



---

Система управления предприятием  
и платформа для цифровизации бизнес-процессов

## **Руководство разработчика**

Global-Marine 2.0 Система управления судостроением и судоремонтом

### **GlobalServerAppGuide**

---

**Управление информационными обменами**

---

# RplDocumentation

Выпуск 0.0.2

мая 28, 2025

## Содержание

<b>1</b>	<b>Концепция интеграции ИС Global и внешних систем</b>	<b>2</b>
1.1	Основные положения интеграции . . . . .	2
1.2	Порядок экспорта из Global . . . . .	4
1.3	Порядок импорта в Global . . . . .	5
1.4	Синхронная модель обменов . . . . .	6
1.5	Мониторинг интеграции . . . . .	6
1.6	Сверка данных после интеграции . . . . .	6
1.7	Примеры типовых сообщений . . . . .	6
<b>2</b>	<b>Система интеграции и репликации</b>	<b>9</b>
2.1	Архитектура классов . . . . .	9
2.2	Настройки контура . . . . .	9
2.3	Методы интеграции . . . . .	11
2.4	Импорт . . . . .	12
2.5	Экспорт . . . . .	14
2.6	Схемы загрузки/выгрузки данных . . . . .	18
2.7	Запуск интеграции . . . . .	20
2.8	Монитор интеграции . . . . .	20
2.9	Тестирование контура . . . . .	20
2.10	Логи импорта . . . . .	20
2.11	Перечень настроек интеграции . . . . .	22
2.12	Примеры сообщений при работе со стандартным пакетным протоколом . . . . .	23
<b>3</b>	<b>Синхронная интеграция через <i>REST</i></b>	<b>28</b>
3.1	Реализация экспорта через REST . . . . .	28
3.2	Реализация импорта через REST . . . . .	31
<b>4</b>	<b>Прикладные сервисы</b>	<b>33</b>
4.1	Сервис открытия карточки объекта по параметрам . . . . .	33
4.2	Сервис открытия карточки объекта по GID (глобальному идентификатору) . . . . .	34

---

# 1 Концепция интеграции ИС Global и внешних систем

## 1.1 Основные положения интеграции

1. Рекомендуется к использованию зеркальный режим обменов, когда все нижеперечисленное реализуется системой Global и внешней системой. Global загружает измененные данные во внешнюю систему, внешняя система загружает изменения в Global (режим 1). Возможны другие варианты, когда Global выгружает все изменения во внешнюю систему и Global опрашивает внешнюю систему для получения изменившихся данных из неё (режим 2), или наоборот, когда вся интеграционная активность выполняется на стороне внешней системы (режим 3).
2. Экспорт работает по протоколу HTTP(HTTPS) методом POST на адрес осуществляющий прием сообщений. В режиме 1 каждая система публикует минимум один сервис для приема сообщений, при необходимости сервисов на прием может быть больше, для разных систем или разделения потоков данных.
3. При очень высокой нагрузке на интеграционные сервисы или высокие требования к скорости загрузки/выгрузки данных, сервисы необходимо переводить в адаптеры с постоянным подключением к интегрируемым системам (веб сокет, например).
4. Формат сообщений - XML.
5. В рамках одного сеанса обменов один объект – одно сообщение. Ответ на сообщение должен идти с указанием идентификатора сообщения.
6. С целью ускорения сетевых обменов сообщения могут группироваться в пакеты, размер пакета настраивается в системе-источнике. Компрессии нет. Пакеты не имеют индивидуальных идентификаторов, т.к. все обмены идут в привязке к идентификатору сообщения.
7. Система-приемник гарантирует успешную отправку ответа на каждое полученное сообщение.
8. Интеграция как на импорт, так и на экспорт осуществляется сеансами.
9. Для каждого контура интеграции, включающего в себя настройку узла-подписчика и узла публикации, ведется собственная последовательность сеансов.
10. Для каждого контура интеграции настраивается перечень сущностей с подчинёнными сущностями (далее бизнес-объект или БО) с перечнем атрибутов, изменение которых должно приводить к выгрузке объекта по настраиваемому контуру.
11. Для каждого БО в рамках контура обменов настраивается фильтрация объектов, определяющая требуется выгрузка объекта по этому контуру или не требуется.
12. В логах системы фиксируется информация об измененных объектах и атрибутах, которые привели объект к подписке на выгрузку.
13. Обмен выполняется бизнес-объектами целиком. Изменение атрибутов классов, составляющих бизнес-объект, инициирует передачу бизнес-объекта во внешние системы. Не отслеживаются изменения атрибутов, данные которых не участвуют в обменах. При приеме данных по-умолчанию выполняется обновление всех полей главной сущности, данные подчиненных сущностей заменяются данными из сообщения.
14. Подтверждение приема или информация об ошибке приема каждого объекта должно идти отдельным сообщением. Подтверждение нескольких объектов одним сообщением не поддерживается.
15. Экспорт каждого объекта из Global может осуществляться для нескольких узлов-подписчиков. В результате каждому объекту в Global потенциально может быть сопоставлено несколько объектов во внешних системах (каждый со своим GUID), один для каждого контура обмена.

16. События импорта и экспорта объектов логируются в привязке к идентификаторам сообщений и идентификаторам объектов обменов.
17. Система-приемник загружает данные из очереди в определенной последовательности, соответствующей логической связанности объектов интеграции (сначала справочники, потом документы). Последовательность формируется системой-источником путем нумерации сообщений для сортировки их на приемной стороне.
18. Одна из систем хранит сопоставление идентификаторов, она же генерирует исходящее сообщение в идентификаторах другой системы. Другая система генерирует сообщения в своих идентификаторах.
19. По умолчанию сопоставления объектов выполняются по идентификаторам, возможен вариант поиска по ключевым полям.
20. Интегрируемые объекты запрещены к физическому удалению. Запрет устанавливается в процессе отправки объекта в любую ИС.
21. Блокировка редактирования отправленных объектов по умолчанию не устанавливается.
22. При работе каждого сеанса экспорта обрабатывается очередь изменившихся объектов, добавляются объекты, требующие повторной отправки, ответы на импортированные сообщения, формируются пакетные сообщения для внешней системы, отправляются в очередь импорта внешней системы.
23. При работе каждого сеанса импорта обрабатывается очередь загруженных внешней системой объектов, формируются ответные сообщения, отправляются в очередь импорта внешней системы.
24. Асинхронная модель обменов в режиме 1 на примере экспорта:
  - Отправка сообщения (пакета), получение ответа об успешной доставке.
    - Если ответ об успешной доставке не получен, сообщение (пакет) остается в очереди на отправку, фиксируется ошибка в логе сессии.
  - Обработка сообщения приемной стороной, формирование ответа.
    - Любые ошибки на этапе обработки сообщения на приеме логируются приемной стороной и передаются в ответном сообщении. При критической ошибке сообщение должно оставаться в очереди на импорт.
  - Получение ответа на сообщение, фиксация подтверждения приема.
    - Если получен неуспешный результат загрузки, объект, который был экспортирован, снова помещается в очередь экспорта.
    - Количество попыток экспорта ограничено, после достижения лимита объект снимается с экспорта, пишется в отдельный журнал требующий ручной обработки.
25. Синхронная модель обменов в режиме 1 на примере экспорта:
  - отправка сообщения
  - обработка сообщения приемной стороной
  - получение ответного сообщения. Любые ошибки выдаются пользователю на экран. Результат выгрузки полученный в ответном сообщении выдается пользователю на экран.

## 1.2 Порядок экспорта из Global

Рассмотрение осуществляется на примере некоторого бизнес-объекта, например, справочник номенклатуры.

1. Пользователь изменяет данные в карточке номенклатуры или любой табличной части, входящей в состав бизнес-объекта.
  - На основе выполненных настроек БО происходит анализ выгружаются ли изменяемые данные в контуры интеграции.
  - В логовые таблицы попадает информация об измененном объекте, измененных полях в привязке ко всем контурам экспорта, на которые измененные данные должны выгружаться.
2. Работающий асинхронно по расписанию агент снэпшотов анализирует логовую таблицу, по каждой строке лога выполняется фильтрация объекта. Пример фильтра – номенклатура должна быть действующей. Лог очищается.
  - Если объект удовлетворяет фильтру, то ему генерируется xml представление (снэпшот), настроенное на контуре, указанном в логе. Сформированное представление помещается в очередь на экспорт.
  - Если объект фильтру не удовлетворяет, то:
    - Если объект ранее успешно выгружался по этому контуру или выгружался без полученного подтверждения, то объекту генерируется xml представление с пометкой удаления и помещается в очередь на экспорт.
    - Иначе объект не выгружается по данному контуру.
3. Работающий асинхронно по расписанию агент экспорта из Global регистрирует новый сеанс интеграции и осуществляет формирование итогового пакета данных из снэпшотов, имеющихся на текущий момент времени.
  - Источники данных для очереди экспорта:
    - Список измененных объектов.
    - Объекты, не отправленные в прошлых сеансах из-за транспортных ошибок.
    - Объекты, отправленные в прошлых сеансах, но с полученным ошибочным результатом импорта, не выбравшие количество попыток выгрузки (например, нарушен порядок загрузки объектов и в результате нарушена ссылочность, объект может быть загружен успешно со 2го или 3го раза). При достижении лимита выгрузок объект перемещается в отдельный журнал для ручной обработки.
    - Ответы на входящие сообщения.
  - Свертка списка измененных объектов до объекта (если объект был помещен в очередь несколько раз).
  - Сортировка объектов экспорта, если настроен особый порядок выгрузки статей интеграции (например, сначала справочники, потом документы). По умолчанию сортировка выполняется в порядке размещения объектов в очереди, в большинстве случаев это позволяет не нарушать ссылочность при импорте.
  - Срез очереди по ограничениям объема экспортируемых данных, если лимиты настроены на контуре. Может измеряться в кол-ве сообщений или размере сообщений.
  - Вне лимитов отправляются ответы на импортированные сообщения. Помещаются в начало очереди. Объекты, не отправленные в прошлых сеансах из-за транспортных ошибок, помещаются в начало очереди. Далее идут измененные объекты в порядке, определенном в пункте

выше. В конец помещаются ошибочные объекты из предыдущих сессий экспорта (только те, которые не присутствуют в списке измененных).

- Генерируется xml представление (снэпшот) по каждому объекту, если не было сделано ранее.
  - Формирование сообщений с уникальными идентификаторами, указанием вложенного класса сообщений, номером сообщения (для сортировки). Номер автоинкрементный в рамках контура обменов.
  - Упаковка сообщений в пакеты. Размер пакета в количестве сообщений настраивается на контуре.
  - Отправка пакетов. Если не получено успешное подтверждение доставки, данные пакета из очереди не удаляются. Фиксируется ошибка в логах сессии. Сессия прерывается с ошибкой.
  - Сессия экспорта завершается
4. При наличии ошибок в интеграции, Global осуществляет оповещения по имеющимся настройкам оповещений.

