2/mysite/users/{userId}/carts?curr=USD&lang=en:Используйте английский язык и американский Доллар за запрос.
- https://localhost:9002/rest/v2/mysite/products/{productCode}?curr=USD:Использовать доллар США и язык по умолчанию магазин mysite.
- https://localhost:9002/rest/v2/mysite/stores/{storeId}?lang=de:Использовать немецкий язык и валюту по умолчанию магазин mysite.

Параметр поля

Для большинства запросов ресурсов существуетполяпараметр, который может быть использован для настройки ответа. Он позволяет вам выбирать поля для каждого объекта, возвращаемого в ответе. Он может состоять из имен полей и уровней, разделенных запятой, например: BASIC_FIELD_LEVEL, eld1,eld2. Он может также иметь вложенную конфигурацию для выбранных полей, таких как:поле1(поле11,поле12).

Параметры пересчета корзины в CartMatchingFilter

V2 позволяет изменять параметры пересчета корзины вФильтр соответствия корзине.

В версии 2 корзина покупателя сохраняется в базе данных и загружаетсяФильтр соответствия корзиныдля каждого входящего запроса ресурса корзины. При загрузке корзины фильтр проверяет, истек ли срок действия корзины. Если это так, корзина перестраивается, но пересчет не выполняется. Новая перестроенная корзина содержит все записи, перезаписанные из просроченной корзины. Если вы хотите указать срок действия корзины, отсчитываемый с момента ее последнего изменения, добавьте следующее свойство к вашемуместные.свойствале:

```
commerceservices.cartValidityPeriod=<секунд>
```

Если срок действия корзины не истек, загруженная корзина пересчитывается только в том случае, если ее валюта отличается от уже установленной валютным фильтром. Вы можете настроить поведение пересчета корзины, изменив значенияобновитьКорзинупараметр запроса и ycommercewebservices.cart.refreshed.by.defaultсобственность.обновитьКорзинупараметр необязателен и может использоваться только с Запросы ресурсов корзины.

Если запрос содержитобновитьКорзинупараметр систинныйценность,Фильтр соответствия корзинызаставляет пересчитать корзину. Смотрите следующий пример запроса «получение способов доставки корзины» собновитьКорзинупараметр, инициирующий пересчет корзины:

```
https://localhost:9002/rest/v2/{baseSiteID}/users/{userID}/carts/{cartID}/deliverymodes?refreshCart=true
```

ЕслиобновитьКорзинупараметр не задан, поведение пересчета корзины зависит от ycommercewebservices.cart.refreshed.by.defaultсвойство. По умолчанию значение свойства установлено наЛОЖЬ:

```
ycommercewebservices.cart.refreshed.by.default=false
```

Значение свойства по умолчанию гарантирует, что корзина будет пересчитана только в том случае, если ее валюта отличается от той, которая уже установлена фильтром валюты. Если вы хотите пересчитать корзину для каждого запроса ресурса корзины, добавьте свойство систинныйценность для вашегоместные.свойствале.

-Примечание

Изменение значения по умолчаниюyоcommercewebservices.cart.refreshed.by.defaultсвойство пересчитывать корзины для каждого запроса ресурсов корзины может замедлить работу вашей системы.

Чтобы явно отключить пересчет корзины, используйтеобновитьКорзинупараметр со значениемЛОЖЬ.Ниже приведен пример запроса «получение способов доставки корзины» с отключенным пересчетом корзины для каждого запроса:

https://localhost:9002/rest/v2/{baseSiteID}/users/{userID}/carts/{cartID}/deliverymodes?refreshCart=false

DTO в теле запроса

Версия 2 включает несколько новых запросов, которые принимают DTO в теле запроса. В таких запросах параметры могут передаваться в формате JSON или XML. Пример этого можно увидеть в столбце Body Parameters - Content-Type: application/json.

Отдельные вызовы и сценарии

Воспользуйтесь следующими ссылками, чтобы узнать больше о выполнении полного сценария покупки с использованием вызовов OCC.

- Сценарии процесса покупки клиента
- Отдельные звонки для сценариев покупки клиентами

Сценарии процесса покупки клиента

Пошаговое руководство по последовательностям вызовов, которые следует использовать для выполнения заданного сценария (например, регистрация клиента).

Обзор

Приведенные ниже сценарии представляют собой полное пошаговое руководство по использованию заданной последовательности вызовов.

-Примечание

Примеры URL используются только для презентационных целей. В настроенной среде необходимо заменить адрес сервера на свой собственный.

Зарегистрированный клиент

Предпосылки

Клиент уже зашел в магазин и указал всю информацию, необходимую для оформления заказа (например, адрес и платежные реквизиты).

Шаги сценария

- Поиск продукта
- Получить подробную информацию о продукте
- Добавить товар в корзину (например,basket)
- Получить токен доступа для клиента
- Выполнить оформление заказа
- Отображение размещенного заказа для клиента
- Выйти

Запрос потока

-Примечание

Вызовы должны быть выполнены в порядке, определенном в таблице request low. Прокрутите вправо, чтобы увидеть полное содержимое таблицы.

Метод	URL
получать	https://localhost:9002/rest/v2/electronics/products/search
получать	https://localhost:9002/rest/v2/electronics/products/489702
почта	https://localhost:9002/rest/v2/electronics/users/anonymous/carts
почта	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4efa0
почта	https://localhost:9002/authorizationserver/oauth/token
почта	https://localhost:9002/rest/v2/electronics/users/current/carts
получать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001
получать	https://localhost:9002/rest/v2/electronics/users/current/addresses
помещать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/addresses/delivery
получать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymodes
помещать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymode
получать	https://localhost:9002/rest/v2/electronics/users/current/paymentdetails
помещать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/paymentdetails
почта	https://localhost:9002/rest/v2/electronics/users/current/orders
получать	https://localhost:9002/rest/v2/electronics/users/current/orders
получать	https://localhost:9002/rest/v2/electronics/users/current/orders/00012002

Новый клиент

Предпосылки

Клиент не зарегистрирован в магазине.

Шаги сценария

- Зарегистрируйте нового клиента и получите токен доступа
- Создайте новый адрес для клиента

- Поиск определенного продукта
- Получить подробную информацию о продукте
- Создайте корзину для клиента
- Добавьте товары в корзину
- Оформление заказа: установка адреса доставки, установка способа доставки, определение данных кредитной карты
- Оформить заказ
- Отображение размещенного заказа для клиента

Запрос потока

-Примечание

Вызовы должны быть выполнены в порядке, определенном в таблице request low. Прокрутите вправо, чтобы увидеть полное содержимое таблицы.

Метод	URL	Запрос Параметры	Б С
ПОЧТА	https://localhost:9002/authorizationserver/oauth/token		с
ПОЧТА	https://localhost:9002/rest/v2/electronics/users		вот &
ПОЧТА	https://localhost:9002/authorizationserver/oauth/token		с &
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/addresses	поля=id	
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/products/search	запрос=штатив	
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/products/3429337		

Метод	URL	Запрос Параметры	Б С
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts		
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/entries		с
ПОМЕЩАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/addresses/delivery		а
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymodes		
ПОМЕЩАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymode		г
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/paymentdetails		а 1
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/orders		с

Метод	URL	Запрос Параметры	Б С
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/orders		
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/orders/00012002		

Забрать в магазине

Предпосылки

Клиент уже зарегистрирован в магазине и указал всю информацию, необходимую для оформления заказа (например, платежную информацию).

Шаги сценария

- Получите токен доступа для клиента
- Создать корзину для клиента
- Добавьте товары, которые можно будет забрать в магазине
- Оформите заказ и выберите способ доставки «самовывоз».
- Покажите клиенту размещенный заказ

Запрос потока

-Примечание

Вызовы должны быть выполнены в порядке, определенном в таблице request low. Прокрутите вправо, чтобы увидеть полное содержимое таблицы.

Метод	URL	Параметры запроса
ПОЧТА	https://localhost:9002/authorizationserver/oauth/token	
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts	
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/entries	поля=statusCode,qua
ПЛАСТЫРЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/entries/0	

Метод	URL	Параметры запроса
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/entries	
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001	поля=код,pickupItem pickupOrderGroups(ru)
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymodes	поля=ОСНОВНЫЕ
ПОМЕЩАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymode	
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/paymentdetails	сохранено=истина
ПОМЕЩАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/paymentdetails	
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/orders	
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/orders	
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/orders/00012002	

Самовывоз из магазина с функцией консолидации магазинов

Предпосылки

Клиент уже зарегистрирован в магазине и указал всю информацию, необходимую для оформления заказа (например, адрес и платежную информацию).

Шаги сценария

-Примечание

Theacceleratorwebservicesaddon Дополнение необходимо установить.

- Получить токен доступа для клиента

- Создать корзину для клиента
- Добавьте товар, который можно будет забрать в магазине
- Добавьте товар, который можно будет забрать в другом магазине
- Используйте функцию консолидации магазина
- Выполнить оформление заказа

-Примечание

Вызовы должны быть выполнены в порядке, определенном в таблице request low. Прокрутите вправо, чтобы увидеть полное содержимое таблицы.

Метод	URL	Параметры запроса
ПОЧТА	https://localhost:9002/authorizationserver/oauth/token	
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts	
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/entries	
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/entries	
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001	поля=код,самовывоз Группы самовывозаЗаказов
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/consolidate	
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/consolidate	
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymodes	

Метод	URL	Параметры запроса
ПОМЕЩАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymode	
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/paymentdetails	сохранено=истина
ПОМЕЩАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/paymentdetails	
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/orders	

Совмещение способа доставки с самовывозом из магазина

Предпосылки

Клиент уже зарегистрирован в магазине и указал всю информацию, необходимую для оформления заказа (например, адрес и платежную информацию).

Шаги сценария

- Получите токен доступа для клиента
- Создать корзину для клиента
- Добавьте продукт, который будет доставлен
- Добавьте товар, который можно будет забрать в магазине
- Выполнить оформление заказа
- Отображение размещенного заказа для клиента

-Примечание

Вызовы должны быть выполнены в порядке, определенном в таблице request low. Прокрутите вправо, чтобы увидеть полное содержимое таблицы.

Запрос потока

Метод	URL	Параметр запроса
ПОЧТА	https://localhost:9002/authorizationserver/oauth/token	
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts	
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/entries	

Метод	URL	Параметр запроса
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/entries	
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001	поля=код,выбор самовывозOrderGrou доставкаItemsQua
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/addresses	
ПОМЕЩАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/addresses/delivery	
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymodes	
ПОМЕЩАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymode	
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/paymentdetails	сохранено=истина
ПОМЕЩАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/paymentdetails	
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/orders	
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/orders	
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/orders/00012002	

Процесс оформления заказа

Оформление заказа с созданием нового адреса и платежной информации для клиента

Предпосылки

Клиент зарегистрирован и имеет корзину с кодом 00012001.

Шаги сценария

- Создайте адрес и установите его как адрес доставки
- Установить режим доставки
- Создайте платежную информацию
- Авторизуйте корзину
- Оформить заказ

Запрос потока

-Примечание

Вызовы должны быть выполнены в порядке, определенном в таблице request low. Прокрутите вправо, чтобы увидеть полное содержимое таблицы.

Метод	URL	Параметры тела
		Тип контента:**ап**
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/addresses	rstName=Джон&
ПОМЕЩАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/addresses/delivery	адресId=87961
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymodes	
ПОМЕЩАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymode	deliveryModeId=p
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/paymentdetails	accountHolderNa 16-19&адрес счета

Метод	URL	Параметры тела Тип контента:**ап**
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/orders	cartId=00012001

Оформление заказа с указанием текущего адреса и платежной информации для клиента

Предпосылки

Клиент зарегистрирован и имеет корзину с кодом 00012001.