### 1.3 Порядок импорта в Global

Рассмотрение осуществляется на примере некоторого бизнес-объекта, например, справочник номенклатуры.

1. Работающий онлайн агент импорта в Global принимает входящие сообщения, выполняет распаковку, проверку формата сообщения, размещает сообщения в очереди импорта, выдает подтверждающие ответы на все сообщения.
  - В случае возникновения ошибки в ответ возвращается код и текст ошибки, сообщение считается не принятым.
2. Асинхронно работающий по расписанию агент обработки входящих сообщений создает новую сессию импорта, обработка очереди импорта.
  - Сортировка очереди по номерам сообщений.
  - Разбор сообщений, выполнение действий.
  - Для ответных сообщений происходит поиск исходного экспортного сообщения по указанному идентификатору и установка результата импорта с указанным комментарием. При положительном результате сообщение считается доставленным и более не выгружается. При отрицательном результате выполняется проверка на достижение лимита выгрузок, если пройдена, то объект будет выгружен в ближайшей сессии экспорта.
  - Особые ситуации с обработкой ответов на ошибочные события при импорте, требующие снятия объекта с экспорта обрабатываются стороной возвращающий ответ. Например, если документ проведен в системе-приемнике, и осуществляется попытка его импортировать, то должен возвращаться отрицательный ответ с указанием комментария соответствующего ситуации, но установленным особым признаком, запрещающим дальнейшую подписку объекта на экспорт. Например, комментарий может быть такой: «Документ не может быть перезаписан, проведен в БУ».
  - Создание или обновление объектов системы по присланным сообщениям.
  - Формирование ответов с положительным или отрицательным результатом импорта, размещение их в очереди экспорта.
  - Завершение сессии импорта.

## 1.4 Синхронная модель обменов

Рассмотрение осуществляется на примере некоторого бизнес-объекта, например, справочник номенклатуры.

1. Пользователь вызывает событие экспорта объекта в выбранном контуре обменов с внешней ИС (например, кнопкой в карточке объекта).
2. Объекту генерируется xml представление (снэпшот) согласно настройкам сущностей на выбранном контуре обменов.
3. Формируется сообщение, упаковывается в транспортный пакет.
4. Выполняется обращение к REST сервису удаленной ИС, загрузка пакета.
5. Ожидание ответного сообщения (пакета).
6. Получение ответного сообщения (пакета), разбор результата, демонстрация результата пользователю.
7. В случае возникновения любых ошибок демонстрация их пользователю.

## 1.5 Мониторинг интеграции

1. Система предоставляет интерфейс анализа сеансов интеграции (публикации данных для экспорта, приема данных при импорте).
2. Система позволяет отобразить и проанализировать объекты, при интеграции которых возникли проблемы.
3. Имеется возможность настроить оповещения о проблемах в ходе интеграции.

## 1.6 Сверка данных после интеграции

1. С настраиваемой периодичностью должна осуществляться сверка данных за период времени. Отбор по периоду осуществляется по некоторой дате бизнес-объекта (свой атрибут для каждого БО). Сверка осуществляется по агрегированным показателям (количество, сумма по некоторым атрибутам). Может быть настроена под конкретный бизнес-объект. Результат анализа, осуществленный со стороны одной системы, выгружается особым сообщением в другую систему, которая загружает данные в отдельную структуру хранения.
2. Предоставляется интерфейс, позволяющий проанализировать результаты сверки и расхождения.

## 1.7 Примеры типовых сообщений

Пример исходящего/входящего сообщения, в пакете одно сообщение. Поддерживается пакетный и одиночный режим.

```
<MessagePack>
  <!--Класс объектов в пакете. можно не указывать и не заполнять если поиск класса_
  ↳разрешается дальше в методе обработки сообщений-->
  <ObjectClass>Bs_ExpenceItem2</ObjectClass>
  <!--Системное имя контура интеграции, в который отправляется сообщение. Можно не_
  ↳указывать, если контур передается иным способом. Например сразу указывается в адресе_
  ↳отправки сообщения.-->
  <Circuit>GNSI001_FR</Circuit>
```

(continues on next page)

```

<!--В пакете может быть несколько вложенных сообщений Message или Response-->
<Message>
  <!--ID сообщения для указания в ответе-->
  <MessageID>FBCFD1F337287731E0530E02A8C07C59</MessageID>
  <!--Тип сообщения, object означает что в тегах будет описание объекта-->
  <MessageType>object</MessageType>
  <!--Порядковый номер сообщения для сортировки на приемной стороне-->
  <MessageOrder>538</MessageOrder>
  <!--дата создания сообщения, для аналитики или сортировки -->
  <CreateDate>2023-05-16T15:45:12</CreateDate>
  <!--система источник сообщения-->
  <SystemSource>Global</SystemSource>
  <!--Блок с данными, имя вложенного тега определяет класс и тип документа, внутри
  ↳ полная информация и документе, статусе удаления и проведения-->
  <Data>
    <!--вложенный документ или справочник, пример ниже взят из контура обмена с 1С-->
    <Документ.ПередачаМатериаловВЭксплуатацию>
      <КлючевыеСвойства>
        <Ссылка>DD4E21FF-FD70-7440-AD43-A8A00342DAA5</Ссылка>
        <Дата>2023-05-12T07:55:16</Дата>
        <Номер>1170</Номер>
        <Организация>
          <Ссылка>4dd04906-208a-11dc-b2d3-0015170eef56</Ссылка>
          <ИНН>4706020609</ИНН>
          <КПП>470601001</КПП>
          <ЮридическоеФизическоеЛицо>ЮридическоеЛицо</ЮридическоеФизическоеЛицо>
        </Организация>
      </КлючевыеСвойства>
      <Ответственный>
        <Наименование>Тверская Оксана Валерьевна</Наименование>
        <ФизическоеЛицо>
          <Ссылка>6c5f5603-5ecc-11e2-b53e-001e671031a0</Ссылка>
          <ФИО>Тверская Оксана Валерьевна</ФИО>
          <Фамилия>Тверская</Фамилия>
          <Имя>Оксана</Имя>
          <Отчество>Валерьевна</Отчество>
          <ДатаРождения>1966-04-16</ДатаРождения>
          <КодВПрограмме>58850170001</КодВПрограмме>
        </ФизическоеЛицо>
      </Ответственный>
      <Товары>
        <Строка>
          <ДанныеНоменклатуры>
            <Номенклатура>
              <Ссылка>bf630b5e-1f36-11dc-b2d3-0015170eef56</Ссылка>
              <НаименованиеПолное>Костюм ИТР (3069, ШТ)</НаименованиеПолное>
              <КодВПрограмме>3069</КодВПрограмме>
              <Наименование>Костюм ИТР (3069, ШТ)</Наименование>
            </Номенклатура>
          </ДанныеНоменклатуры>
          <ЕдиницаИзмерения>
            <Ссылка>38DDB753-4475-FB4B-E053-0F02A8C07E1B</Ссылка>

```

(continues on next page)

```

        <Код>ШТ</Код>
        <Наименование>Штука</Наименование>
    </ЕдиницаИзмерения>
    <Количество>1.000</Количество>
    <КоличествоУпаковок>1.000</КоличествоУпаковок>
    <ТипЗапасов>СобственныеТовары</ТипЗапасов>
    <ФизическоеЛицо>
        <Ссылка>1a5a8fa3-efcd-11ed-8113-d8d385e18888</Ссылка>
        <ФИО>Чупраков Андрей Павлович</ФИО>
        <Фамилия>Чупраков</Фамилия>
        <Имя>Андрей</Имя>
        <Отчество>Павлович</Отчество>
        <ДатаРождения>1990-04-03</ДатаРождения>
        <КодВПрограмме>767665540001</КодВПрограмме>
        <ИНН>290408981899</ИНН>
    </ФизическоеЛицо>
    <СрокЭксплуатации>24</СрокЭксплуатации>
</Срока>
</Товары>
</Документ.ПередачаМатериаловВЭксплуатацию>
</Data>
</Message>
</MessagePack>

```

Пример входящего сообщения, в пакете одно сообщение с ответом на исходящее.

```

<MessagePack>
    <!--В пакете может быть несколько вложенных сообщений Message или Response-->
    <Response>
        <!--ID сообщения на который пришел ответ-->
        <MessageID>FBCFD1F337287731E0530E02A8C07C59</MessageID>
        <!--ID ответа, напрямую не используется, для анализа ошибок-->
        <ResponseID>FBCFD1F337297731E0530E02A8C07C59</ResponseID>
        <!--дата создания сообщения, для аналитики или сортировки -->
        <CreateDate>2023-05-16T15:46:12</CreateDate>
        <!--ID Объекта созданного или измененного в рамках сообщения, для сохранения в
        ↳таблицу сопоставления идентификаторов-->
        <ObjectID>D7965B84-53AC-4442-ACB7-77CC1D4A6983</ObjectID>
        <!--Результат приема сообщения-->
        <Result>false</Result>
        <!--Код результата приема сообщения-->
        <ResultCode>0</ResultCode>
        <!--Комментарий для пользователя, может быть указан как для положительных, так и для
        ↳отрицательных результатов-->
        <Description>Нет достаточного количества на складе «Оптовый»</Description>
        <!--признак снятия объекта с экспорта, по умолчанию false, true передавать в особы
        ↳ситуациях-->
        <NoPublish>true</NoPublish>
    </Response>
</MessagePack>

```

## 2 Система интеграции и репликации

### 2.1 Архитектура классов

**Btk\_ExtSystem** - класс внешней системы.

**Rpl\_Circuit** - контур интеграции. Описывает интерфейс взаимодействия с внешней системой. Можно настроить импорт, экспорт или оба процесса сразу.

**Rpl\_IntXmlInMsg** и **Rpl\_IntXmlState** - сообщения для импорта и экспорта, прикреплены к контуру. Сообщения остаются в системе как лог после обработки.

**Rpl\_IntInSession/Rpl\_IntOutSession** - сессия интеграции. Хранит историю одной попытки импортировать или экспортировать объекты. К ней прикреплены обработанные сообщения и логи.

Новые сообщения, ждущие обработки, не прикреплены к сессии и имеют **idpIntSession === -1**.

### 2.2 Настройки контура

Контур обеспечивает связь **Global** с внешней системой. Эту связь обеспечивают несколько методов - обработки, создания сообщений и так далее.

Большинство настроек хранится в классе-спутнике **Rpl\_IntegrationCircuitExt**. В нём есть поле **sProps** с JSON объектом, в котором хранятся названия методов интеграции и прочие настройки.

Эти настройки можно отредактировать во вкладке «Свойства» раздела «Настройка каналов интеграции»:

Параметры контура	Параметры интеграции	Ответ на входящее сообщение({id})	Бизнес объекты	Типы объектов, требующие Акцепт
Тип интеграции	Репликация			
Тип обмена данными по интеграции	Xml Rpl			
Внешняя система	G1			
Работа контура по новой схеме(через журнал)	<input type="checkbox"/>			
Функция генерации XML	Rpl_GenerateMessagePkg().genXmlStub			
Функция загрузки XML				
Свойства	<pre>{   "sMetaSchema": "88",   "sProcessIncomingMessageMethod": "Rpl_ProcessIncomingMessagePkg.processIncomingByBtsAndCreateAnswer",   "sSendMessageMethod": "Rpl_SendWebRequestPkg().sendPackageRequest",   "sWebAddress": "http://192.168.24.101:8080/httpodb/",   "nPackageCount": "1",   "requestTimeoutMs": "10000",   "bExport": "1" }</pre>			
Функция интеграции	Rpl_RunIntegrationPkg.run("G1/G4")			
Функция подтверждения импорта				
Хранить лог сессий (дней)				
Проводить диагностику	<input type="checkbox"/>			
Функция диагностики				
Хранить лог самодиагностики (дней)				

Напоминания [x]
Монитор интеграции [x]
**Контур** [x]

[illegible]

10

## Параметры выполнения интеграции

Для увеличения возможности параметризации процесса интеграции введены параметры выполнения. Это переменные контекста к которым есть доступ во время выполнения запуска интеграции. Организованы эти переменные как `executionParameters: Map[String, String]`. Инициализация происходит в момент запуска контура интеграции. Доступ возможен через метод `ru.bitec.app.rpl.channel.intg.Rpl_RunIntegrationPkg.getExecutionParameters`

## Источники

Имеется два источника параметров по убыванию приоритета:

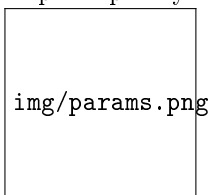
1. Параметры переданные с обращением через REST для запуска контура (schedule). Должны передаваться в теле запроса в формате XML следующим форматом:

```
<Rpl_ExecutionParameters>
  <ExecutionParameter>
    <Code>testFromRest</Code>
    <Value>var r = "13"; r;</Value>
    <isJexl>1</isJexl>
  </ExecutionParameter>
</Rpl_ExecutionParameters>
```

Где:

- Code - имя переменной(на eng раскладке)
- isJexl - определяет является ли поле Value вычисляемым через jexl(должен вернуть String) или константой
- Value - значение

1. Параметры указанные на закладке «Параметры выполнения» в настройке контура



Атрибуты соответствуют Code, Value, isJexl из предыдущего пункта с теми же требованиями. Добавлено поле описание. Это комментарий для понимания, он не влияет на работу с переменной.

## 2.3 Методы интеграции

### Алгоритм интеграции:

Порядок вызова методов интеграции:

- Импорт (`Rpl_IntXmlInMsg`), если включён (`"bImport": "1"`)
  1. (опционально) Подготовка сообщений - метод `sGetMessageForImport`. Если метод не указан, шаг пропускается.
  2. Создание сессии интеграции. Сообщения из очереди на импорт прикрепляются к сессии.

3. Обработка - метод импорта `sProcessIncomingMessageMethod` применяется к каждому входящему сообщению.
- Экспорт (`Rpl_IntXmlState`), если включён ("`bExport`": "1")
  1. Генерация содержимого сообщения - метод `sGenerateMessageMethod`
  2. Создание сессии интеграции. Сообщения из очереди на экспорт прикрепляются к сессии.
  3. Обработка - метод экспорта `sSendMessageMethod` применяется к каждому исходящему сообщению.
  4. (опционально) Если указан, к обработанным сообщениям экспорта применяется метод в `sAfterSendMethod`

Ключ в свойствах контура	Сигнатура метода
<code>sGetMessageForImport</code>	<code>(ropCircuit: Rpl_CircuitApi#ApiRop): Unit</code>
<code>sProcessIncomingMessageMethod</code>	<code>(ropMessage: Rpl_IntXmlInMsgApi#ApiRop): NGid</code>
<code>sGenerateMessageMethod</code>	<code>(ropXmlState: Rpl_IntXmlStateApi#ApiRop): Unit</code>
<code>sSendMessageMethod</code>	<code>(ropXmlState: Rpl_IntXmlStateApi#ApiRop, responseBody: String, responseCode: Integer): SendResult</code>
<code>sAfterSendMethod</code>	<code>(List[Rpl_IntXmlStateApi#ApiRop]): Unit</code>

Методы `ProcessIncomingMessage` и `AfterSend` обрабатывают сразу несколько сообщений в списке.

## 2.4 Импорт

### Получение данных

Сообщения импорта `Rpl_IntXmlInMsg` попадают в очередь контура двумя способами:

1. Сообщения записывают в очередь на импорт по мере получения
2. Данные получают при запуске интеграции, в методе, указанном в `sGetMessageForImport`

Разница между подходами в том, когда именно `Global` связывается с внешней системой. В первом случае это происходит вне сессии интеграции. Сообщения накапливаются в очереди, а затем обрабатываются. Во втором случае данные получают уже во время сессии интеграции.

Метод помещает данные в `Rpl_IntXmlInMsg`, где сообщение хранится в виде файла. Важно указывать класс `Global`, к которому принадлежит объект.

### Получение файлов из хранилища

В `Rpl_IntXmlImportApiPkg` есть несколько функций, которые читают файлы из доступных `Global` директорий, создают для них сообщения и помещают в очередь импорта. Директория указывается в настройках контура по ключу "`sFolderPath`" с возможностью указать "`sFileStorage`".

## Получение данных через REST

В пакете `Rpl_RequestHandlerPkg` настроен REST интерфейс, который принимает данные и записывает их в виде входящего сообщения в указанный контур. Он принимает сообщения на `/app/sys/rest/ss/pkg/Rpl_RequestHandlerPkg/sendMessage`

## Генерация ответов

Для отправки ответа после импорта объекта добавьте `Rpl_IntXmlState` с ответным сообщением в очередь экспорта. Так как экспорт происходит после импорта, ответное сообщение отправится в ту же сессию интеграции. Установите `bIsAnswer = 1`, чтобы отличить ответ от подписанного на экспорт БО.

## Пример обработки сообщения

```
lazy val idvSystem = Btk_ExtSystemApi().findByMnemonicCode("example_system")
lazy val idvCircuit = Rpl_CircuitApi().findByMnemonicCode("example_circuit")

def processMessage(rpMessage: Rpl_IntXmlInMsgApi#ApiRop): NGid = {
  val xmlMessage = Utility.trim(Btk_XmlBuildPkg().loadXmlUpperCased(
    Rpl_IntXmlInMsgApi().getStringFromFileByUtf8(rpMessage)
  ))

  // Нам передали id элемента во внешней системе. Ищем, есть ли уже такой в Global
  val idvExisting = Cur_CurrencyApi().findByIdExtSystemIdExt(idvSystem, (xmlMessage \ "ID"
  →).text)

  val rvCur =
    if(idvExisting.isNotNull) {
      Cur_CurrencyApi().load(idvExisting)
    } else{
      val r = Cur_CurrencyApi().insert()
      Cur_CurrencyApi().setIdExtForExtSystem(r, idvSystem, (xmlMessage \ "ID").text)
      r
    }

  Cur_CurrencyApi().setsCodeOKV(rvCur, (xmlMessage \ "OKV").text)
  Cur_CurrencyApi().setsCaption(rvCur, (xmlMessage \ "CAPTION").text)

  val answer =
    <RESULT>
      <STATUS>OK</STATUS>
    </RESULT>

  // Ответное сообщение об успешном импорте
  Rpl_IntXmlStateApi().register(
    idpCircuit = idvCircuit,
    spXml = Btk_XmlBuildPkg().getClob(answer, addProlog = true),
    bpIsAnswer = 1.nn,
    gidObj = None.toGid
  )
}
```

(continues on next page)

```

rvCur.gid
}

```

Вместо того чтобы писать парсер, можно было создать схему загрузки.

## 2.5 Экспорт

### Структура экспортного сообщения

Экспортное сообщение - это либо подписанный объект, либо ответ. Оно называется `Rpl_IntXmlState`, потому что на раннем этапе разработки предполагалось, что в нём содержится XML для экспорта. На самом деле он может содержать данные любого формата.

Экспортное сообщение бывает двух видов - подписанный объект и ответ.

**Подписанный объект** обозначает БО, который нужно экспортировать. В сообщении указан его `Gid` и данные для экспорта. Обычно эти данные - содержимое БО в формате XML. Поэтому функция отправки сообщения не обращается к базе данных - все нужные данные содержатся в отправляемом `Rpl_IntXmlState`.

Данные в экспортном сообщении либо записываются в момент его создания, либо генерируются на этапе экспорта

функцией в `sGenerateMessageMethod`, которая получает `Rpl_IntXmlState` и заполняет их данные. Обычно это работает так: когда нужно экспортировать БО, его «подписывают» на экспорт, то есть добавляют в очередь экспорта сообщение с `Gid` этого объекта. `sGenerateMessageMethod` получает данные объекта с этим `Gid` (например, его выгрузку в XML) и записывает в то же сообщение. После этого функция отправки читает эти данные и шлёт внешней системе.

**Ответ** обозначается флагом `bIsAnswer = 1`. Это сообщение, не связанное напрямую с БО, поэтому `Gid` можно не указывать. Такие сообщения экспортируют в ответ на информационный запрос и их данные обычно заполняются при создании.