Шаги сценария

- Попробуйте получить способы доставки
- Укажите адрес доставки
- Установите режим доставки
- Укажите платежную информацию
- Авторизуйте корзину
- Оформить заказ

Запрос потока

-Примечание

Вызовы должны быть выполнены в порядке, определенном в таблице request low. Прокрутите вправо, чтобы увидеть полное содержимое таблицы.

Метод	URL	Параметры запроса
получать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymodes	
получать	https://localhost:9002/rest/v2/electronics/users/current/addresses	
помещать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/addresses/delivery	

Метод	URL	Параметры запроса
получать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymodes	поля=ОСНОВНЫЕ
помещать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymode	
получать	https://localhost:9002/rest/v2/electronics/users/current/paymentdetails	сохранено=истина&
помещать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/paymentdetails	
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/orders	

Процесс оформления заказа с использованием СОП

Предпосылки

Клиент зарегистрирован и имеет корзину с кодом 00012001.

Шаги сценария

- Укажите адрес доставки
- Установите режим доставки
- Получите информацию, необходимую для создания подписки на платеж
- Создайте подписку, связавшись напрямую с поставщиком платежных услуг
- Обработайте ответ от поставщика платежных услуг и создайте платежные реквизиты.
- Авторизуйте карту и оформите заказ

Запрос потока

-Примечание

Вызовы должны быть выполнены в порядке, определенном в таблице request low. Прокрутите вправо, чтобы увидеть полное содержимое таблицы.

Метод	URL	Запрос Параметры
получать	https://localhost:9002/rest/v2/electronics/users/current/addresses	
получать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymodes	поля=ОСНОВНЫЕ

Метод	URL	Запрос Параметры
получать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/payment/sop/request	
ПОЧТА	*postUrl*получил в предыдущем запросе	
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/payment/sop/response	
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/orders	
помещать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/addresses/delivery	
помещать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymode	

Процесс оформления заказа с использованием стандартной операционной процедуры и расширенного обратного вызова продавца

Предпосылки

Клиент зарегистрирован и имеет корзину с кодом 00012001.

Шаги сценария

- Укажите адрес доставки
- Установите режим доставки
- Получите информацию, необходимую для создания подписки на оплату
- Создайте подписку, связавшись напрямую с поставщиком платежных услуг
- Спросите OCC об ответе платежного провайдера — отрицательный ответ
- Удалить информацию об ответе на платеж
- Создайте подписку, связавшись напрямую с поставщиком платежных услуг
- Спросите OCC об ответе платежного провайдера — положительный ответ
- Авторизуйте карту и оформите заказ

Запрос потока

-Примечание

Вызовы должны быть выполнены в порядке, определенном в таблице request low. Прокрутите вправо, чтобы увидеть полное содержимое таблицы.

Метод	URL	Запрос Параметр
получать	https://localhost:9002/rest/v2/electronics/users/current/addresses	
помещать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/addresses/delivery	
получать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymodes	поля=BAS
помещать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/deliverymode	
получать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/payment/sop/request	
ПОЧТА	*URL-адрес поставщика платежных услуг*(postUrl получен в предыдущем запросе)	
получать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/payment/sop/response	
	-Примечание Предположение: ответ на предыдущий звонок был отрицательным (например, клиент указал неправильный номер карты)	
УДАЛИТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/payment/sop/response	
ПОЧТА	*URL-адрес поставщика платежных услуг*	
получать	https://localhost:9002/rest/v2/electronics/users/current/carts/00012001/payment/sop/response	

Метод	URL	Запрос Параметр
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/orders	

Сценарий оформления заказа гостем с созданием полной учетной записи

Предпосылки

Никто.

Шаги сценария

- Поиск продукта
- Получить подробную информацию о продукте
- Создать пустую корзину
- Добавить товар в корзину
- Получить токен доступа клиента
- Представьтесь как гость
- Проверить
- Отображение размещенного заказа
- Преобразовать гостевую учетную запись в полноценную учетную запись клиента (по желанию)
- Войти как клиент
- Получить список заказов

Запрос потока

-Примечание

Вызовы должны быть выполнены в порядке, определенном в таблице request low. Прокрутите вправо, чтобы увидеть полное содержимое таблицы.

Метод	URL
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/products/search
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/products/489702
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts

Метод	URL
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4e
ПОЧТА	https://localhost:9002/authorizationserver/oauth/token
ПОМЕЩАТЬ	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4e
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4e
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4e
ПОМЕЩАТЬ	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4e
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4e

Метод	URL
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/orders
ПОЧТА	https://localhost:9002/rest/v2/electronics/users
ПОЧТА	https://localhost:9002/authorizationserver/oauth/token
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/orders
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/orders/00002009

Сценарий оформления заказа гостем с проверкой статуса заказа

Предпосылки

Никто.

Шаги сценария

- Поиск продукта
- Получить подробную информацию о продукте
- Создать пустую корзину
- Добавить товар в корзину
- Получить токен доступа клиента

- Представьтесь как гость
  - Выполнить оформление заказа
  - Получить информацию о заказе

Запрос потока

-Примечание

Вызовы должны быть выполнены в порядке, определенном в таблице request low. Прокрутите вправо, чтобы увидеть полное содержимое таблицы.

Метод	URL
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/products
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/products /489702
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4efa0
ПОЧТА	https://localhost:9002/authorizationserver/oauth/token
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4efa0
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4efa0

Метод	URL
получать	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4efa0
помещать	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4efa0
почта	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4efa0
почта	https://localhost:9002/rest/v2/electronics/users/anonymous/orders
получать	https://localhost:9002/rest/v2/electronics/users/anonymous/orders/51ecf334a103df146a85b486af01aad57df4efa

Использование ваучера

Предпосылки

В системе определен ваучер с кодом HY2008.

Шаги сценария

- Создать пустую корзину
- Добавить товары в корзину
- Применить ваучер
- Получить корзину
- Зарегистрируйте нового клиента и получите для него токен доступа
- Назначить корзину клиенту
- Создайте новый адрес для клиента
- Оформление заказа: установка адреса доставки, установка способа доставки, установка данных кредитной карты
- Разместить заказ
- Отображение размещенных заказов для клиента

Запрос потока

-Примечание

Вызовы должны быть выполнены в порядке, определенном в таблице request low. Прокрутите вправо, чтобы увидеть полное содержимое таблицы.

Метод	URL
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4efa0
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4efa0
ПОЛУЧАТЬ	https://localhost:9002/authorizationserver/oauth/token
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4efa0
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01aad57df4efa0

Метод	URL
ПОЧТА	https://localhost:9002/rest/v2/electronics/users
ПОЧТА	https://localhost:9002/authorizationserver/oauth/token
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/addresses
ПОМЕЩАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00002036/addresses/delivery
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00002036/deliverymodes
ПОМЕЩАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00002036/deliverymode
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts/00002036/paymentdetails

Метод	URL
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/orders
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/orders
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/orders/00002037

Объединение корзин

Предпосылки

Клиент уже зарегистрирован в магазине и имеет корзину с одним товаром (код товара = 1382080). Код корзины = 00001002 и guid корзины = 51ecf334a103df146a85b486af01aad57df4efa0.

Шаги сценария

- Поиск продуктов
- Получить подробную информацию о продукте
- Создать пустую корзину
- Добавьте товар в корзину
- Получить токен для клиента
- Получить клиентские корзины
- Объединить последнюю измененную корзину клиента с анонимной корзиной
- Получить корзину

Запрос потока

-Примечание

Вызовы должны быть выполнены в порядке, определенном в таблице request low. Прокрутите вправо, чтобы увидеть полное содержимое таблицы.

Метод	URL
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/products/search
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/products/489702
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/02017b6dad8834ba6f3fd26eb9ab4f270e9b1858
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/02017b6dad8834ba6f3fd26eb9ab4f270e9b1858
ПОЧТА	https://localhost:9002/authorizationserver/oauth/token
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/carts
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/carts/00001002

Метод	URL

Сопутствующая информация

Реализация RESTful в v2

Конфигурация ответа в v2

Отдельные звонки для сценариев покупки клиентами

Отдельные звонки для сценариев покупки клиентами

Примеры отдельных шагов сценариев процесса покупки для Omni Commerce Connect (OCC) в v2.

Одиночные звонки

В таблицах ниже представлены наборы доступных вызовов, которые могут быть выполнены с использованием API v2.

-Примечание

Звонки были сгруппированы в темы, отражающие их цель, однако их следует рассматривать как отдельные примеры, а не как полные сценарии. Для полных бизнес-сценариев см.Сценарии процесса покупки клиента .

-Примечание

Примеры URL используются только для презентационных целей. В настроенной среде необходимо заменить адрес сервера на свой собственный.

Авторизация

Основная тема	Метод	URL	Запрос Параметры	Параметры тела
				Тип контента:**приложение/x-www**
Новый Клиент Регистрация	ПОЧТА	https://localhost:9002/authorizationserver/oauth/token		client_id=$CLIENT_ID$&client_secret
	ПОЧТА	https://localhost:9002/rest/v2/electronics/users		логин=john.doe@mail.com &пароль= &rstName=Джон&lastName=До&тит
Получать Клиент Токен	ПОЧТА	https://localhost:9002/authorizationserver/oauth/token		client_id=$CLIENT_ID$&client_secret &grant_type=пароль&имя пользователя=jo

Управление профилем клиентов

Основная тема	Метод	URL	Запрос Р
Авторизация	ПОЧТА	https://localhost:9002/authorizationserver/oauth/token	
Адрес управление	ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/addresses	поля=a
	ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/addresses	
	ПЛАСТЫРЬ	https://localhost:9002/rest/v2/electronics/users/current/addresses/8796158590999	
Оплата Управление	ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/paymentdetails	сохранено=т
	ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/paymentdetails/8796191359018	поля=i срок действияДа
	УДАЛИТЬ	https://localhost:9002/rest/v2/electronics/users/current/paymentdetails/8796191359018	
Клиент Пролетарский Управление	ПЛАСТЫРЬ	https://localhost:9002/rest/v2/electronics/users/current	

Основная тема	Метод	URL	Запрос Р
Пароль Управление	ПОЧТА	https://localhost:9002/occ/v2/electronics/users/current/password	Никто

Управление профилем клиентов (с использованием Customer Manager)

Основная тема	Метод	URL	Запрос Параме
Авторизация	ПОЧТА	https://localhost:9002/authorizationserver/oauth/token	
Адрес Управление	получать	https://localhost:9002/rest/v2/electronics/users/current/addresses	поля=D
	ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/addresses	
	ПЛАСТЫРЬ	https://localhost:9002/rest/v2/electronics/users/current/addresses/8796158590999	
Оплата Управление	получать	https://localhost:9002/rest/v2/electronics/users/current/paymentdetails	сохранено=т
	получать	https://localhost:9002/rest/v2/electronics/users/current/paymentdetails/8796191359018	
	УДАЛИТЬ	https://localhost:9002/rest/v2/electronics/users/current/paymentdetails/8796191359018	
Клиент Пролетарский Управление	ПЛАСТЫРЬ	https://localhost:9002/rest/v2/electronics/users/ john.smith@mail.com	

Основная тема	Метод	URL	Запрос Параме
Пароль Управление	ПОЧТА	https://localhost:9002/occ/v2/electronics/users/current/password	Никто