### Контроль изменений

Система фиксации изменений в `Global` отслеживает изменения объектов и отправляет их на экспорт. В настройках контура на вкладке “Бизнес объекты” укажите, изменения каких объектов фиксировать и какие поля учитывать:

Системное имя	Наименование	Тип контура	Тип объекта
MDM	Контур обмена с MDM	Xml-интеграция	Контур интеграции (совместимость)
dMDM	Контур обмена с dMDM	Xml-интеграция	Контур интеграции (совместимость)
SAP_ERP	Контур обмена с SAP	Xml-интеграция	Контур интеграции (совместимость)
SAP_RFID	Сканер меток	Xml-интеграция	Контур интеграции (совместимость)
CUR	Загрузка курсов валют (SAP PO)	Xml-интеграция	Контур интеграции (совместимость)
EGRUL	Выписки из EGRUL	Xml-интеграция	Контур интеграции (совместимость)
		Xml-интеграция	Контур интеграции (совместимость)
EDO	Контур обмена с EDO	Xml-интеграция	Контур интеграции (совместимость)
Galaktika	Контур обмена с Galaktika	Xml-интеграция	Контур интеграции (совместимость)
IDM	Прием запросов от IDM	Xml-интеграция	Контур интеграции (совместимость)
SAP_IPC	Контур обмена с SAP через шину	Xml-интеграция	Контур интеграции (совместимость)
SAP_ASF	Контур синхронизации инвентарных объектов	Xml-интеграция	Контур интеграции (совместимость)
DRX_CREATE_WORK_GROUP	Интеграция с Directum RX (Справочник груп...	Xml-интеграция	Контур интеграции
DRX_CREATE_GRAPH_PSDKR	Интеграция с Directum RX (Справочник ...	Xml-интеграция	Контур интеграции
DRX_CREATE_DESIGN_OBJ_STATUS	Интеграция с Directum RX (Справочник ...	Xml-интеграция	Контур интеграции
DRX_CREATE_SET_OF_DOC	Интеграция с Directum RX (Справочник ...	Xml-интеграция	Контур интеграции
DRX_CREATE_KRPO_STRUCT_PLAN	Интеграция с Directum RX (Структура ПКР)	Xml-интеграция	Контур интеграции
DRX_CREATE_INV_NUM_KR	Интеграция с Directum RX (Объекты ПКР ПО)	Xml-интеграция	Контур интеграции
DRX_CREATE_DESIGN_OBJECT	Интеграция с Directum RX (Объекты ...	Xml-интеграция	Контур интеграции
DRX_CREATE_DESIGN_OBJECT_ESTIMATE	Интеграция с Directum RX (Объекты ...	Xml-интеграция	Контур интеграции

Параметры контура	Параметры интеграции	Ответ на входящее сообщение({id})	Бизнес объекты	Типы объектов, требующие Акцепт
Класс-шапка бизнес объекта	Класс-шапка бизнес объекта	Условие фильтрации	Условие фильтрации PostData	Очередность экспор...
Pm_PayMove	Взаиморасчет	exists ( select 1 from pm_paymove f join ...		1
Bdg_Forecast	Бизнес прогноз	(t.idStateMC = '300' and ...		5
Cnt_Contract	Договоры	(coalesce(cast(t.jobattrs_dz -...		2
Pm_PayRequest	Заявка на платеж	(t.idStateMC = '350' and ...	(t.sAttrName = 'idStateMC' and ...	1
Prs_PurchReq	Заявка на закупку	(exists(select 1 from prs_purchreqver where ...		3
Prs_Request	Заявка на потребность	(coalesce(cast(t.jobattrs_dz -...		6
Stm_ActIn	Приходная накладная/акт/возврат	(t.idStateMC >= '300' and ...		7
Act_TransDoc	Хозяйственная операция	exists (select 1 from stm_actin a ...		4

Состав
Класс Класс
Cnt_Contract Договоры

Атрибуты
Атрибут Логируется Сохранять старое значение
id <input type="checkbox"/> <input type="checkbox"/>
bCalcSumByChild <input type="checkbox"/> <input type="checkbox"/>
bDirectDelivery <input type="checkbox"/> <input type="checkbox"/>
bIsGovernmentContract <input type="checkbox"/> <input type="checkbox"/>
bManInput <input type="checkbox"/> <input type="checkbox"/>
bMultiCurrency <input type="checkbox"/> <input type="checkbox"/>
bMultiVAT <input type="checkbox"/> <input type="checkbox"/>
bManativeCum <input type="checkbox"/> <input type="checkbox"/>

Система захватывает изменения в момент фиксации транзакции БД и отправляет в журнал изменившихся объектов. В менеджере заданий работает задача «Агент Xml-Snapshot», которая подписывает изменённые объекты на экспорт. Для каждого изменённого объекта создаётся `Rpl_IntXmlState` с Gid БО.

Имеется возможность добавить в отслеживание по БО класс, который не входит в данный БО. Для этого на закладке «Состав бизнес объекта контура» воспользуйтесь операцией «Добавить класс, не принадлежащий БО». Затем заполните поле «Атрибут для связи с классом-предком» (это определяет по какому значению из зависимого класса будет производиться запрос к шапке БО). Отредактируйте «Запрос для связи с классом-предком». Верхняя его часть с конструкцией `with` не должна изменяться.

Значение `saLinkAttrValue` это сохранённое строковое (при необходимости приведите к нужному типу в запросе) значение «Атрибута для связи с классом-предком», на котором будет основываться ваш поиск шапки БО. Пример: Имеется зависимый класс `Rpl_IntegrationCircuitExt` и класс шапка БО `rpl_circuit`. Они не связаны в единый БО, но `Rpl_IntegrationCircuitExt` имеет ссылку `idCircuit` на `rpl_circuit`. В данном случае настройка будет выглядеть так:

"Атрибут для связи с классом-предком" - `idCircuit`.

"Запрос для связи с классом-предком":

```

    ...
    with originalObjectValues as (select idMarkedLog, sLinkAttrValue from unnest({idaLog}
→, {saLinkAttrValue})) as t(idMarkedLog, sLinkAttrValue))
    select s.idMarkedLog, t.gid as gidForSwap
    from originalObjectValues s
    join rpl_circuit t on t.id = s.sLinkAttrValue :: int8
    ...

```

Если вы неверно настроили, то при попытке подписания и обработки снапшотером в лог запишется текущая ошибка. Посмотреть её можно в мониторе интеграции. Закладка для сессий экспорта «Ошибки подписки на контур».

`Rpl_IntXmlMarkedObjLog` - журнал изменений. Можно отмечать объекты как изменённые, добавляя их в журнал. Снапшотер подпишет их на экспорт.

## Генерация сообщений

В Global есть система выгрузки БО в формате XML. Обычно для генерации сообщений используется именно она. Все необходимые функции `Rpl_GenerateMessagePkg`. Этот пакет использует схемы выгрузки `Bts_XmlSchema`.

## Отправка сообщений

Сообщение экспорта содержит `Gid` объекта и данные в формате XML. `sSendMessageMethod` преобразует эти данные в формат для отправки и передаёт во внешнюю систему.

Пример отправки:

```

def sendMessages(ropaIntXmlState: List[Rpl_IntXmlStateApi#ApiRop]): List[SendResult] = {
  for (rvMsg <- ropaIntXmlState) yield {
    val reqPkg = Btk_SendHttpRequestPkg()

    val svRequestXml = Rpl_IntXmlStateApi().getFileAsString(rvMsg)

    val requestParams =
      JObject(Map(
        reqPkg.Keys.method -> reqPkg.methodPost,
        reqPkg.Keys.Content.key -> JObject(Map(
          reqPkg.Keys.Content.sType -> reqPkg.contentTypeXml.sName,
          reqPkg.Keys.Content.sContent -> svRequestXml
        )),
        reqPkg.Keys.url -> "https://example.url"
      ))
  })
}

```

(continues on next page)

```

var result: SendResult = null
reqPkg.withRequest(requestParams) { resp =>
    val code = resp.getStatusLine.getStatusCode
    if (code >= 200 && code < 300) {
        // noPublish определяет, пытаться ли отправить сообщение в следующую сессию
        ↪ интеграции после ошибки.
        // Значение 0 - отправлять заново, 1 - не отправлять
        result = SendResult(
            idXmlState = rvMsg.idJ,
            isSended = 1.nn,
            sResultCode = code,
            sDescription = "Успешно отправлено",
            noPublish = 0.nn
        )
    }
    else {
        result = SendResult(
            idXmlState = rvMsg.idJ,
            isSended = 1.nn,
            sResultCode = code,
            sDescription = s"Не удалось отправить: ${resp.getStatusLine.getReasonPhrase}
        ↪ ",
            noPublish = 0.nn
        )
    }
}
result
}

```

В системе есть готовые методы отправки данных:

- Отправка по HTTP: пакет `Rpl_SendWebRequestPkg`
- Выгрузка сообщений в виде файлов: пакет `Rpl_PutRequestIntoFolderPkg`

## Информация об экспорте объектов

### Формирование `Rpl_IntExpObjInfo`:

Сначала, в момент отправки данных, `Rpl_RunIntegrationPkg.sendMessagePackage` формирует лог отправки — фиксирует успешность, ошибки, описание ответа и другую служебную информацию. Эти данные записываются в буферную таблицу логирования `rpl_intexpobjinfo_log`.

Затем, по расписанию, выполняется `Rpl_IntXmlSnapshotAgentApi.run`, который переносит сформированные записи из логов в агрегированную статистику по выгрузкам объектов с помощью `rpl_intexpobjinfo.updateFromLog`.

Название параметра	Описание, формирования параметров
<b>dFirstSubs</b>	Дата первой подписки. Берется как минимальное из всех значений Даты первой подписки ( <b>dsubs</b> ) из Лога выгрузки объектов ( <b>Rpl_IntExpObjInfoLog</b> ) по объекту и идентификатору контура.
<b>dFirstOut</b>	Дата первой выгрузки. Берется как минимальное из всех значений Даты первой выгрузки ( <b>dout</b> ) из Лога выгрузки объектов ( <b>Rpl_IntExpObjInfoLog</b> ) по объекту и идентификатору контура.
<b>dLastOut</b>	Дата последней выгрузки. Берется как максимальное из всех значений Даты последней выгрузки ( <b>dout</b> ) из Лога выгрузки объектов ( <b>Rpl_IntExpObjInfoLog</b> ) по объекту и идентификатору контура.
<b>dLastSuccessfulOut</b>	Дата последней успешной выгрузки. Берется как максимальное из значений <b>dout</b> , отфильтрованных по условию успешной выгрузки ( <b>bislastoutsuccess = 1</b> ) в таблице <b>rpl_intexpobjinfolog</b> .
<b>sLastError</b>	Текст ошибки. Берется как первое значение из массива <b>sLastError</b> , отсортированного по убыванию даты выгрузки в таблице <b>rpl_intexpobjinfolog</b> .
<b>sLastErrorStack</b>	Стек ошибки. Берется как первое значение из массива <b>sLastErrorStack</b> , отсортированного по убыванию даты выгрузки в таблице <b>rpl_intexpobjinfolog</b> .

## 2.6 Схемы загрузки/выгрузки данных

Global позволяет преобразовывать бизнес-объекты в XML (сериализация) и обратно. Для каждого типа БО нужно настроить схему загрузки/выгрузки **Bts\_Schema**

### Структура классов

- **Bts\_XmlSchema**: коллекция версий схемы
- **Bts\_XmlSchemaVer**: настройки загрузки/выгрузки
- **Bts\_MetaSchema**: стандарт обмена данными - набор **Bts\_XmlSchema** с указанием конкретных версий

## Схемы загрузки (сериализации)

Код	Наименование	Основной класс	Метод загрузки
Nordsy_QJad	Тип корректировки	Требование корректировки	load
Mct_SpKb_OrderSheetEgp_304_	Загрузка ВЗ оборудования - СПКБ	Список документов (Шлюз)	load
QI_FilesFromRedmine_535	Файлы из редмайна	Файлы вопроса	byParent
Mct_GateWayDocListForSPKb_49	Загрузка СП - СПКБ	Список документов (Шлюз)	load
PartPositionInRoot_234	позиция	Состав документа (Шлюз)	byParent
PartAssembly_283	Раздел	Состав документа (Шлюз)	byParent
PartPositionInAssembly_2	позиция в разделе	Состав документа (Шлюз)	byParent
QI_QuestionInRoot_500	Загрузка вопросов - Алмаз (Заказ)	Проекты журнала вопросов	load
Mct_SpKb_OrderSheetMat_43	Загрузка ВЗ материалов - СПКБ	Список документов (Шлюз)	load
Mct_XmlAlmazSP	Загрузка СП - Алмаз	Список документов (Шлюз)	load
QI_QuestionFromRedmine	Вопрос из редмайна	Вопрос	load
QI_FilesFromRedmine_274_1	Файлы из майна	Файлы вопроса	byParent
QI_Messages	Переписка	Обсуждение вопроса	byParent

№	XML-тег	Тип тега	Источник	Имя атрибута	Формат	Значение/Функция	Sql/jexl текст	XML-схема
20	project@name	Нода	SqJ текст - XML	idPrj			with xmldata (data)	
30	subject	Нода	Атрибута класса	sTheme				
40	id	Нода	Атрибута класса	sQuestionNumPror				
50	description	Нода	Атрибута класса	sQuestion				
60	id	Нода	Атрибута класса	sOrder				
70		Нода	Константа	idObjectType		83151		
100		Нода	SqJ текст - XML	sAnswer			with xmldata (data)	
110		Нода	SqJ текст - XML	sActualAuthor			with xmldata (data)	
120		Нода	SqJ текст - XML	sAuthorPhone			with xmldata (data)	
130		Нода	SqJ текст - XML	sSketchNum			with xmldata (data)	
140		Нода	SqJ текст - XML	idPkbCipher			with xmldata (data)	
150		Нода	SqJ текст - XML	idFactoryCipher			with xmldata (data)	
160		Нода	SqJ текст - XML	idVpCipher			with xmldata (data)	

Схема выгрузки описывает, как из бизнес-объекта получить XML. Для каждого XML-тега указано, каким данным он соответствует: это может быть атрибут класса или select/jexl, а также фиксированное значение. На данные для каждого тега можно поставить функции предобработки.

Схемы вложены друг друга. Сгенерированный XML будет повторять эту структуру - XML, сгенерированный для детализаций, вкладывается в XML шапки БО в качестве тега. Так для всего БО генерируется один большой XML.

## Схемы выгрузки (десериализации)

На схеме выгрузки можно указать функции постобработки, чтобы преобразовать данные из XML в тре-

Код	Наименование	Основной класс	Метод загрузки	Примечание	Условие
041_SK_ActWriteOff	041 - ПоступлениеПродукцииИзПереработки	Акт на списание по складу	load		1=2
041-0-bis_department	041-0-Подразделение (СТАТИС)	Подразделение	load		
041-1_IW_Продукция	041-1-IW_Продукция	Позиции акта на списание	byParent		1=2
041-2_IW_ИспользованныеМатериалы	041-2-IW_ИспользованныеМатериалы	Позиции акта на списание	byParent		
041-3_IW_ВозвращенныеМатериалы	041-3-IW_ВозвращенныеМатериалы	Позиции акта на списание	byParent		1=2
023_STK_WarrantOut	023 - ПередачаВПроизводство (PCO)	Расходные складские ордера	load		( exists ( ...
041_SK_ProductionAct	041 - ПоступлениеПродукцииИзПереработки	Поступление из переработки	load		
027_SK_InternalWarrant	027 - ПеремещениеТоваров	Внутренние перемещения	load		(tidStockIn <= tidStockOut or tidMasterIn ...
022_SK_WarrantIn	022 - ВозвратныеОтходы	Приходные складские ордера	load		exists(select 1 ...
QI_MessageRoot	Выгрузка сообщений Книги вопросов	Обсуждение вопроса	load		
022_SK_InternalWarrant	022 - ВозвратныеОтходы	Внутренние перемещения	load		1=2...
031_SK_InternalWarrant	031 - СписаниеТоваров	Внутренние перемещения	load		(tidshortageandiosesstem is not null and ...
002_STM_ActIn_Invoice	002 - Поступление товаров и услуг	Приходная накладная/акт/возврат	load		exists(select 1 from btk_objectType g where ...
028_STM_ActOut_Return	028 - Возврат Товаров Поставщику	Расходная накладная/акт/возврат	load		exists(select 1 from btk_objectType g where ...
XMLfunc_bs_department_Static	XML - Подразделение (Для XML-функций) ...	Подразделение	load		1=2
006_bs_contrast	006 - Контрагенты	Контрагент	load		tidCountry is not null and tidext_dz >> 10.
007_BS_Goods	007 - Номенклатура	ТМЦ	load		
XMLfunc_BS_GovernmentContract	XML - Госконтракт	Госконтракт	load		1=2
XMLfunc_bs_department	XML - Подразделение (Для XML-функций)	Подразделение	load		1=2
004_BS_BankAcc	004 - Банковские счета	Банковские реквизиты	load		
014_Gds_Kind	014 - Виды номенклатуры	Вид номенклатуры	load		
018_BS_DepOwner	018 - Организации	Организация	load		
320_STK_cons	320 - Серии номенклатуры	Партия прихода	load		
318_SK_SeparationAct	318 - Сборка (разборка) товаров (-)	Акт разукрупнения	load		

№	XML-тег	Тип тега	Источник	Имя атрибута	Формат	Значение/Функция	Sql/jexl текст	XML-схема
40	дата	Нода	Атрибута кл					
10	ID	Нода	Атрибута кл					
190	СистемаИсточник	Нода	Константа					
60	Активация	Нода	Константа					
30	Выгрузка	Нода	Константа					

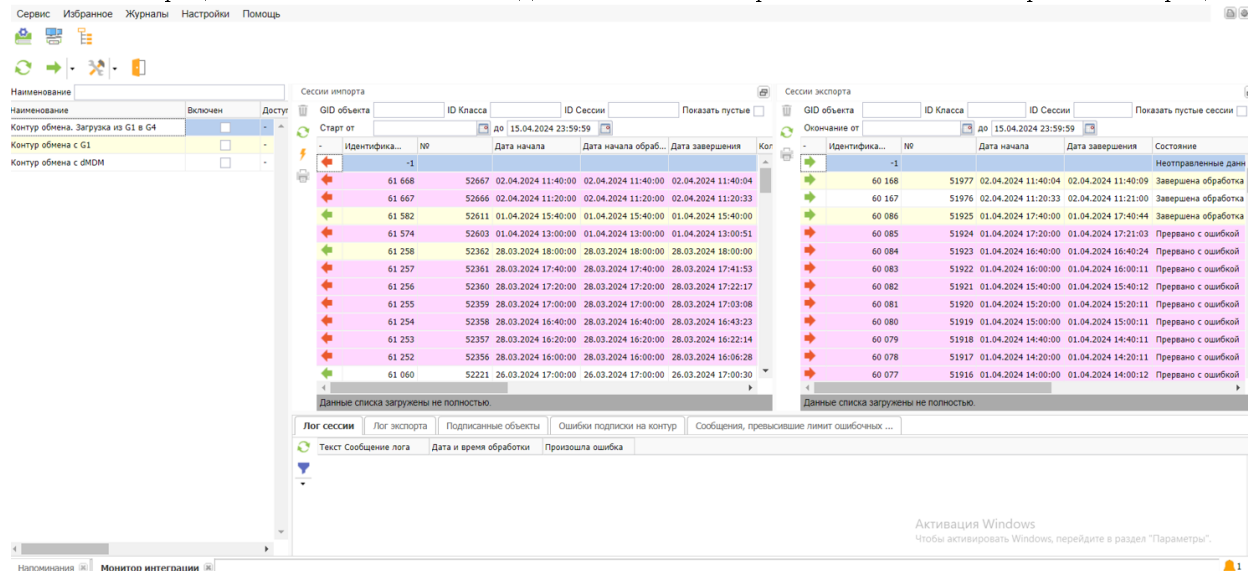
буемый формат.

## 2.7 Запуск интеграции

Интеграция запускается методом `Rpl_RunIntegrationPkg().run("CircuitName")`. В менеджере заданий есть job, который периодически вызывает этот метод, поддерживая актуальность данных.

## 2.8 Монитор интеграции

За интеграцией можно наблюдать в отображении «Монитор интеграции»:



## 2.9 Тестирование контура

Чтобы протестировать контур, нужно добавить сообщение в очередь импорта. Это удобно делать через REST обработчик

```
curl -X POST -H "Content-Type: application/xml" --data @C:\файл\с\сообщением.xml http://localhost:8080/SNG-INTERNAL-DEV/app/sys/rest/ss/pkg/Rpl_RequestHandlerPkg/sendMessageSAP_IPC?access_token=токен_доступа
```

- Токен доступа можно получить в приложении Администратор в настройках пользователя.
- Название контура указывают сразу после `sendMessage`. В этом случае - `sendMessageSAP_IPC`.
- Обязательно укажите базу данных - в этом примере `SNG-INTERNAL-DEV`.

## 2.10 Логи импорта

Можно создавать записи в лог импорта:

```
val idvNewLog = Rpl_IntXmlImportApiPkg().newLogByHead(rpMessage.get(.idCurLog))

/*****

Rpl_IntImportLogApi().setsStackLT(idvNewLog, "Подробности ошибки")
```

Очень важно записывать сообщение, используя логирующую транзакцию (обозначается LT в конце имени метода). Логирующая транзакция гарантирует, что даже если во время импорта произойдёт ошибка и изменения в БД откатятся, лог всё равно будет записан.