Поиск продукта

Основной Тема	Метод	URL	Параметры
Продукт Поиск	ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/products/search	запрос=:цена-возрастание:категория:575
	ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/products/search	запрос=:имя-описание:бренд:бренд_10
	ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/products/search	query=камера&pageSize=40&поля=продукты(название
	ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/products/search	запрос=камера:релевантность:категория:575& поля=продукты(ОСНОВНОЙ),пагинация(ПО УМОЛЧАНИЮ)

Основной Тема	Метод	URL	Параметры
	ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/products/search	запрос=камера:релевантность:категория:575:бренд:бренд
	ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/products/search	запрос=камера:релевантность:категория:575:бренд:бренд

Будущая доступность запасов

Основной Тема	Метод	URL	Параметры
Будущее Запас Доступность	ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/futureStocks	Коды продукта=489702
	ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/futureStocks/489702	

Поиск ближайшего магазина, в котором есть определенный товар

Основной Тема	Метод	URL	Параметры
Магазин Поиск	ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/products/search	запрос=EOS

Основной Тема	Метод	URL	Параметры
	получать	https://localhost:9002/rest/v2/electronics/products/1382080	
	получать	https://localhost:9002/rest/v2/electronics/products/1382080/stock	location=Tokio&elds=магазины(имя),страница

Добавление товара в корзину для анонимного пользователя

Основная тема	Метод	URL
Создание Пустой Корзина	ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts
Корзина Управление	ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01
	ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01

Основная тема	Метод	URL
	ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01
	ПЛАСТЫРЬ	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01
	ПЛАСТЫРЬ	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01
	ПОМЕЩАТЬ	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01
	УДАЛИТЬ	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01
	ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01
Корзина Проверка	ПОЧТА	https://localhost:9002/rest/v2/electronics/users/anonymous/carts/51ecf334a103df146a85b486af01

Основная тема	Метод	URL

Процесс оформления заказа

Подробную информацию о доступных звонках можно найти по ссылке:Процесс покупки клиента - Сценарии процесса оформления заказа

Процесс оформления заказа с использованием СОП

Подробную информацию о доступных звонках можно найти по ссылке:Процесс покупки клиента - Сценарии процесса оформления заказа

Управление заказами клиентов

Основная тема	Метод	URL	Параметры запроса
Заказ Управление	ПОЧТА	https://localhost:9002/authorizationserver/oauth/token	
	получать	https://localhost:9002/rest/v2/electronics/users/current/orders	поля=ОСНОВНЫЕ
	получать	https://localhost:9002/rest/v2/electronics/users/current/orders/00001002	поля=statusDisplay,создать

Поиск магазина, отображение сведений о магазине

Основной Тема	Метод	URL	Параметры
Магазин Поиск	получать	https://localhost:9002/rest/v2/electronics/stores	запрос=Tokio&fields=stores(BASIC),пагинация(BASIC)

Основной Тема	Метод	URL	Параметры
Отображать Магазин Подробности	получать	https://localhost:9002/rest/v2/electronics/stores/Накано	поля=ПО УМОЛЧАНИЮ

Объединение корзин

Подробную информацию о доступных звонках можно найти по ссылке:Сценарии процесса покупки клиента — объединение корзин .

REST API управления согласием

API-интерфейсы управления согласием обеспечивают самостоятельное управление согласием для вашей несвязанной витрины, позволяя вам создавать и извлекать предпочтения в отношении согласия клиентов.

API управления согласием позволяют вам:

- Получить список всех согласий пользователя
- Получите конкретное согласие
- Запись предоставления согласия
- Отзыв согласия на запись

-Примечание

Все вызовы документированы в документах OpenAPI в SAP Commerce Cloud:https://HOST:9002/rest/v2/swagger-ui.html

Получение всех согласий

В следующем примере показан вызов REST API для получения всех согласий:

https://{{HOST}}:9002/rest/v2/{{STOREFRONT}}/users/{{USERID}}/consenttemplates

В следующем примере показан ответ, который генерируется в результате этого вызова:


```
{
	"consentTemplates":	[
		{
			"currentConsent":	{
				"код": "000000RT",
				"consentGivenDate":	"2018-06-12T19:00:55+0000",
				"consentWithdrawnDate":	"2018-06-12T19:02:22+0000"
			},
			"description": "Это пример описания согласия, которое необходимо обновить или заменить, базовый "id":
			"MARKETING_NEWSLETTER",
			"name": "Я одобряю этот образец согласия", "version": 0

		}
	]
}
```


Страны REST API для выставления счетов и доставки Страны

API REST стран позволяет получить список стран, определенных для выставления счетов или доставки.

REST API для стран выставления счетов дополняет /страны доставкиREST API. Функциональность REST API Billing Countries дает возможность различать страны, в которые вы осуществляете доставку, и страны, из которых вы можете выставлять счета. Например, если ваш магазин осуществляет доставку только в страны Европейского сообщества, вы все равно можете разрешить выставление счетов из стран за пределами Европы. Этот вызов предназначен для использования

заполнить формы адреса. Окончательная проверка и принятие страны осуществляется через платежного провайдера. Использование функциональности стран выставления счетов повышает точность данных и улучшает пользовательский опыт для вашего отделенного магазина.

-Примечание

Все вызовы документированы в документах OpenAPI в SAP Commerce Cloud:https://HOST:9002/rest/v2/swagger-ui.html

Получение всех стран выставления счетов и доставки

В следующем примере показан вызов REST API для получения всех стран выставления счетов:

```
https://{{HOST}}:9002/rest/v2/{{STOREFRONT}}/countries?type=billing
```

В следующем примере показан ответ, который генерируется в результате этого вызова:

```
{
      "страны":          [
            {
                  "изокод":       "АЛ",
                  "имя": "Албания"
            },
            {
                  "изокод": "AD",
                  "имя": "Андорра"
            },
            {
                  "изокод": "AT",
                  "имя": "Австрия"
            },
            . . .
}
```

Список всех стран доставки можно получить, указавтип=доставка,как показано в следующем примере:

```
https://{{HOST}}:9002/rest/v2/{{STOREFRONT}}/countries?type=shipping
```

Вы также можете получить список всех стран, определенных в SAP Commerce Cloud, не указывая тип, как показано в следующем примере:

```
https://{{HOST}}:9002/rest/v2/{{STOREFRONT}}/countries
```

-Примечание

API-вызовGET/страны доставкибыл устарел.

Страны REST API для регионов

API REST Countries для регионов позволяет получить все регионы (например, штаты или провинции), которые связаны с определенной страной. Этот вызов предназначен для заполнения форм адресов, например, для выставления счетов или доставки.

-Примечание

Все вызовы документированы в документах OpenAPI в SAP Commerce Cloud:https://HOST:9002/rest/v2/swagger-ui.html

Получение всех регионов

В следующем примере показан вызов REST API для получения всех регионов для указанной страны. Страна, указанная в этом примере, — Соединенные Штаты:

```
https://{{HOST}}:9002/rest/v2/{{STOREFRONT}}/countries/us/regions
```

В следующем примере показан ответ, который генерируется в результате этого вызова:

```
{
      "регионы":     [
            {
                  "изокод":        "США-АЛ",
                  "имя": "Алабама"
            },
            {
                  "isocode": "US-AK",
                  "имя": "Аляска"
            },
            {
                  "isocode": "US-AS",
                  "имя": "Американское Самоа"
            },
            . . .
}
```


Сценарии отмены и возврата заказа

Пошаговое руководство по последовательностям вызовов, которые следует использовать для выполнения заданного сценария.

Обзор

Приведенные ниже сценарии представляют собой полное пошаговое руководство по использованию заданной последовательности вызовов.

-Примечание

Примеры URL используются только для презентационных целей. В настроенной среде необходимо заменить адрес сервера на свой собственный.

Отменить заказ

Предпосылки

- Клиент зарегистрирован.
- У клиента есть заказ, который еще не отправлен.
    - Подробную информацию о звонках в процессе оформления заказа см.Сценарии процесса покупки клиента иОтдельные звонки для сценариев покупки клиентами .

Шаги сценария

- Получите детали заказа.
- Отменить заказ.

Запрос потока

Метод	URL	Параметры запроса	Бо Ко
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/orders/00000001	поля=ПО УМОЛЧАНИЮ	
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/orders/00000001/cancellation		

Метод	URL	Параметры запроса	Бо Ко

Возврат заказа

Предпосылки

- Клиент зарегистрирован.
- У клиента есть заказ, который частично отправлен.
    - Подробную информацию о звонках в процессе оформления заказа см.Сценарии процесса покупки клиента иОтдельные звонки для сценариев покупки клиентами .
    - Более подробную информацию о процессе отправки заказов с помощью OMS см.Панель поддержки клиентов бэк-офиса — заказы и возвраты .

Шаги сценария

- Получите детали заказа.
- Возврат заказа.

Запрос потока

Метод	URL	Параметры запроса	Параметры тела Тип содержимого: приложение
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/orders/00000001	поля=ПО УМОЛЧАНИЮ	

ПОЧТА	https://localhost:9002/rest/v2/electronics/users/current/orderReturns		{
			"orderCode
			"returnReq
			{
			"о
			"д
			},
			{
			"о
			"д
			}
			]
			}

Отменить запрос на возврат

Предпосылки

- Клиент зарегистрирован.

- У клиента есть заказ, который частично отправлен.
    - Подробную информацию о звонках в процессе оформления заказа см.Сценарии процесса покупки клиента иОтдельные звонки для сценариев покупки клиентами .
    - Более подробную информацию о процессе отправки заказов с помощью OMS см.Панель поддержки клиентов бэк-офиса — заказы и возвраты .
- У клиента есть отменяемый запрос на возврат.

Шаги сценария

- Получайте запросы на возврат.
- Получить детали запроса на возврат
- Отмените запрос на возврат.

Запрос потока

Метод	URL	Параметры запроса	Параметры тела
			Тип контента
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/orderReturns	поля=ПО УМОЛЧАНИЮ	
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/electronics/users/current/orderReturns/00000000	поля=ПО УМОЛЧАНИЮ	
ПЛАСТЫРЬ	https://localhost:9002/rest/v2/electronics/users/current/orderReturns/00000000		{"статус

Сохранить корзину в OCC

Функция сохранения корзины позволяет сохранять и восстанавливать сохраненные корзины позднее.

Обзор

Функциональность сохранения корзины предоставляется как набор методов, встроенных в независимые стратегии, которые могут быть подключены относительно различных бизнес-требований и реализаций front-end, для которых они используются. Эти стратегии можно легко расширить с помощью pre/post-hooks.

В настоящее время поддерживаются следующие операции сохранения корзины:

- Сохранить корзину сеанса как сохраненную корзину
- Сохраните определенные идентификаторы корзин для внутренних операций в качестве сохраненных корзин
- Отобразить список сохраненных корзин
- Отображение деталей сохраненной корзины
- Восстановить сохраненную корзину в активную корзину сеанса

- Удалить сохраненные корзины
- Клонировать сохраненные корзины
- Отредактируйте название и описание сохраненной корзины.

Изменения в веб-сервисах

Для поддержки функции сохранения корзиныusсommerceвеб-сервисыикоммерциявебсервисcommonsрасширения были расширены. В следующих разделах описываются изменения, связанные с сохранением корзины, которые были внесены в эти расширения.

СохранитьCartController

Вusсommerceвеб-сервисырасширение,СохранитьCartControllerкласс был добавлен. Этот класс делегирует входящие вызовы сохранения корзины реализацииСохранитьCartFacadeинтерфейс, который и выполняет фактическую работу.

Бобы

Вкоммерцияwebservicecommons/resources/коммерцияwebservicecommons-spring.xmlle, вы можете настроить передачу данных объект, который служит промежуточным объектом между моделями в ServiceLayer и клиентом веб-службы. Ниже приведен пример:

```
<боб>
. . .
<боб    class="de.hybris.platform.commercewebservicescommons.dto.order.Sa
<свойство    имя="savedCartData" тип="de.hybris.platform.commercewebserv
</боб>

</бобов>
```

Сохранить корзину Ресурс REST Вызовы

Ниже приведен список всех вызовов, связанных с функцией сохранения корзины:

- /users/{userId}/carts/{cartId}/сохранить
- /users/{userId}/carts/{cartId}/savedcart
- /users/{userId}/carts
- /users/{userId}/carts/{cartId}/flagfordeletion
- /users/{userId}/carts/{cartId}/restoresavedcart
- /users/{userId}/carts/{cartId}/clonesavedcart

-Примечание

Вы также можете использовать метод /users/{userId}/carts/{cartId}/save для обновления существующей сохраненной корзины, указав имя и/или описание корзины.

Картирование DTO и конфигурация ответа

Механизм сопоставления данных упрощает сопоставление данных между исходными и целевыми объектами.

Обзор

Механизм сопоставления данных поддерживает конфигурирование полей, что означает, что вы можете предоставить список полей, которые вы хотите сопоставить. Механизм сопоставления данных может использоваться в веб-службах для сопоставления данных между объектами модели и DTO веб-служб или между объектом данных уровня коммерции и DTO веб-служб. Подробнее о механизме сопоставления данных см.:Механизм отображения данных .

Настройка механизма отображения

Основная функциональность раскрывается черезDataMapperИнтерфейс, который предоставляет несколько методов для сопоставления между исходными и целевыми объектами.DefaultDataMapperиспользуетFieldSetBuilderиGeneralFieldFilterдля сопоставления только указанных полей объекта назначения. Чтобы использовать этот механизм сопоставления данных в расширении веб-сервисов, вам необходимо определитьDataMapper, GeneralFieldFilterи FieldSetBuilderв веб-контексте. Основная конфигурация находится в /ycommercewebservices/web/webroot/WEB-INF/config/v2/dto-mappings-v2-spring.xmlle. Конфигурация, связанная сFieldSetBuilderописан в Полях
Раздел конфигурации ниже.

```
. . .
    <!-- DataMapper - - >
    <псевдоним псевдоним="dataMapper" имя="defaultDataMapper"/>
    <боб id="defaultDataMapper" class="de.hybris.platform.webservicescommons.mapping.impl.DefaultDataMapper">
        <свойство имя="fieldSetBuilder" ref="fieldSetBuilder"/>
    </боб>

    <!-- Орика: Фильтры -->
    <bean class="de.hybris.platform.webservicescommons.mapping.filters.GeneralFieldFilter">
        <имя свойства="fieldSelectionStrategy" ref="fieldSelectionStrategy"/> </bean>

. . .
```

Настройка механизма отображения

Механизм отображения основан на Orika, популярном фреймворке Java Bean mapper. Вы можете использовать все функции, доступные в поддерживаемых версиях Orika, либо создавая пользовательские классы, либо настраиваяDataMapperвыполнение.

-Примечание

TheWsDTOMappingаннотация — это то, что вам понадобится в процессе настройки, потому чтоDefaultDataMapperрегистрирует только аннотированные бины в Orica Mapper Factory. Вот почему вам нужно аннотировать ваши пользовательские конвертеры, мапперы и фильтры с помощью WsDTOMapping.

-Примечание

При создании пользовательского картографа вы также можете расширитьAbstractCustomMapper -этот класс уже аннотирован и дополнительно поддерживает механизм выбора полей.

Ниже вы можете найти пример картографов и конвертеров, определенных вycommerceвеб-сервисырасширение bdto-отображения-v2-весна.xmlле.

```
. . .
     <!-- Orika : Mappers --> <bean
        class="de.hybris.platform.ycommercewebservices.mapping.mappers.AddressValidationDataMapper"
        parent="abstractCustomMapper"/>
    <боб class="de.hybris.platform.ycommercewebservices.mapping.mappers.SpellingSuggestionMapper"
        parent="abstractCustomMapper"/>
    <боб class="de.hybris.platform.ycommercewebservices.mapping.mappers.CCPaymentInfoDataMapper"
        parent="abstractCustomMapper"/>

    <!-- Орика: Конвертеры --> <bean
        class="de.hybris.platform.ycommercewebservices.mapping.converters.StockLevelStatusConverter"/>
    <боб class="de.hybris.platform.ycommercewebservices.mapping.converters.OrderStatusConverter"/>
    <боб class="de.hybris.platform.ycommercewebservices.mapping.converters.ConsignmentStatusConverter"/>
    <боб class="de.hybris.platform.ycommercewebservices.mapping.converters.DeliveryStatusConverter"/>
    <боб class="de.hybris.platform.ycommercewebservices.mapping.converters.ProductReferenceTypeEnumConverter"/
. . .
```

FieldMapper

Если вам нужно сопоставить только одно поле с другим с другим именем, вы можете использовать de.hybris.platform.webservicescommons.mapping.config.FieldMapperкласс илиfieldMapperабстрактный бин. Чтобы узнать как, см. конфигурацию вdto-отображения-v2-spring.xmlниже:

```
. . .
    <!-- Сопоставление полей: Пользователь
    --> <bean parent="fieldMapper">
        <свойство    имя="исходныйКласс"
                     значение="de.hybris.platform.commercewebservicescommons.dto.user.UserSignUpWsDTO"/>
        <свойство    имя="destClass"
                     значение="de.hybris.platform.commercefacades.user.data.RegisterData"/>
        <свойство    имя="fieldMapping">
            <карта>
                <запись   ключ="uid" значение="логин"/>
            </карта>
        </свойство>
    </боб>
    <bean parent="fieldMapper">
        <свойство    имя="исходныйКласс"
                     значение="de.hybris.platform.commercefacades.user.data.CustomerData"/>
        <свойство    имя="destClass"
                     значение="de.hybris.platform.commercewebservicescommons.dto.user.UserWsDTO"/>
        <свойство    имя="fieldMapping">
            <карта>
                <запись   ключ="defaultShippingAddress" значение="defaultAddress"/>
            </карта>
        </свойство>
    </боб>
. . .
```

Конфигурация полей

TheDataMapperподдерживает конфигурацию полей, что означает, что вы можете предоставить список полей, которые вы хотите иметь в ответе. Вы также можете создать уровни полей, которые являются предварительно настроенными наборами (например, BASIC, DEFAULT, FULL) для использования в дальнейшем при работе с картографом.

DefaultDataMapperиспользует конструктор набора полей для разбора конфигурации полей. Компонент конструктора набора полей по умолчанию определен в /ycommercewebservices/web/webroot/WEB-INF/config/v2/dto-level-mappings-v2-spring.xml

```
. . .
    <псевдоним   псевдоним="fieldSetBuilder" имя="defaultFieldSetBuilder"/>
    <боб      идентификатор="defaultFieldSetBuilder"
              class="de.hybris.platform.webservicescommons.mapping.impl.DefaultFieldSetBuilder"> <свойство
                  name="defaultRecurrencyLevel" name="defaultMaxFieldSetSize"
    <свойство     name="fieldSetLevelHelper" ref="fieldSetLevelHelper"/>
    <свойство
    </боб>
    <псевдоним   alias="fieldSetLevelHelper" id="defaultFieldSetLevelHelper"/>
    <боб    class="de.hybris.platform.webservicescommons.mapping.impl.DefaultFieldSetLevelHelper"> </bean>

. . .
```

Определение уровня поля

Уровни полей — это предопределенный набор полей, который можно определить в конфигурации Spring. Уровни набора полей по умолчанию для класса DTO можно найти в Файл /ycommercewebservices/web/webroot/WEB-INF/config/v2/dto-level-mappings-v2-spring.xml

```
. . .
    <bean parent="fieldSetLevelMapping">
        <свойство    имя="dtoClass"
                     значение="de.hybris.platform.commercewebservicescommons.dto.user.UserGroupWsDTO"/>
        <свойство    имя="levelMapping">
            <карта>
                <запись   key="BASIC" value="membersCount,subGroups,members,uid,name"/>
```

```
                    <запись  ключ="ПО УМОЛЧАНИЮ"
                             значение="membersCount,subGroups(DEFAULT),members(DEFAULT),uid,name"/>
                    <запись  key="FULL"
                             значение="membersCount,subGroups(FULL),members(FULL),uid,name"/>
                </карта>
            </свойство>
        </боб>

        <bean parent="fieldSetLevelMapping">
            <свойство    имя="dtoClass"
                         значение="de.hybris.platform.commercewebservicescommons.dto.user.PrincipalWsDTO"/>
            <свойство    имя="levelMapping">
                <карта>
                    <запись  key="BASIC" value="uid,name"/>
                    <запись  key="DEFAULT" value="uid,name"/>
                    <запись  key="FULL" value="uid,name"/>
                </карта>
            </свойство>
        </боб>
    . . .
```

Ответы об ошибках OCC

Рекомендации по использованию ответов об ошибках OCC.

Обзор

Механизм обработки ошибок OCC определен ввеб-сервисысommonsрасширение. Механизм имеет несколько преимуществ:

- Предоставление общих форматов ответов об ошибках
- Отображение статуса ответа для исключений

TheRestExceptionResolverкласс заменяетRestHandlerExceptionResolver.TheRestExceptionResolverпозволяет вам определить общие сообщения об ошибках, чтобы внутренняя информация не просочилась. Примером внутренней информации является структура класса.

RestHandlerExceptionResolver

-Примечание

TheRestHandlerExceptionResolverустарело с версии 2105.

TheRestHandlerExceptionResolverкласс является устаревшей реализациейHandlerExceptionResolverИнтерфейс Spring. Он устарел с версии 2105 и далее.RestHandlerExceptionResolverкласс находится ввеб-сервисысommonsрасширение. Подробности см.: RestHandlerExceptionResolver .

Ниже приведен пример конфигурации Spring для предыдущей версииycommerceвеб-сервисырасширение.

```
<bean id="abstractRestHandlerExceptionResolverV2" abstract="true">
    <имя свойства="webserviceErrorFactory" ref="webserviceErrorFactory" /> <имя
    свойства="messageConverters" ref="messageConvertersV2" /> </bean>


<bean id="restHandlerExceptionResolverV2" class="de.hybris.platform.webservicescommons.resolver.RestHandlerExce
    parent="abstractRestHandlerExceptionResolverV2"> <свойство
                имя="propertySpecificKey"          значение="ycommercewebservices"/>
    <свойство   имя="configurationService"           ref="configurationService"/>
</боб>

<util:list id="exceptionResolversV2">
```

```
        <ref bean="restHandlerExceptionResolverV2" </util:list>     />
```

Конгурация

Конфигурацию можно задать в свойствахycommercewebservices/project.propertiesle в следующем формате:

```
webservicescommons.resthandlerexceptionresolver.{extensionName}.{exceptionName}.logstack=true или false
webservicescommons.resthandlerexceptionresolver.{extensionName}.{exceptionName}.status=HTTP_STATUS_CODE
```

Пример конфигурации показан здесь:

```
webservicescommons.resthandlerExceptionresolver.ycommercewebservices.CartAddressException.logstack=true
webservicescommons.resthandlerExceptionresolver.ycommercewebservices.CartAddressException.status=400
```

-Примечание

Простое имя класса определяет код статуса. Каноническое имя класса не определяет код статуса. Имя исключения должно быть уникальным.