## 2.11 Перечень настроек интеграции

Поле cProps	Название в интерфейсе	Описание
sGetMessageFilter	Методов фильтрации сообщений для импорта	
sProcessIncomingMessage	Метод обработки входящего сообщения	
nPackageCount	Кол-во сообщений в пакете отправки	
bImport	Выполнять импорт	
bExport	Выполнять экспорт	
nMaxObjectToExport	Максимальное количество объектов в одной сессии экспорта	
nMaxObjectToImport	Максимальное количество объектов в одной сессии импорта	
sAddAndFilterMessageToExport	Методы добавления и фильтрации объектов на экспорт	Больше не используется
sExportFilterExpression	Выражение для добавки и фильтрации объектов для экспорта	Условие, по которому Rpl_Intxmlstate t попадает в сессию экспорта
sAllExportRequestParameters	Параметры запроса на экспорт	Используется в Rpl_ExportRequestHandlerPkg
sAnswerProcessor	Метод обработки ответа. Обрабатывает ответ от внешней системы, который следует сразу синхронно за отправкой	Используется в Rpl_SendWebRequestPkg. Сигнатура: (ropXmlState: Rpl_IntXmlStateApi#ApiRop, sAnswer: String): List[SendResult]
bIsCircuitInTest	Находится ли контур в тестировании	Если в тестировании, успешно отправленные/принятые сообщения не удаляются из очередей, а переподписываются для удобства тестирования.
sMetaSchema	Метасхема	
sUploadMetaSchema	Схема выгрузки	
sDownloadMetaSchema	Схема загрузки	
sMainClassName	Имя класса шапки бизнес объекта	
sWebAddress	Адрес для отправки http запроса	
sParseIdInClassMethod	Метод поиска класса объекта	Ищет класс объекта. Сигнатура: (ropMessage: Rpl_IntXmlInMsgApi#ApiRop): NGid
sIdInClassPath	Путь для поиска класса объекта	
sAuthToken	Токен авторизации	
sWebUser	Пользователь для выполнения Http запросов	Используется в Rpl_SendWebRequestPkg
sWebPassword	Пароль для выполнения Http запросов	Используется в Rpl_SendWebRequestPkg

## 2.12 Примеры сообщений при работе со стандартным пакетным протоколом

### Импорт

Для работы с пакетным импортом необходимо использовать следующие настройки контура:

- Адрес для отправки сообщений в систему Global должен быть типа: Отправить пакетное сообщение. Отображается на закладке «конечные точки» в настройке контура интеграции.
- Метод обработки входящего сообщения: может быть как встроенные в систему метод обработки через схемы, так и кастомный.

### Пример сообщения импорта:

```
<MessagePack>
  <!--Класс объектов в пакете. можно не указывать и не заполнять если поиск класса
  →разрешается дальше в методе обработки сообщений-->
  <ObjectClass>Bs_ExpenceItem2</ObjectClass>
  <!--Системное имя контура интеграции, в который отправляется сообщение. Можно не
  →указывать, если контур передаётся иным способом. Например сразу указывается в адресе
  →отправки сообщения.-->
  <Circuit>GNSI001_FR</Circuit>
  <!--В пакете может быть несколько вложенных сообщений Message или Response-->
  <Message>
    <!--ID сообщения для указания в ответе-->
    <MessageID>FBCFD1F337287731E0530E02A8C07C59</MessageID>
    <!--Тип сообщения, object означает что в тегах будет описание объекта-->
    <MessageType>object</MessageType>
    <!--Порядковый номер сообщения для сортировки на приемной стороне-->
    <MessageOrder>538</MessageOrder>
    <!--дата создания сообщения, для аналитики или сортировки -->
    <CreateDate>2023-05-16T15:45:12</CreateDate>
    <!--система источник сообщения-->
    <SystemSource>Global</SystemSource>
    <!--Блок с данными, имя вложенного тега определяет класс и тип документа, внутри
    →полная информация и документе, статусе удаления и проведения-->
    <Data>
      <!--вложенный документ или справочник, пример ниже взят из контура обмена с 1С-->
      <Документ.ПередачаМатериаловВЭксплуатацию>
        <КлючевыеСвойства>
          <Ссылка>DD4E21FF-FD70-7440-AD43-A8A00342DAA5</Ссылка>
          <Дата>2023-05-12T07:55:16</Дата>
          <Номер>1170</Номер>
          <Организация>
            <Ссылка>4dd04906-208a-11dc-b2d3-0015170eef56</Ссылка>
            <ИНН>4706020609</ИНН>
            <КПП>470601001</КПП>
            <ЮридическоеФизическоеЛицо>ЮридическоеЛицо</ЮридическоеФизическоеЛицо>
          </Организация>
        </КлючевыеСвойства>
        <Ответственный>
          <Наименование>Тверская Оксана Валерьевна</Наименование>
          <ФизическоеЛицо>
```

(continues on next page)

```

<Ссылка>6с5f5603-5е8с-11е2-б53е-001е671031а0</Ссылка>
<ФИО>Тверская Оксана Валерьевна</ФИО>
<Фамилия>Тверская</Фамилия>
<Имя>Оксана</Имя>
<Отчество>Валерьевна</Отчество>
<ДатаРождения>1966-04-16</ДатаРождения>
<КодВПрограмме>58850170001</КодВПрограмме>
</ФизическоеЛицо>
</Ответственный>
<Товары>
  <Строка>
    <ДанныеНоменклатуры>
      <Номенклатура>
        <Ссылка>bf630b5e-1f36-11dc-b2d3-0015170eef56</Ссылка>
        <НаименованиеПолное>Костюм ИТР (3069, ШТ)</НаименованиеПолное>
        <КодВПрограмме>3069</КодВПрограмме>
        <Наименование>Костюм ИТР (3069, ШТ)</Наименование>
      </Номенклатура>
    </ДанныеНоменклатуры>
    <ЕдиницаИзмерения>
      <Ссылка>38DDB753-4475-FB4B-E053-0F02A8C07E1B</Ссылка>
      <Код>ШТ</Код>
      <Наименование>Штука</Наименование>
    </ЕдиницаИзмерения>
    <Количество>1.000</Количество>
    <КоличествоУпаковок>1.000</КоличествоУпаковок>
    <ТипЗапасов>СобственныеТовары</ТипЗапасов>
    <ФизическоеЛицо>
      <Ссылка>1а5а8fa3-efcd-11ed-8113-d8d385e18888</Ссылка>
      <ФИО>Чупраков Андрей Павлович</ФИО>
      <Фамилия>Чупраков</Фамилия>
      <Имя>Андрей</Имя>
      <Отчество>Павлович</Отчество>
      <ДатаРождения>1990-04-03</ДатаРождения>
      <КодВПрограмме>767665540001</КодВПрограмме>
      <ИНН>290408981899</ИНН>
    </ФизическоеЛицо>
    <СрокЭксплуатации>24</СрокЭксплуатации>
  </Строка>
</Товары>
</Документ.ПередачаМатериаловВЭксплуатацию>
</Data>
</Message>
</MessagePack>

```

## Пример ответа о приёме сообщения

При приёме сообщения через REST система Global даёт быстрый ответ о статусе приёма сообщения вида:

```
<MessagePack>
  <!--В пакете может быть несколько вложенных сообщений Message или Response-->
  <Response>
    <!--ID сообщения на который пришел ответ-->
    <MessageID>FBCFD1F337287731E0530E02A8C07C59</MessageID>
    <Result>false</Result>
    <ResultCode>E01</ResultCode>
    <!--Комментарий для пользователя, может быть указан как для положительных, так и для
    ↳отрицательных результатов-->
    <Description>Нет достаточного количества на складе рогов и копыт.</Description>
  </Response>
</MessagePack>
```

## Пример ответа об обработке сообщения:

```
<MessagePack>
  <!--В пакете может быть несколько вложенных сообщений Message или Response-->
  <Response>
    <!--ID сообщения-->
    <MessageID>FBCFD1F337287731E0530E02A8C07C59</MessageID>
    <!--Тип сообщения, object означает что в тегах будет описание объекта-->
    <MessageType>answer</MessageType>
    <!--Порядковый номер сообщения для сортировки на приемной стороне-->
    <MessageOrder>538</MessageOrder>
    <!--система источник сообщения-->
    <SystemSource>Global</SystemSource>
    <!--Блок с данными, имя вложенного тэга определяет класс и тип документа, внутри
    ↳полная информация и документе, статусе удаления и проведения-->
    <Data>*/
      <Response>
        <!--ID изначального входящего сообщения, на который система Global
        ↳отправляет ответ-->
        <MessageID>FBCFD1F337287731E0530E02A8C07C59</MessageID>
        <!--ID ответа, напрямую не используется, для анализа ошибок-->
        <ResponseID>FBCFD1F337297731E0530E02A8C07C59</ResponseID>
        <!--дата создания сообщения, для аналитики или сортировки -->
        <CreateDate>2023-05-16T15:46:12</CreateDate>
        <!--ID Объекта созданного или измененного в рамках сообщения, для
        ↳сохранения в таблицу сопоставления идентификаторов-->
        <ObjectID>D7965B84-53AC-4442-ACB7-77CC1D4A6983</ObjectID>
        <!--Результат приема сообщения-->
        <Result>true</Result>
        <ResultCode></ResultCode>
        <!--Комментарий для пользователя, может быть указан как для
        ↳положительных, так и для отрицательных результатов-->
        <Description></Description>
```

(continues on next page)

```

        <!--признак снятия объекта с экспорта, по умолчанию false, true_
→ передавать в особых ситуациях-->
        <NoPublish>true</NoPublish>
    </Response>
</Data>
</Response>
</MessagePack>

```

## Экспорт

Для работы с пакетным экспортом необходимо использовать следующие настройки контура:

- Адрес для отправки ответных сообщений в систему Global должен быть типа: Отправить пакетное сообщение. Отображается на закладке «конечные точки» в настройке контура интеграции;
- Метод отправки сообщений: Rpl\_SendWebRequestPkg().sendPackageRequest. В случае работы через Http запросы.

## Пример исходящего сообщения экспорта:

```

<MessagePack>
  <!--В пакете может быть несколько вложенных сообщений Message или Response-->
  <Message>
    <!--ID сообщения для указания в ответе-->
    <MessageID>FBCFD1F337287731E0530E02A8C07C59</MessageID>
    <!--Тип сообщения, object означает что в тегах будет описание объекта-->
    <MessageType>object</MessageType>
    <!--Порядковый номер сообщения для сортировки на приемной стороне-->
    <MessageOrder>538</MessageOrder>
    <!--дата создания сообщения, для аналитики или сортировки -->
    <CreateDate>2023-05-16T15:45:12</CreateDate>
    <!--система источник сообщения-->
    <SystemSource>Global</SystemSource>
    <!--Блок с данными, имя вложенного тэга определяет класс и тип документа, внутри_
→ полная информация и документе, статусе удаления и проведения-->
    <Data>
      <!--вложенный документ или справочник, пример ниже взят из контура обмена с IC-->
      <Документ.ПередачаМатериаловВЭксплуатацию>
        <КлючевыеСвойства>
          <Ссылка>DD4E21FF-FD70-7440-AD43-A8A00342DAA5</Ссылка>
          <Дата>2023-05-12T07:55:16</Дата>
          <Номер>1170</Номер>
          <Организация>
            <Ссылка>4dd04906-208a-11dc-b2d3-0015170eef56</Ссылка>
            <ИНН>4706020609</ИНН>
            <КПП>470601001</КПП>
            <ЮридическоеФизическоеЛицо>ЮридическоеЛицо</ЮридическоеФизическоеЛицо>
          </Организация>
        </КлючевыеСвойства>
        <Ответственный>