RestExceptionResolver

TheRestExceptionResolverкласс является реализациейHandlerExceptionResolverИнтерфейс Spring. RestExceptionResolverкласс находится ввеб-сервисыcommonsрасширение. Класс используется вycommerceвеб-сервисы

расширение для обработки исключений. Для получения более подробной информации см.RestExceptionResolver .

```
<боб    id="restHandlerExceptionResolverV2" class="de.hybris.platform.webservicescommons.resolver.RestExceptionRe
        parent="wsBaseRestExceptionResolver">
    <имя свойства="webserviceErrorFactory" ref="webserviceErrorFactory" /> <имя
    свойства="messageConverters" ref="messageConvertersV2" /> <имя
    свойства="extensionName" value="ycommercewebservices" /> </bean>


<util:list id="exceptionResolversV2">
    <ref bean="restHandlerExceptionResolverV2" </util:list>     />
```

Конгурация

Конфигурация может быть выполнена через свойства вycommercewebservices/project.propertiesle. Новое реализованное решение позволяет вам определять такие свойства, какmessageFormatterTypeисообщение.Примеры конфигурации показаны ниже.

Этот пример демонстрируетИсключение адреса корзиныпересылается как есть:

```
webservicescommons.resthandlerExceptionresolver.ycommercewebservices.CartAddressException.logstack = true
webservicescommons.resthandlerExceptionresolver.ycommercewebservices.CartAddressException.status=400
webservicescommons.resthandlerExceptionresolver.ycommercewebservices.CartAddressException.messageFormatterType=
```

Этот пример демонстрируетИсключение адреса корзиныбудучи покрытымОБЩИЙсообщение:

```
webservicescommons.resthandlerExceptionresolver.ycommercewebservices.CartAddressException.logstack=true
webservicescommons.resthandlerExceptionresolver.ycommercewebservices.CartAddressException.status=400
webservicescommons.resthandlerExceptionresolver.ycommercewebservices.CartAddressException.message=Корзина                адрес
webservicescommons.resthandlerExceptionresolver.ycommercewebservices.CartAddressException.messageFormatterType=
```

WebserviceErrorFactory

TheWebserviceErrorFactoryиспользуется в механизме обработки ошибок для преобразования исключений вОшибкаWsDTOобъекты. Его реализация по умолчанию DefaultWebserviceErrorFactoryиспользует списокАннотацияОшибкаКонвертерв процессе преобразования. Вы можете повлиять на то, как исключения будут преобразованы вОшибкаWsDTOобъект, добавив ваш конвертер в этот список в /ycommercewebservices/web/webroot/WEB-INF/config/common/error-config-spring.xml.

```
. . .
    <alias alias="webserviceErrorFactory" name="defaultWebserviceErrorFactory" /> <bean
    id="defaultWebserviceErrorFactory"                класс="de.hybris.platform.webservicescommons.errors.factory.impl.D
        <имя свойства="конвертеры">
            <список>
                <реф  bean="validationErrorConverter" />
                <реф  bean="cartVoucherValidationListErrorConverter"                />
                <реф  bean="cartModificationDataListErrorConverter"                />
                <реф  bean="webserviceExceptionConverter" />
                <реф  bean="exceptionConverter" />
            </список>
        </свойство>
    </боб>
. . .
```

-Примечание

Все конвертеры проверяются на предмет поддержки объектов проверки. Поэтому возможно наличие двух разных конвертеров, которые поддерживают один и тот же объект, но выдают разные ошибки.

Конвертеры по умолчанию

По умолчанию предоставляются конвертеры для следующих типов:

Конвертеры по умолчанию

Ошибка преобразователя	Поддерживаемые типы
ValidationErrorConverter	org.springframework.validation.Ошибки
КорзинаМодификацияСписок данныхОшибкаКонвертер	de.hybris.platform.commercefacades.order.data.CartModificationDataList
КорзинаМодификацияДанныеОшибкаКонвертер	de.hybris.platform.commercefacades.order.data.CartModificationData
WebserviceExceptionConverter	de.hybris.platform.webservicescommons.errors.converters.WebserviceExcept
ИсключениеКонвертер	java.lang.Exception исключая потомков de.hybris.platform.webservicescommons.errors.converters.WebserviceExcept

Преобразователи HTTP-сообщений

API OCC позволяет преобразовывать объекты DTO в их текстовое представление (JSON или XML), используемое в вызовах REST, или обратно.

OCC позволяет преобразовывать объекты DTO в/из их текстового представления (JSON или XML), используемого в вызовах REST. Для каждого вызова REST выходные данные возвращаются как стандартный объект DTO. Объекты DTO — это обычные классы Java, которые автоматически генерируются стандартным механизмом платформы на основекоммерцияявебсервисыcommons-beans.xmlle. Эти объекты также могут использоваться в качестве параметров. Для использования в вызовах REST объекты DTO преобразуются в/из текста (JSON/XML) с помощью механизма преобразования, доступного ввеб-сервисыcommons Расширение.ycommerceвеб-сервисырасширение настроено на использование определенных в нем преобразователей HTTP-сообщений.

```
. . .
<псевдоним имя="jaxbMessageConverters" псевдоним="messageConvertersV2" />
```

```
. . .


. . .
      <bean id="abstractRestHandlerExceptionResolverV2" abstract="true">
          <имя свойства="webserviceErrorFactory" ref="webserviceErrorFactory" /> <имя
          свойства="messageConverters" ref="messageConvertersV2" /> </bean>

. .


. . .
       <bean id="oAuth2ExceptionRendererV2"          parent="oAuth2ExceptionRenderer">
          <имя_свойства="messageConverters" </        ref="messageConvertersV2"/>
          bean>
. .
```

Выбор формата ответа

По умолчанию толькоПриниматьзаголовок включен и используется в согласовании содержимого для входящих запросов OCC API. Согласование содержимого с использованием параметра запроса формата и расширения пути отключено в OCC API. Согласно документации Spring Framework, использование расширений пути не рекомендуется и считается устаревшим с версии 5.3. В качестве альтернативы в OCC можно включить резервный режим и использовать параметр запроса формата для выражения приемлемого формата ответа.

Чтобы включить резервный режим, добавьте в конфигурацию следующее свойство:

```
ycommercewebservices.content.negotiation.legacy=true
```

Сопутствующая информация

Преобразователи HTTP-сообщений

Концепция WsDTO

WsDTO — это уровень данных, используемый REST API в OCC.

Концепция

Модель WsDTO добавляет стабильности REST API, устраняя зависимость от модели данных коммерческих сервисов. В результате изменения в модели данных коммерческих сервисов не влияют напрямую на REST API. Она также повышает гибкость, внедряя механизм, позволяющий контролировать возвращаемые поля. Таким образом, теперь вы можете влиять на то, как будет выглядеть возвращаемый объект WsDTO. Вы можете явно запросить заполнение определенных полей в объекте или использовать заранее определенный набор полей.

Стабильный API

В v1 REST API сервисы возвращают объекты данных, что может привести к неожиданному поведению при добавлении или изменении полей в модели данных коммерческих сервисов. Каждое изменение модели данных в коммерческих сервисах может повлиять на REST API и, как следствие, на совместимость с потребителями сервисов. Уровень WsDTO устойчив к изменениям в модели данных коммерческих сервисов, что обеспечивает стабильность API.

Динамическая конфигурация поля

Новый слой WsDTO был представлен с новым механизмом, который позволяет определять поля для возврата через API. По умолчанию есть три конфигурации:

- BASIC — заполняется только базовый набор полей, идентифицирующих объект. Например, имя, код и идентификатор. DEFAULT — средний
- набор полей, определенных для наиболее распространенных вариантов использования.
- ПОЛНЫЙ - возвращаются все поля.

Все наборы могут быть изменены или новые могут быть добавлены на стороне сервера путем предоставления определенной конфигурации. Потребители могут влиять на возвращаемые поля, явно отправляя их в качестве параметра или неявно отправляя имя набора в параметре.

Структура WsDTO

Объекты WsDTO изначально создавались максимально приближенными к объектам данных коммерческих услуг, поэтому сопоставление выполняется просто и автоматически с помощью Orika mapper. Они создаются вкоммерциявебсервисыcommons-beans.xmlle как обычные объекты bean.

```
. . .
        <bean class="de.hybris.platform.commercewebservicescommons.dto.order.CartWsDTO"
            extends="de.hybris.platform.commercewebservicescommons.dto.order.AbstractOrderWsDTO"> <свойство
                name="totalUnitCount" type="Integer"/> name="potentialOrderPromotions"
        <свойство    type="java.util.List<de.hybris.platform.commercewebservicescommons.dto.product.PromotionRe
                name="potentialProductPromotions"
        <свойство
                type="java.util.List<de.hybris.platform.commercewebservicescommons.dto.product.PromotionRe
    </боб>
        . . .
```

При добавлении нового объекта WsDTO рекомендуется также добавить конфигурацию отображения уровня в /WEB-INF/config/v2/dto-level-отображения-v2-spring.xmlле.

Различия в объектах данных коммерческих служб

Несмотря на то, что большинство объектов были определены максимально близко к объектам данных, существуют некоторые исключения.

Никаких универсальных объектов

Нет общих типов WsDTO. Все типы WsDTO являются конкретными объектами без типов параметров, чтобы избежать неоднозначности возвращаемых объектов службами REST.

Упаковка коллекций корневого уровня

Все методы в OCC REST API возвращают объекты WsDTO, что означает, что все типы коллекций, которые могут быть возвращены службами, были упакованы в ListWsDTOтипа. БольшинствоListWsDTOтипы — это просто классы-обертки, имеющие поле, являющееся типом коллекции.

Коллекции внутри объектов WsDTO

Поля в объектах WsDTO противоположны корневым объектам, которые не должны быть заключены вListWsDTOтип, поэтому разрешено определять поля как типы коллекций. Например, списки.

Изменения в API

После внедрения нового слоя WsDTO пришлось изменить и OCC REST API. Самое важное изменение заключается в том, что ни один метод больше не возвращает объекты данных коммерческих сервисов. Каждый объект Data имеет свой аналог в WsDTO и соответствующее сопоставление. Все методы OCC REST API в v2 изменили свои сигнатуры для возврата объектов WsDTO.

Сопутствующая информация

Картирование DTO и конфигурация ответа

Оплата в OCC

Описание потока вызовов процесса оплаты, включая определение платежных реквизитов и авторизацию карты.

Потоки платежей

Процедура оплаты является важнейшей частью процесса оформления заказа клиентом. В рамках этого процесса можно выделить два основных этапа:

. Определение платежных реквизитов.

. Авторизация карты.

Поток отправки платежных реквизитов

В этом платежном потоке два основных шага, описанных выше, могут быть достигнуты с помощью следующих запросов:

Метод	URL	Описание
ПОЧТА	https://localhost:9002/rest/v2/{baseSiteId}/users/{userId}/carts/{cartId}/paymentdetails	Денес оплата подробности для клиент.
ПОЧТА	https://localhost:9002/rest/v2/{baseSiteId}/users/{userId}/orders	Уполномочивает карта и места заказ.

-Примечание

На использование платежного потока SOP распространяются следующие ограничения:

- Этот способ оплаты не рекомендуется для большинства случаев использования, поскольку для работы с серверами SAP Commerce требуется соответствие PCI.
- Этот платежный поток не поддерживается дляцисКиберресурспоставщик платежей - доступен вцисплатежAddOn. Если установлен addOn cispayment и поставщиком платежных услуг для базового магазина является cisCybersource, тоНеподдерживаемое исключение запросавыдается для запроса на создание платежных реквизитов.

Потоки платежей SOP

Эти платежные потоки позволяют пользователю использовать платежный шлюз Silent Order POST для кредитных карт. Клиент создает подписку на платеж, отправляя запрос напрямую поставщику платежей.

Существует два возможных потока платежей SOP. Основное различие между ними заключается в способе создания платежных реквизитов на основе ответа платежного провайдера:

. В первом случае платежные реквизиты определяются, когда клиент пересылает ответ платежного провайдера в OCC. В этом случае обратный вызов продавца устанавливает только флаг проверки.

Во втором потоке реквизиты платежа определяются в обратном вызове продавца.

Требуемые расширения

-Примечание

Чтобы использовать платеж SOP, вам необходимо установитьacceleratorwebservicesaddon Дополнение .

Этот аддон расширяет API OCC следующими вызовами:

- /v2/{baseSiteId}/users/{userId}/carts/{cartId}/payment/sop/request
- /v2/{baseSiteId}/users/{userId}/carts/{cartId}/payment/sop/response

Theускорительвеб-сервисыдополнениеДополнение зависит отускорительфасады иуслуги ускорителя расширения, где оплата SOP Функциональность определена.услуги ускорителярасширение также предоставляет макет платежного шлюза, который полезен в процессе разработки.

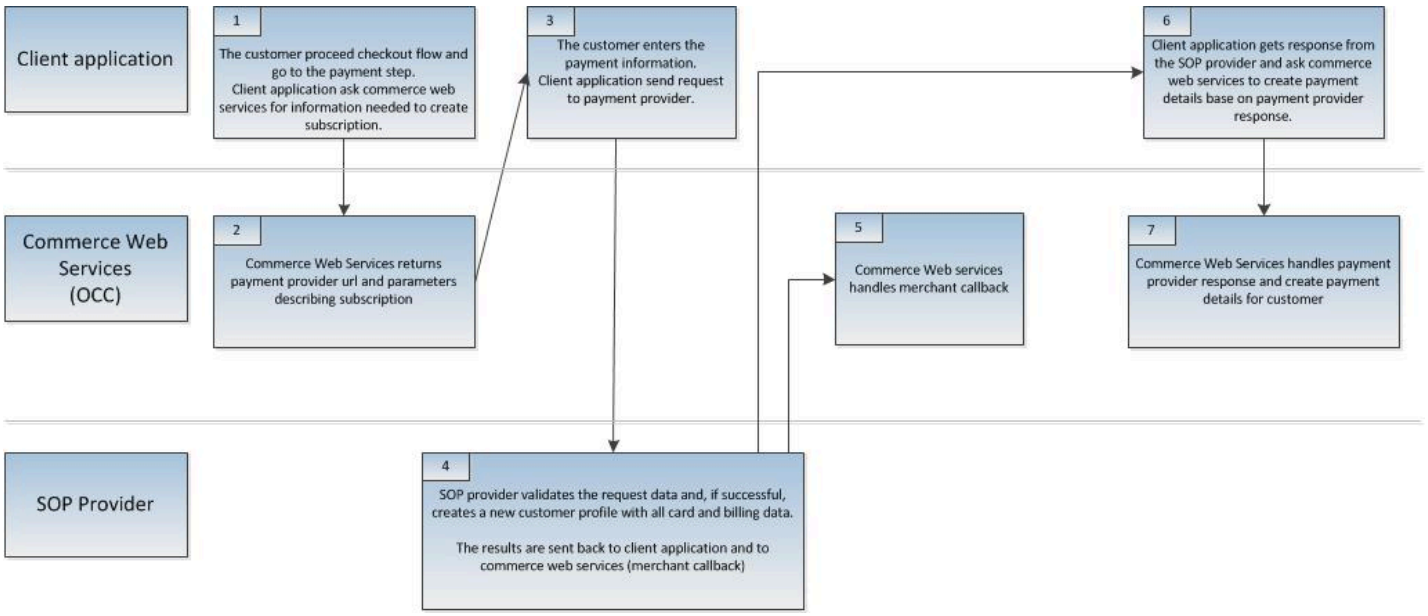
-Примечание

Подробную информацию о запросах СОП см. в доступных ресурсах, описанных вacceleratorwebservicesaddon Дополнение документ.

Поток платежей по СОП

На рисунке ниже представлен обзор платежного потока SOP. Он представляет два ответа от поставщика платежа:

- первый ответ отправляется клиентскому приложению — данные из этого ответа пересылаются в OCC и определяются реквизиты платежа.
- второй ответ отправляется напрямую в OCC (обратный вызов продавца). Затем он проверяется, и в платежных реквизитах устанавливается флажок проверки подписки.



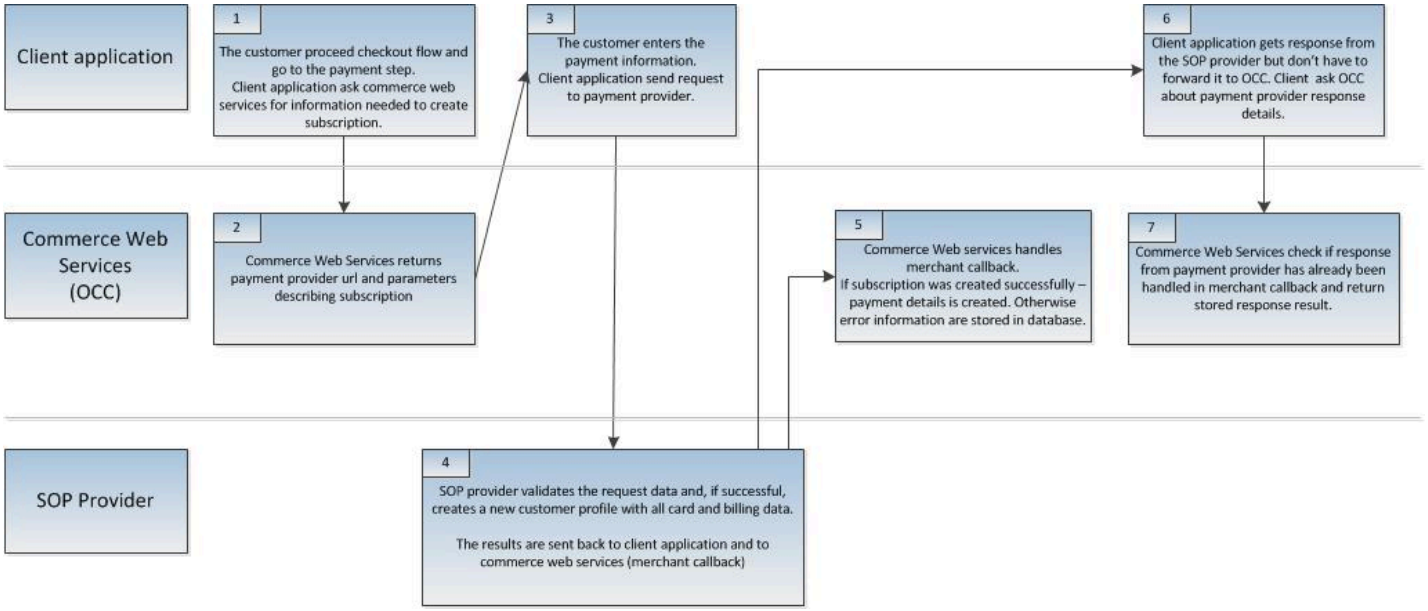
Чтобы завершить шаг создания платежных реквизитов в этом потоке, вам необходимо отправить запросы, перечисленные ниже:

Метод	URL	Описание
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/{baseSiteId}/users/{userId}/carts/{cartId}/payment/sop/request?responseUrl=sampleUrl	Получает информация необходимо для создание подписка контактируя напрямую с the оплата поставщик
ПОЧТА	postUrl получен в предыдущем запросе	Создает подписка контактируя напрямую с the оплата провайдер.
ПОЧТА	https://localhost:9002/rest/v2/{baseSiteId}/users/{userId}/carts/{cartId}/payment/sop/response	Ручки ответ от оплата поставщик и создает оплата подробности.

Метод	URL	Описание
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/{userId}/orders	Уполномочивает карта - это остается то же самое, что и в случай стандарт вл.

Поток платежей SOP с расширенным обратным вызовом торговца

На рисунке ниже представлен обзор расширенного платежного потока SOP. Он показывает, что обработка обратного вызова продавца для этого потока сложнее. OCC хранит информацию от поставщика платежей, чтобы иметь возможность информировать клиента о результате процесса создания подписки. Если он успешен, обработчик обратного вызова продавца также определит платежные реквизиты для клиента.



Чтобы завершить шаг создания платежных реквизитов в этом потоке, вам необходимо отправить запросы, перечисленные ниже:

Метод	URL	Описание
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/{baseSiteId}/users/{userId}/carts/{cartId}/payment/sop/request ?responseUrl=sampleUrl& extendedMerchantCallback=true	Получает информацию подписка платеж пров -Примечание Это важно расширенный потому что это URL будет изменен перезвонить.
ПОЧТА	postUrl получен в предыдущем запросе	Создает подпрограмму с оплатой
ПОЛУЧАТЬ	https://localhost:9002/rest/v2/{baseSiteId}/users/{userId}/carts/{cartId }/payment/sop/response	Получить информацию ответ WHi обратный вызов. следующее значение • 200 (возврат шаг о

Метод	URL	Описание
		• 202 (получать оплата клиент • 400 (прови подпункт информировать возврат
ПОЧТА	https://localhost:9002/rest/v2/electronics/users/{userId}/orders	Уполномочивает са в случае

Удаление сохраненных ответов поставщика платежных услуг

В этом платеже провайдера ответы хранятся в базе данных, чтобы клиент мог получить их от OCC. Они автоматически удаляются во время процесса размещения заказа, но вы должны помнить об удалении их между отправкой нескольких запросов платежному провайдеру. Для получения дополнительной информации см.: Оформление заказа с помощью SOP и расширенный обратный вызов продавца вСправочник вызовов - v2 . Также есть cronjob для удаления старых ответов платежного провайдера. определено:oldPaymentSubscriptionResultRemovalCronJob.

Конфигурация, необходимая для SOP

В таблице ниже представлено описание основных свойств, используемыхуслуги ускорителярасширение.

Ключ от собственности	Описание
sop.post.url	Это URL для Silent Order Post. Рекомендуется использовать защищенное соединение для безопасности данных. Пример : #sop.post.url=https://orderpagetest.ic3.com/hop/ProcessOrder.do sop.post.url=/ acceleratorservices/sop-mock/process
hop.cybersource.sharedSecret	Это ключ, сгенерированный в вашем профиле киберресурса, который подписывает данные транзакции и требуется для каждой транзакции. -Кончик Для настроек, специфичных для сайта, в проекте с несколькими витринами добавьте название вашего сайта. Пример : hop.cybersource.sharedSecret.wsIntegrationTest=MIGfMA0GCSqGSIb3DQEBAQUAA_Your_sha
hop.cybersource.merchantID	Это присвоенный CyberSource идентификатор продавца, который выдается вам при создании учетной записи. -Кончик Для настроек, специфичных для сайта, в проекте с несколькими витринами добавьте название вашего сайта. Пример : hop.cybersource.merchantID.wsIntegrationTest=your_merchant_id

В таблице ниже представлено описание основных свойств, используемыхacceleratorwebservicesaddon Дополнение .