```

(continues on next page)

```

<Наименование>Тверская Оксана Валерьевна</Наименование>
<ФизическоеЛицо>
  <Ссылка>6c5f5603-5e5c-11e2-b53e-001e671031a0</Ссылка>
  <ФИО>Тверская Оксана Валерьевна</ФИО>
  <Фамилия>Тверская</Фамилия>
  <Имя>Оксана</Имя>
  <Отчество>Валерьевна</Отчество>
  <ДатаРождения>1966-04-16</ДатаРождения>
  <КодВПрограмме>58850170001</КодВПрограмме>
</ФизическоеЛицо>
</Ответственный>
<Товары>
  <Строка>
    <ДанныеНоменклатуры>
      <Номенклатура>
        <Ссылка>bf630b5e-1f36-11dc-b2d3-0015170eef56</Ссылка>
        <НаименованиеПолное>Костюм ИТР (3069, ШТ)</НаименованиеПолное>
        <КодВПрограмме>3069</КодВПрограмме>
        <Наименование>Костюм ИТР (3069, ШТ)</Наименование>
      </Номенклатура>
    </ДанныеНоменклатуры>
    <ЕдиницаИзмерения>
      <Ссылка>38DDB753-4475-FB4B-E053-0F02A8C07E1B</Ссылка>
      <Код>ШТ</Код>
      <Наименование>Штука</Наименование>
    </ЕдиницаИзмерения>
    <Количество>1.000</Количество>
    <КоличествоУпаковок>1.000</КоличествоУпаковок>
    <ТипЗапасов>СобственныеТовары</ТипЗапасов>
    <ФизическоеЛицо>
      <Ссылка>1a5a8fa3-efcd-11ed-8113-d8d385e18888</Ссылка>
      <ФИО>Чупраков Андрей Павлович</ФИО>
      <Фамилия>Чупраков</Фамилия>
      <Имя>Андрей</Имя>
      <Отчество>Павлович</Отчество>
      <ДатаРождения>1990-04-03</ДатаРождения>
      <КодВПрограмме>767665540001</КодВПрограмме>
      <ИНН>290408981899</ИНН>
    </ФизическоеЛицо>
    <СрокЭксплуатации>24</СрокЭксплуатации>
  </Строка>
</Товары>
</Документ.ПередачаМатериаловВЭксплуатацию>
</Data>
</Message>
</MessagePack>

```

При этом система ожидает ответа на запрос с данными следующего вида:

```

<MessagePack>
  <!--В пакете может быть несколько вложенных сообщений Response-->
  <Response>

```

(continues on next page)

```

<!--ID сообщения на который пришел ответ-->
<MessageID>FBCFD1F337287731E0530E02A8C07C59</MessageID>
<!--Результат приема сообщения-->
<Result>false</Result>
<ResultCode>E01</ResultCode>
<!--Комментарий для пользователя, может быть указан как для положительных, так и для
↪отрицательных результатов-->
<Description>Нет достаточного количества на складе рогов и копыт.</Description>
</Response>
</MessagePack>

```

**Пример сообщения о результате обработки сообщения экспорта:**

```

<MessagePack>
<!--В пакете может быть несколько вложенных сообщений Message или Response-->
<Response>
<!--ID сообщения на который пришел ответ-->
<MessageID>FBCFD1F337287731E0530E02A8C07C59</MessageID>
<!--ID ответа, напрямую не используется, для анализа ошибок-->
<ResponseID>FBCFD1F337297731E0530E02A8C07C59</ResponseID>
<!--дата создания сообщения, для аналитики или сортировки -->
<CreateDate>2023-05-16T15:46:12</CreateDate>
<!--ID Объекта созданного или измененного в рамках сообщения, для сохранения в
↪таблицу сопоставления идентификаторов-->
<ObjectID>D7965B84-53AC-4442-ACB7-77CC1D4A6983</ObjectID>
<!--Результат приема сообщения-->
<Result>false</Result>
<ResultCode>E01</ResultCode>
<!--Комментарий для пользователя, может быть указан как для положительных, так и для
↪отрицательных результатов-->
<Description>Нет достаточного количества на складе рогов и копыт.</Description>
<!--признак снятия объекта с экспорта, по умолчанию false, true передавать в особых
↪ситуациях-->
<NoPublish>true</NoPublish>
</Response>
</MessagePack>

```

## 3 Синхронная интеграция через REST

### 3.1 Реализация экспорта через REST

Работает по принципу экспорта. Сообщения синхронного экспорта принимает пакет Rpl\_RequestHandlerPkg на /app/sys/rest/ss/pkg/Rpl\_RequestHandlerPkg/getSynchro(Название контура) Для работы синхронного REST экспорта должен быть установлен («bExport»: «1»)

## Работа синхронного REST экспорта:

### 1. Фильтрация и регистрация объектов.

- Метод: `sAddAndFilterMessageForExport` **Описание:** Фильтрует входящие данные запроса, получая параметры запроса через `Rpl_RunIntegrationPkg().getExecutionParameters.get("#параметр")`. Возвращает список ID подписанных схем. Подпись схем для дальнейших сообщений выполняется с помощью метода `Rpl_IntXmlStateApi().register`.  
Пример базового метода фильтрации для получения `gid` из параметров запроса:

```
def addAndFilterMessageForExport(ropCircuit: Rpl_CircuitApi#ApiRop): List[Long] = {
  val gidObj = Rpl_RunIntegrationPkg().getExecutionParameters.get("gid")
  if (gidObj.isEmpty){
    AppException(s"gid отсутствует в запросе")
  }
  val xmlState = Rpl_IntXmlStateApi().register(ropCircuit.get(_id), gidObj)
  get.toGid, spAction = Rpl_MarkPkg.ActionUser
  session.commit()
  List(
    xmlState.get(_id)
  )
}
```

### 2. Генерация сообщения.

- Метод: `sGenerateMessageMethod`  
**Описание:** Генерирует сообщения на основе состояния объекта экспорта. Сгенерированное XML сообщение передается в `Rpl_IntXmlStateApi().setcStringIntoFile`.  
При наличии готовой метасхемы можно использовать методы `ru.bitec.app.rpl.channel.intg.Rpl_GenerateMessagePkg`, указав метасхему в параметре `sMetaSchema`.

### 3. Отправка ответа синхронного экспорта.

- Метод: `sSendMessageMethod`  
**Описание:** Данные для отправки ответа передаются через мапу `Rpl_RunIntegrationPkg().integrationResults` с ключом `sSynchroAnswerKey`. Метод возвращает `SendResult` по каждому объекту. `integrationResults` является инкрементной mapой, данные в `integrationResults` добавляются (накапливаются) по мере перебора сообщений готовых к отправке (ответу на запрос). Итоговый XML-ответ формируется как объединение трёх строк: префикса (`sSynchroAnswerPrefixKey`), основной части (`sSynchroAnswerKey`, данные из `sSendMessageMethod`) и постфикса (`sSynchroAnswerPostfixKey`). Поскольку `integrationResults` инкрементная, значения формируют список. Префикс и постфикс позволяют обернуть этот список, например `<Objects>(объекты из sSynchroAnswerKey)</Objects>`.
- Пример отправки ответа интеграции через REST:

```
def sendMessageMethod(ropaXmlState: List[Rpl_IntXmlStateApi#ApiRop]): List[SendResult] = {
  ropaXmlState.map(ropXmlState => {
    val integrationResultsXmlState = Rpl_RunIntegrationPkg().integrationResults.
    getOrElse(Rpl_RequestHandlerPkg().sSynchroAnswerKey, "")
    Rpl_RunIntegrationPkg().integrationResults.put(
      Rpl_RequestHandlerPkg().sSynchroAnswerKey,
      integrationResultsXmlState + Rpl_IntXmlStateApi().
    getFileAsString(ropXmlState).toString)
  })
}
```

(continues on next page)

(продолжение с предыдущей страницы)

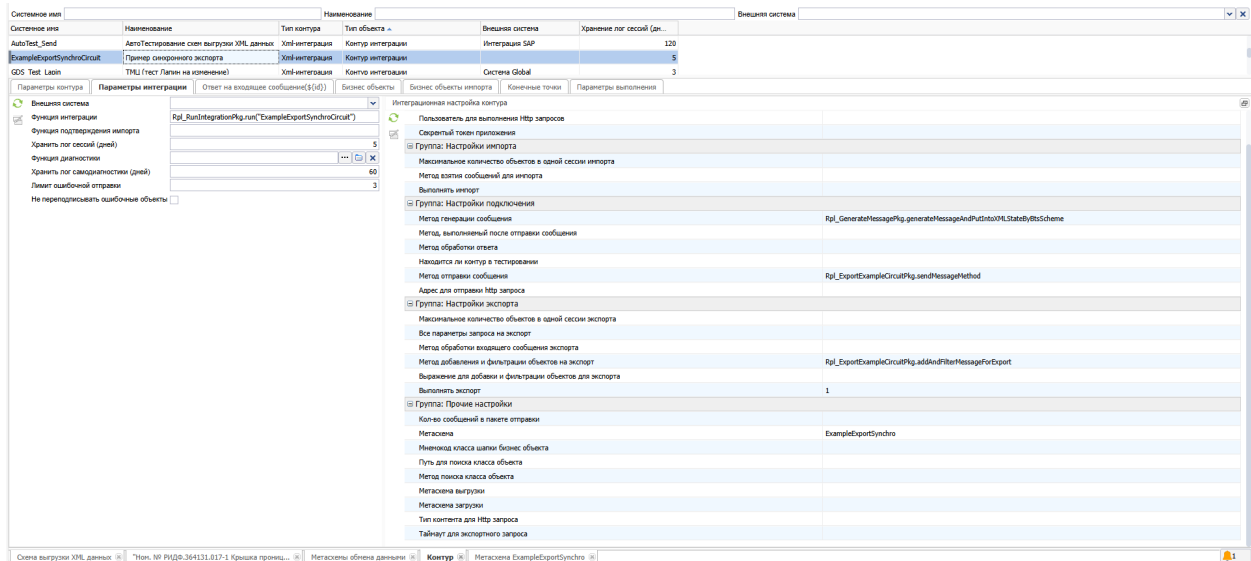
```
SendResult(  
    ropXmlState.get(_.id),  
    true,  
    "200",  
    "200",  
    false  
)  
})  
}
```

## Методы реализации экспорта через REST:

Для работы синхронного экспорта необходимо реализовать методы из таблицы.

Название метода	Описание	Сигнатура
sAddAndFindMessageForExport	Добавляет данные запроса (получить параметры запроса через <code>Rpl_RunIntegrationPkg().getExecutionParameters.get("параметр запроса")</code> ). Возвращает список ID подписанных схем. Подпись схем для дальнейших сообщений выполняется с помощью <code>Rpl_IntXmlStateApi().register</code> .	(ropCircuit: Rpl_CircuitApi#ApiRop): List[NLong]
sGenerateMessageMethod	Сообщение на основе состояния объекта, XML для отправки передается через <code>Rpl_IntXmlStateApi().setCStringIntoFile</code> как файл.	(ropXmlState: Rpl_IntXmlStateApi#ApiRop): Unit
sSendMessageMethod	Отправляет сообщение, данные для отправки ответа передаются через мапу <code>Rpl_RunIntegrationPkg().integrationResults</code> с ключом <code>sSynchroAnswerKey</code> . Возвращает результат отправки.	(ropaXmlState: List[Rpl_IntXmlStateApi#ApiRop], ropAnswer: List[SendResult])

## Пример настройки контура синхронного экспорта через REST



Методы настройки синхронного экспорта указываются в настройках контура (пример на скриншоте выше).