Ключ от собственности	Описание
webroot.commercewebservices.http webroot.commercewebservices.https	Это базовый URL для коммерческих веб-сервисов. Он необходим для создания полного URL обратного вызова продавца. Пример: webroot.commercewebservices.http=http://localhost:9001/rest webroot.commercewebservices.https=https://localhost:9002/rest

Ключ от собственности	Описание
	-Кончик Для настроек, специфичных для сайта, в проекте с несколькими витринами добавьте название вашего сайта после жebroot.commerceвеб-сервисычасть. Пример: webroot.commercewebservices.wsIntegrationTest.http=http://localhost:9001/rest webroot.commercewebservices.wsIntegrationTest.https=https://localhost:9002/rest

Сопутствующая информация

Расширение ycommercewebservices
acceleratorservices Продление
оплаты Продление

Включение и использование очереди статусов заказов

Очередь обновления статуса заказа включена и доступна по умолчанию вusommerceвеб-сервисырасширение. Однако оно не привязано ни к одному бизнес-процессу, отвечающему за обработку заказов, поскольку обработка заказов — сложный механизм, имеющий множество настраиваемых точек соприкосновения. Поэтому клиент должен указать их все и охватить все возможные изменения статуса заказа.

Если вы используете B2C Accelerator, наиболее интересной отправной точкой будет настройка процесса обработки заказов. (yacceleratorfullfilmentprocess). Другой точкой соприкосновения может бытьссockpitи любое другое место, которое влияет на статус заказа. Разделы ниже расскажут вам, как быстро подключиться к работе заказа, чтобы уведомить очередь об изменении статуса заказа.

Подключение к работе с заказами

Функциональность Order Status Queue использует Spring Messaging System для связи между различными расширениями, не делая их зависимыми друг от друга. Она использует специальный канал сообщений, называемыйorderStatusUpdateChannel.Первое, что вам нужно сделать, чтобы использовать его в yacceleratorfullfilmentprocessэто объявить этот канал в конфигурации пружины, как показано ниже. Измените yacceleratorfullfilmentprocess-spring.xmlle для внесения изменений.

```
<!--Функциональность очереди обновления статуса заказа --
> <int:channel id="orderStatusUpdateChannel"/>
```

Далее вам нужно взглянуть на процесс оформления заказа, который вы найдете взаказ-процесс.xmlle и узнайте, где происходит фактическое изменение статуса заказа. Когда вы найдете нужное место, вы сможете отправить сообщение оттуда:

```
/**
  * Это действие реализует авторизацию платежа с использованием {@link CreditCardPaymentInfoModel}. Любой другой платежный мод
  * быть реализовано здесь или в отдельном действии, если последовательность процесса отличается.
  * /
открытый класс CheckAuthorizeOrderPaymentAction расширяет AbstractSimpleDecisionAction<OrderProcessModel> {

    private MessageChannel productExpressUpdateChannel; // канал должен быть введен здесь

    @Переопределить
    общественный переход выполнитьAction (окончательный процесс OrderProcessModel) {

        окончательный OrderModel заказ = процесс.getOrder();
        если (заказ != null)
        {
            если (order.getPaymentInfo() экземплярInvoicePaymentInfoModel) {
```

```
                возвращаться Переход.OK;
            }
            еще
            {
                для (финальная транзакция PaymentTransactionModel: order.getPaymentTransactions()) {

                    для (окончательная запись PaymentTransactionEntryModel: transaction.getEntries()) {

                        если (entry.getType().equals(ТипПлатежнойТранзакции.АВТОРИЗАЦИЯ)
                                && TransactionStatus.ACCEPTED.name().equals(entry.getTransactionStatus())
                        {

                            заказ.установитьСтатус(СтатусЗаказа.ОПЛАТА_АВТОРИЗОВАНА);
                            sendOrderToChannel(order); // отправить сообщение в очередь обновления статуса заказа return
                            Transition.OK;
                        }
                    }
                }
            }
        }
        возврат Перехода.NOK;
    }

    /**
    * Этот метод помогает отправлять весеннее сообщение в orderStatusUpdateChannel
    * /
    защищенный логический sendOrderToChannel(окончательный заказ OrderModel) {

        пытаться
        {
            окончательное сообщение <OrderModel> сообщение = MessageBuilder.withPayload(order).build();
            return getOrderStatusUpdateChannel().send(message);
        }
        catch (финальное исключение MessageDeliveryException) {

            LOG.warn("Вероятно, прослушиватель для orderStatusUpdateChannel отсутствует", exception);
        }
        вернуть ложь;
    }

    /* ... */
}
```

Вставьте канал сообщения в соответствующее действие взаказ-процесс-весна.xmlле.

```
<bean id="checkAuthorizeOrderPaymentAction" class="de.hybris.platform.yacceleratorfulfilmentprocess.actions.ord
    <имя_свойства="orderStatusUpdateChannel" ref="orderStatusUpdateChannel"/> </bean>
```

Включение заданий Cron

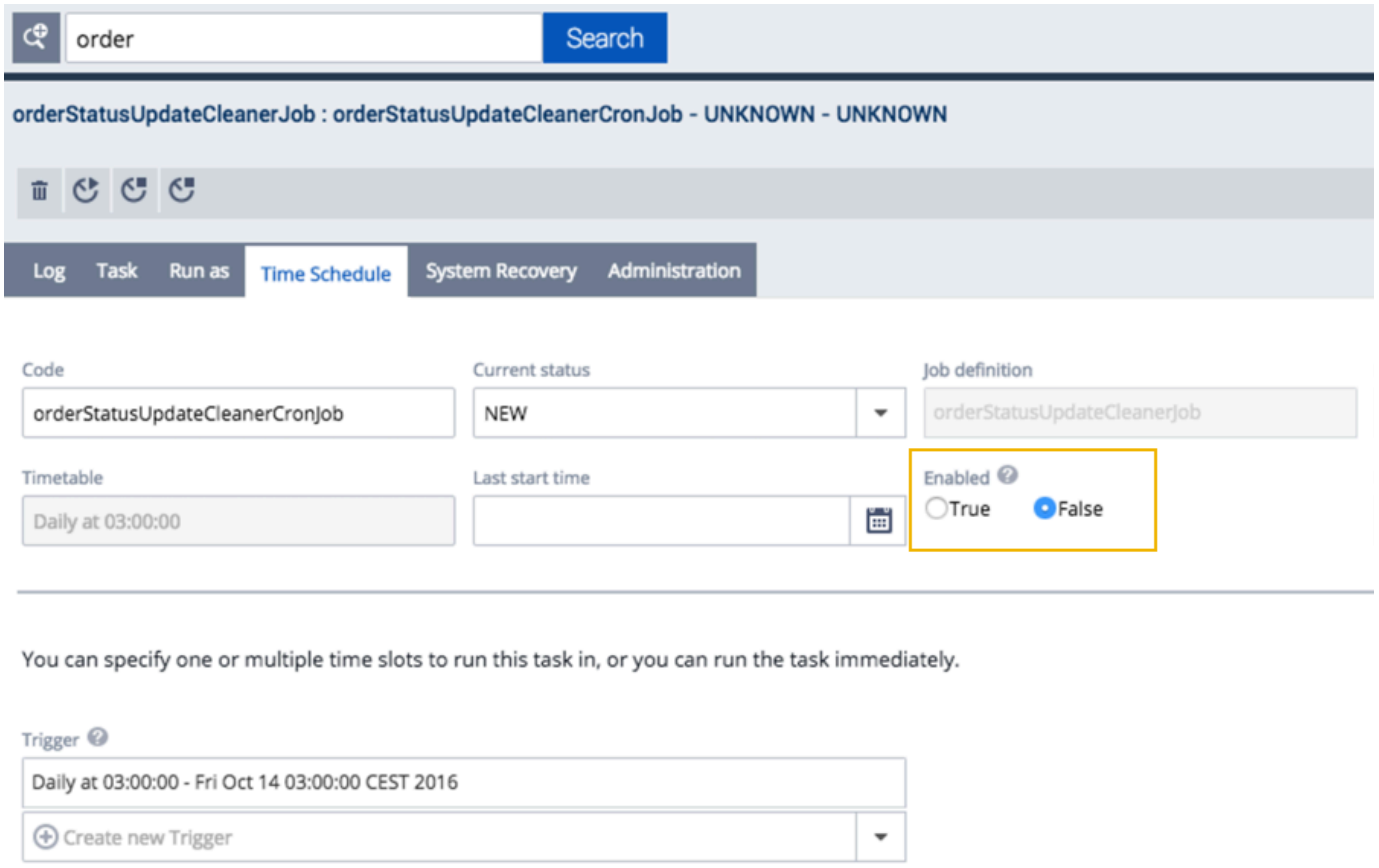
Очередь обновления статуса заказа — это структура в памяти, которую время от времени необходимо очищать.OrderStatusUpdateCleanerРабота предопределен для вас, но не активен по умолчанию.

-Примечание

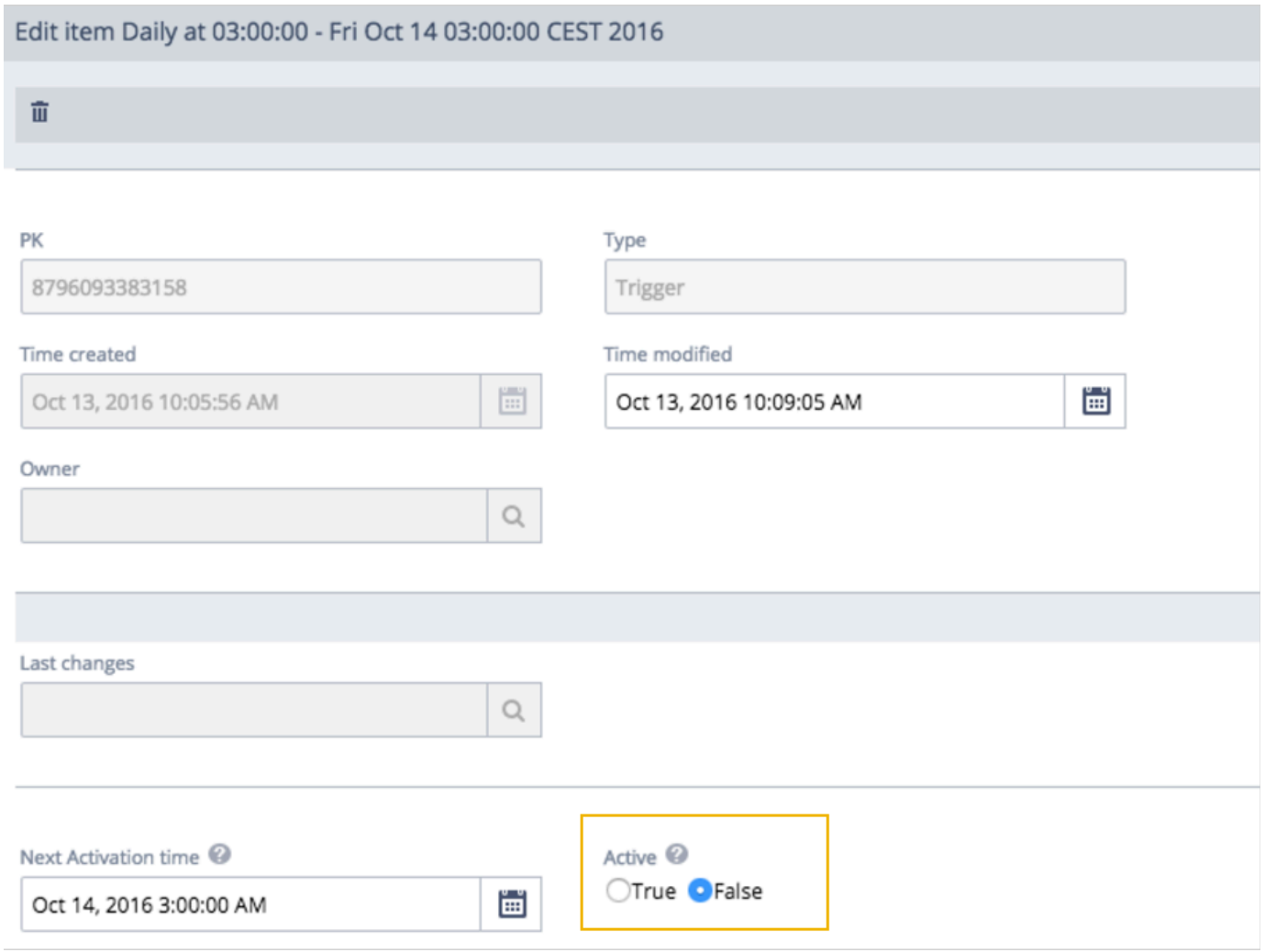
ThezаказСтатусОбновитьУборщикРаботанастройки загружаются вместе сycommercewebservicesДанные проекта расширения. Если данные проекта не загружены, вам необходимо определить cronjob на основезаказСтатусОбновитьУборщикРаботаопределение работы.

Чтобы запустить его, откройте панель администрирования Backoffice и перейдите кСистема > Фоновые процессы > CronJobs > Выберите OrderStatusUpdateCleanerРаботаиз списка и выполните следующие действия:

- УстановитеВключеноag в True дляOrderStatusUpdateCleanerJob.



- УстановитеАктивныйag в значение True для триггера.



Ресурс отдыха

Конечную точку для доступа к обновлению статуса заказа можно найти в контроллере каналов:

```
GET v2/{site}/feeds/orders/statusfeed?timestamp=2013-12-12T12%3A00%3A00Z
```

конечная точка доступна для роли TRUSTED_CLIENT и возвращает только элементы для указанного базового сайта, обновленные после указанной временной метки.

Сопутствующая информация

Включение экспресс-обновления продукта

Включение резервного языка с помощью OCC

Обеспечьте правильный языковой резерв в случаях, когда определенный язык недоступен для запрашиваемых данных.

Контекст

При запросе данных из интерфейса OCC с параметром языка URL, если нет данных на запрошенном языке, вместо ожидаемых данных резервного языка может быть возвращен пустой контент. Например, если немецкий язык имеет английский в качестве резервного языка, при запросе данных о продукте с параметром URL &яз=нем,и продукт имеет {имя[de]=null и имя[en]=test},то для наименования продукта может быть возвращено пустое значение.

В витрине Accelerator при посещении страниц следующие два атрибута сеанса должны быть установлены в значение TRUE, чтобы включить резервный язык.

```
getSessionService().setAttribute(LocalizableItem.LANGUAGE_FALLBACK_ENABLED, Boolean.TRUE);
getSessionService().setAttribute(AbstractItemModel.LANGUAGE_FALLBACK_ENABLED_SERVICE_LAYER, Boolean.TRUE);
```

В OCC эти два атрибута не установлены по умолчанию. Однако вы можете установить их в пользовательской реализации, чтобы включить резервный язык на основе ваших фактических требований. Настройка также может быть достигнута путем внедренияI18NObслуживание и вызов i18NService.setLocalizationFallbackEnabled(true).

Включение отката с помощью occaddon

Следуйте этому процессу для дополнений OCC на основеycommercewebservices.

Процедура

Установите требуемые параметры в значение true в любом предпочтительном месте, прежде чем будут извлечены соответствующие данные. Например, следующий пример показывает de.hybris.platform.ycommercewebservices.filter.SessionLanguageFilter.doFilterInternal(HttpServletRequest, HttpServletResponse, FilterChain):

```
@Autowired
частный I18NService i18NService;


@Переопределить
protected void doFilterInternal(финальный запрос HttpServletRequest, финальный ответ HttpServletResponse,
    окончательный FilterChain filterChain) выдает ServletException, IOException
{
  getContextInformationLoader().setLanguageFromRequest(запрос);
  i18NService.setLocalizationFallbackEnabled(true);


  filterChain.doFilter(запрос, ответ);
}
```

Включение отката с расширением occ

Следуйте этому процессу для расширений OCC на основейокк.

Контекст

Веб-сервисы OCC, реализованные с использованиемуоссшаблон зависит откоммерциявеб-услуги.Вкоммерциявеб-услугиthe Фильтр языка сеансаОпределение фильтра находится в следующем файле:

```
/commercewebservices/web/webroot/WEB-INF/config/v2/filter-config-v2-spring.xml
```

Весенняя фасольФильтр языка сеансаявляетсякоммерцияWebServicesSessionLanguageFilterV2,и он собран в коммерцияWebServicesFilterChainListV2,который затем используется как цепочка фильтров.

```
<bean id="commerceWebServicesSessionLanguageFilterV2" class="de.hybris.platform.commercewebservices.core.filter <property
  name="contextInformationLoader" ref="wsContextInformationLoaderV2" />
</боб>

<alias name="defaultCommerceWebServicesFilterChainListV2" alias="commerceWebServicesFilterChainListV2" /> <util:list
id="defaultCommerceWebServicesFilterChainListV2">
    . . .
    <!-- Соответствующие фильтры -->
    <ref    bean="commerceWebServicesEurope1AttributesFilterV2"                />
    <реф  bean="commerceWebServicesSessionLanguageFilterV2"              />
    <реф  bean="commerceWebServicesSessionCurrencyFilterV2"              />
    <реф  bean="cartMatchingFilter" />
```

```
    . . .
  </util:list>
```

Процедура

. ПереопределитьВеб-сервисыSessionLanguageFilterV2в вашем настроенном расширении occ, создав собственную реализацию для замены ФильтрЯзыкаСессии.

Можно использовать как подкласс, так и новый класс, поскольку это переопределение компонента Spring, как в следующем примере.

```
<bean id="commerceWebServicesSessionLanguageFilterV2" class="customized_filter_implementation">
  <имя_свойства="contextInformationLoader" ref="wsContextInformationLoaderV2" /> </bean>
```

. Вы можете управлять списком фильтров вкоммерцияВеб-сервисыФильтрЦепочкаСписокV2с использованием Spring. На него можно ссылаться для настройки цепочки фильтров, как в следующем примере изcmsocc:

/cmsocc/resources/occ/v2/cmsocc/web/spring/cmsocc-web-spring.xml

```
<bean зависит-от="commerceWebServicesFilterChainListV2" parent="listMergeDirective">
  <имя свойства="add" ref="cmsPreviewTicketFilter" /> <имя
  свойства="afterBeanNames">
    <list value-type="java.lang.String">
      <value>commerceWebServicesSessionLanguageFilterV2</value> </list>

  </свойство>
  <имя_свойства="beforeBeanNames">
  <тип_значения_списка="java.lang.String">
      <value>OccConsentLayerFilter</value>
      <value>cxOccPersonalizationFilter</value>
      <value>cartMatchingFilter</value>
      <value>guestRoleFilterV2</value>
    </список>
  </свойство>
</боб>
```

Включение интерактивной документации OCC REST API

Для включения интерактивной документации OCC REST API необходимо определить и авторизовать клиентов OAuth.

Контекст

Интерактивная документация OCC REST API в настоящее время поддерживается в B2C Accelerator.

Процедура

. Зарегистрируйте пользователя на своей витрине.

. Откройте страницу импорта ImpEx в консоли администрирования SAP Commerce:https://localhost:9002/console/impex/import.

. Скопируйте следующие данные ImpEx:

```
INSERT_UPDATE OAuthClientDetails;Идентификатор клиента[уникальный=истина]  ;Идентификаторы ресурсов ;объем          ;авторизованныйGran
                                 ;клиентская сторона                       ;гибрис               ;базовый         ;неявный,клиент
                                 ;мобильный_андроид                        ;гибрис               ;базовый         ;авторизация_
```

Эти данные ImpEx предоставляют вам пользователя с именеммобильный_андроид,и соответствующий пароль,секрет.Эти данные используются для включения интерактивных вызовов в документации OCC REST API.

. Вставьте данные ImpEx вИмпортировать контентокно, затем щелкнитеИмпортировать контент.

. Скопируйте следующие данные ImpEx:

```
INSERT_UPDATE OAuthClientDetails;Идентификатор клиента[уникальный=истина]  ;Идентификаторы ресурсов ;объем          ;авторизованныйGran
                                 ;доверенный_клиент                        ;гибрис               ;расширенный     ;авторизация_
```

Эти данные ImpEx предоставляют вам пользователя с именемдоверенный_клиентс соответствующим паролем. Эти данные используются для включения интерактивных вызовов в документации OCC REST API, требующих расширенных разрешений.

. Вставьте данные ImpEx вИмпортировать контентокно, затем щелкнитеИмпортировать контент.

. Откройте интерактивную документацию OCC REST API в своем веб-браузере, перейдя по следующей ссылке:http://<имя_хоста>:<порт>/rest/v2/swagger-ui.html.

Например, если на вашем локальном компьютере установлена витрина, вы можете получить доступ к документации OCC REST API с помощью следующей команды: связь:https://localhost:9002/rest/v2/swagger-ui.html.

. НажмитеАвторизоватьв верхней части веб-страницы документации OCC REST API.

. ВДоступные разрешенияВ появившемся окне введите учетные данные пользователя вашего магазина в полеИмя пользователяиПарольполя.

. ВТипвыпадающий список, выберитеБазовая аутентификация.

. ВClientIdВ появившемся поле введитемобильный_андроид,и вСекретполе, введитесекрет.

. Выберитебазовыйустановите флажок и нажмитеАвторизовать.

Следующие шаги предназначены для авторизациидоверенный_клиентпользователь.

. НажмитеАвторизоватьв верхней части веб-страницы документации OCC REST API.

. ВДоступные разрешенияВ появившемся окне прокрутите вниз до второгоOAuth2.0раздел, затем выберитеБазовая аутентификацияизТип раскрывающийся список.

. ВClientIdВ появившемся поле введитедоверенный_клиент,и вСекретполе, введитесекрет.

. Выберитерасширенныйустановите флажок и нажмитеАвторизовать.

Выполнение вызовов REST в интерактивной документации OCC REST API

Интерактивная документация OCC REST API для коммерческих веб-сервисов позволяет вам опробовать вызовы REST API непосредственно на странице документации. Если вы вносите изменения в витрину магазина, эти изменения отражаются в теле ответа любых вызовов REST API, которые вы делаете.

Контекст

Клиенты OAuth должны быть определены и авторизованы для включения интерактивной документации OCC REST API. Для получения дополнительной информации см.Включение интерактивной документации OCC REST API .

Процедура

. Установите B2C Accelerator.

. Откройте интерактивную документацию OCC REST API в своем веб-браузере, перейдя по следующей ссылке:http://<имя_хоста>:<порт>/rest/v2/swagger-ui.html.

Например, если на вашем локальном компьютере установлена витрина, вы можете получить доступ к документации OCC REST API с помощью следующей команды: связь:https://localhost:9002/rest/v2/swagger-ui.html

. Нажмите на любой контроллер. Например, нажмите наТележкиконтроллер.

Появляются все доступные методы для этого контроллера.

. Нажмите на любой метод. Например, нажмите наПолучатьМетодТележкиконтроллер.

Появляется вся информация, связанная с этим методом.

. BbaseSiteIdполе введите название вашего магазина, напримерэлектроникаилиодежда-uk. . BID пользователяВ поле введите имя пользователя, которого вы зарегистрировали в своем магазине.

Этот пользователь создан в рамках процедурыВключение интерактивной документации OCC REST API .

-Примечание

Не все методы требуют userId. Например, методы дляАкцииконтроллеру требуется только baseId.

. НажмитеПопробуйте!

Выполняется вызов REST, и ответ отображается нижеПопробуйте!кнопка.