## 3.2 Реализация импорта через REST

Работает по принципу импорта. Сообщения синхронного импорта принимает пакет `Rpl_RequestHandlerPkg` на `/app/sys/rest/ss/pkg/Rpl_RequestHandlerPkg/sendSynchro` (Название контура) Для работы синхронного REST импорта должен быть установлен («bImport»: «1»)

### Работа синхронного REST импорта:

#### 1. Подготовка сообщений для импорта.

- Метод: `sGetMessageForImport`

**Описание:** Метод для создания импортных сообщений на основе тела запроса. Тело запроса извлекается через параметр `sSynchroDataKey` мапы `ru.bitec.app.rpl.channel.intg.Rpl_RunIntegrationPkg#getExecutionParameters()` для создания импортного сообщения через `Rpl_IntXmlInMsgApi().createMessage`.

Пример метода создания сообщения для импорта:

```
def getImportMessage(ropCircuit: Rpl_CircuitApi#ApiRop): Unit = {
    val requestBody = Rpl_RunIntegrationPkg().getExecutionParameters.get(Rpl_
    RequestHandlerPkg().sSynchroDataKey)
    requestBody match {
        case Some(value) => {
            Rpl_IntXmlInMsgApi().createMessage(ropCircuit.get(_sSystemName),
            value).get(_id).ns
            session.commit()
        }
        case _ =>
    }
}
```

#### 2. Создание сессии интеграции.

- **Описание:** `runImportSynchro` запускает синхронный вызов импорта, создавая сессию интеграции и иницилируя обработку сообщений.

#### 3. Обработка входящих сообщений.

- Метод: `sProcessIncomingMessageMethod`

**Описание:** Метод обрабатывает сообщения импорта, применяясь к каждому сообщению из очереди. Используется для обработки импортных сообщений и формирования ответа. Ответ формируется через мапу `ru.bitec.app.rpl.channel.intg.Rpl_RunIntegrationPkg.integrationResults` по ключу `sSynchroAnswerKey`.

Для обработки сообщения по метасхеме можно использовать методы из `ru.bitec.app.rpl.channel.intg.Rpl_ProcessIncomingMessagePkg` с указанием метасхемы в атрибуте `sDownloadMetaSchema` Пример обработки входящего сообщения:

```
def processIncomingCircuitSynchro(ropMessage: Rpl_IntXmlInMsgApi#ApiRop):
NGid = {
    val vGid = Rpl_ProcessIncomingMessagePkg().processIncomingMessageByBts_
    SchemaNew(ropMessage)
    session.commit()
}
```

(continues on next page)

(продолжение с предыдущей страницы)

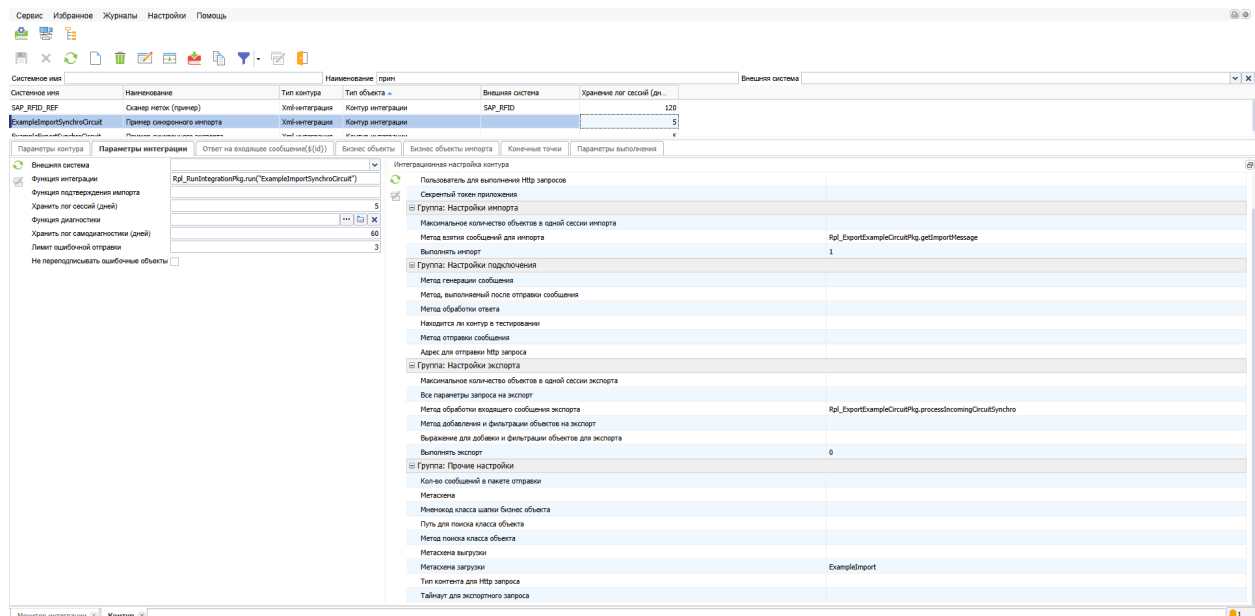
```
//Сделано для возврата ответа
//В данном примере "sMetaSchema" аналогична "sDownloadMetaSchema"
val vMetaSchema = Bts_MetaSchemaApi().findByMnemonicCode(Rpl_
IntegrationCircuitExtApi().getProp(ropMessage.get(_idCircuit),"sMetaSchema"))
val xml = Bts_XmlSchemaVerApi().getXMLText(vGid, vMetaSchema).toString
Rpl_RunIntegrationPkg().integrationResults.put(
    Rpl_RequestHandlerPkg().sSynchroAnswerKey,
    xml)
vGid
}
```

## Методы реализации импорта через REST:

Для работы синхронного импорта необходимо реализовать методы из таблицы.

Название метода	Описание	Сигнатура
sGetMessageForImport	Создает сообщение для импорта из тела запроса. Тело запроса извлекается через <code>Rpl_RunIntegrationPkg().getExecutionParameters.get("sSynchroDataKey")</code> .	(ropCircuit: Rpl_CircuitApi#ApiRop): Unit
sProcessIncomingSynchronous	Обработывает входящие сообщения и возвращает ответ через <code>integrationResults</code> , если это необходимо.	(ropMessage: Rpl_IntXmlInMsgApi#ApiRop): NGid

## Пример настройки контура синхронного импорта через REST



## 4 Прикладные сервисы

### 4.1 Сервис открытия карточки объекта по параметрам

Сервис `Btk_UrlObjectFinder` предназначен для открытия карточки объекта в системе по его атрибутам и атрибутам связанных коллекций. Поиск по сущностям происходит на основании параметров, переданных в URL query string.

URL можно создать с помощью `Btk_UrlObjectFinderPkg#createExternalCardUrl()`. Поиск производится по любому количеству атрибутов объекта, пока не превышена максимальная длина URL (рекомендуется не выходить за предел 1024-2048 символов)

#### Структура запроса

```
http://<host>:<port>/<cluster>/gtk-ru.bitec.app.btk.utils.Btk_UrlObjectFinder
↪%23UrlFinder/?ex;sTableName_dz=<TableName>[&ex;<FieldName>=<Value>]
```

- `<host>` – доменное имя или IP-адрес сервера
- `<port>` – порт сервера (по умолчанию 8080)
- `<cluster>` – рабочая база, например PGDEV, SNG-INTERNAL, OSSZ и др.
- `Btk_UrlObjectFinder%23UrlFinder` – выборка и отображение, реализующие логику сервиса
- `sTableName_dz=<TableName>` – имя таблицы (имя объекта), в которой производится поиск

**Внимание:** Имя таблицы должно быть указано в верном регистре. Например: `Btk_UrlObjectFinder`, а не `btk_urlobjectfinder`

- `<FieldName>=<Value>` – передача ключевого поля и его значения. Начинается с символов `?ex;`, каждый следующий параметр отделяется с помощью `&ex;`. Если передано несколько условий, то они комбинируются логической операцией AND
  - `Bs_Performer.id=1234` - поиск в связанной коллекции `Bs_Performer` по параметру `id` со значением 1234
  - `gidsrcobj=1/1` - поиск по собственному атрибуту `gidsrcobj`
  - `dEnd=07.07.1997` - поиск по дате `dEnd`

**Примечание:** Обратите внимание, что названия атрибутов искомого объекта регистронезависимы

В случае поиска по дате, необходимо использовать формат представления дат и временных отметок, принятый в России:

- Дата: `ДД.ММ.ГГГГ`  
Пример: **22.03.2007**
- Дата и время: `ДД.ММ.ГГГГ ЧЧ:ММ:СС`  
Пример: **23.02.2023 16:25:01**

## Поиск в связанных таблицах (коллекциях)

Сервис поддерживает поиск в связанных таблицах (настроенных на объекте коллекциях). Для указания параметра из связанной таблицы используется точечная нотация:

```
<связанная-таблица>.<псевдоним-атрибута>=<значение>
```

Если необходимо найти контрагента (Bs\_Contras) по принадлежащей Ж/Д станции (Bs\_ContrasRWStation), идентификатор idRailwayStation которой равен 2701234677, то запрос будет выглядеть следующим образом:

```
http://<host>:<port>/<cluster>/gtk-ru.bitec.app.btk.utils.Btk_UrlObjectFinder
↳%23UrlFinder/?ex;sTableName_dz=Bs_Contras&ex;Bs_ContrasRWStation.
↳idRailwayStation=2701234677
```

## 4.2 Сервис открытия карточки объекта по GID (глобальному идентификатору)

Предназначен для открытия карточки объекта в системе по его глобальному идентификатору (GID). Поиск осуществляется путем передачи идентификатора в URL query string.

Ссылка для доступа к текущей карточке находится в поле Гиперссылка на экземпляр БО модального окна Информация об объекте, открываемого с помощью операции Информация > Информация об объекте или по горячей клавише CTRL + ALT + I

### Структура запроса

```
http://<host>:<port>/<cluster>/gtk-ru.bitec.app.btk.Btk_OpenByGid%23Card/?ex;gid=<gid>
```

- <host> – доменное имя или IP-адрес сервера
- <port> – порт сервера (по умолчанию 8080)
- <cluster> – рабочая база, например PGDEV, SNG-INTERNAL, OSSZ и др.
- gtk-ru.bitec.app.btk.Btk\_OpenByGid%23Card – выборка и отображение, реализующие логику сервиса
- gid=<gid> – глобальный идентификатора для поиска объекта в системе
  - gid=1260600/27213