X-ART

Россия, Санкт-Петербург, ул. Аэродромная, д. 6, офис 29-30, тел. (812) 394-64-63, факс (812) 394-01-19, WEB: http://www.x-art.ru, E-mail: office@x-art.ru

Интеграционный гид EDI-ART

Версия 1.1
20.06.2018

Содержание

СОДЕРЖАНИЕ 2
1. ВВЕДЕНИЕ 3
2. ПОЛУЧЕНИЕ ИНФОРМАЦИИ О ЗАКАЗАХ 4
3. ОТПРАВКА ПОДТВЕРЖДЕНИЯ ЗАКАЗОВ 6
3.1 Описание взаимодействия по протоколу XML-RPC для адресата confirm80OUT 6
3.2 Пример реализации 8

1. Введение

В документе представлен процесс обмена информации между Внешними системами и системой EDI-ART для подтверждения заказов «ТД Вимос».

Информационный обмен реализован посредством взаимодействия веб-сервисов через API, путём отправки по HTTP GET и POST запросов.

Информация из системы ART-TRADE передается в формате JSON или XML.

Информация для загрузки в систему EDI-ART передается по протоколу XML-RPC[1].

Инициатором взаимодействия является Внешняя система.

2. Получение информации о заказах

Для получения информации о заказах необходимо отправить GET-запрос по адресу:

http://vi2.edi-art.net/ibapi/l/list-orders.json?INTYPE=<Тип>&PASS=<Код доступа>

Параметры вызова:

PASS – Код доступа – обязательное поле

INTYPE – Тип документа – обязательно поле.

Возможные значения:

*91-0 – Новый заказ – доступен для подтверждения

*91-2 – Обработанный заказ – уже подтвержденный заказ

*80 – Подтверждение заказа от поставщика

INZA – Номер заказа – не обязательное поле.
Возвращает данные только для конкретного заказа.
Может быть использовано для получения результата подтверждения интересуемого заказа.

Пример запроса для новых заказов

http://vi2.edi-art.net/ibapi/l/list-orders.json?INTYPE=91-0&PASS=<Код доступа>

Формат ответа

Параметр	Описание
ordernum	номер заказа
inkey	ключ документа
ttype	тип документа (91-заказ, 80-подтверждение заказа)
insost	статус документа (для 91: 0,1–новый, 2-заказ; для 80: 0-доступно к редактированию, 2-недоступно к редактированию)
inwcode	код склада
wname	наименование склада
direction	название филиала
dcode	код поставщика
dname	наименование поставщика
indate	дата заказа
innumber	номер документа
innsum	общая сумма заказа
rpos	номер позиции
gcode	внутренний артикул товара
gdealcode	артикул товара в номенклатуре поставщика (если присутствует)
gexcode	штрих-код товара
gexname	наименование товара
amount	количество заказанного
price	цена по прайсу
sumpos	сумма по позиции

JSON может быть двух видов, которые регулируются параметром **cfg.format (1|2)**:

1. В виде массива массивов (по умолчанию)

Первым элементом является массив объектов, где каждый объект содержит описание соответствующей колонки.
Последующие элементы содержат данные в виде массива значений.

http://vi2.edi-art.net/ibapi/l/list-orders.json?INTYPE=91-0&PASS=<pass>&cfg.format=1

2. В виде массива объектов.

Каждый элемент представлен в виде пары «имя поля : значение поля».

http://vi2.edi-art.net/ibapi/l/list-orders.json?INTYPE=91-0&PASS=<pass>&cfg.format=2

Ответ может быть представлен в виде **XML.**

Для этого в адресе запросе необходимо изменить **json** на **xml:**

http://vi2.edi-art.net/ibapi/l/list-orders.xml?INTYPE=<Тип>&PASS=<Код доступа>

Также можно указать формат xml-файла через доп. параметры:

cfg.xmlroot - наименование таблицы - по умолчанию "table"
cfg.xmlrow - наименование строки - по умолчанию "row"
cfg.xmlcell - наименование ячейки - по умолчанию отсутствует (берется имя колонки)
cfg.xmlattr - наименование атрибута - по умолчанию "name"

3. Отправка подтверждения заказов

3.1 Описание взаимодействия по протоколу XML-RPC для адресата confirm80OUT

Для отправки информации о продаже необходимо отправить POST-запросы по адресу:

http://vi2.edi-art.net/rpc/x-art/gxds

Сервис **GXDS** предоставляет 2 метода:

1. gxds.load_targets - получение описания цели с форматами для отправки.

Метод вызывается с параметром **targets.** Значением параметра является массив с именами адресатов.

<= [{ "targets": [<target_name>:str ...] }]

Где:

target_name – Имя адресата для отправки – константа **confirm80OUT**

Метод возвращает описание запрошенных адресатов:

```
=> [ { "name":   <target_name>:str,
       "desc":   <target_title>:str,
       "header": [ { "name":     <field_name>:str,
                     "desc":     <field_title>:str,
                     "vision":   <widget_type>:str,
                     "validator": <validation_rule>:str,
                     "variants":[ { "name": <variant_name>:str,
                                    "desc":  <variant_title>:str } ... ],
                     "default":   <default_value>:str } ... ],
       "rowset": [ { "name":     <column_name>:str,
                     "desc":     <column_title>:str,
                     "validator": <validation_rule>:str } ... ],
       "cookie": <magic_cookie>:str } ... ]
```

Где:

name – Внутреннее имя адресата, для отправки ответа

desc – Имя адресата для показа в списке

header – Спецификации полей заголовка. Описывает поля заголовка, которые должны быть заполнены пользователем

header.name – Внутреннее имя поля заголовка

header.desc – Имя поля заголовка для показа в форме

header.vision – Тип виджета, используемого для поля ввода:

header.validator – Валидатор значения поля. (Поле необязательное, может отсутствовать в структуре). Если задано, то введенное значение проверяется на соответствие регулярному выражению

header.variants – Вариант для выбора. Поле обязательно только при vision=dropdown. Задает список вариантов для выбора

header.variants.name – Внутренний вариант выбора. Это значение записывается в поле

header.variants.desc – Текст варианта выбора. Этот текст демонстрируется в dropdown

header.default – Значение по умолчанию. (Поле необязательное, может отсутствовать в структуре)

Если присутствует, то поле предзаполняется данным значением

rowset – Спецификация колонок таблицы. Описывает, какие колонки таблицы должны быть выбраны

rowset.name – Внутреннее имя колонки

rowset.desc – Имя колонки для показа

rowset.validator – Валидатор ячеек колонки. (Поле необязательное, может отсутствовать в структуре). Если поле задано, то значение каждой ячейки таблицы проверяется на соответствие регулярному выражению

cookie – Используется сервером для отслеживания версии спецификации. Это значение надо без изменений передать в функцию **gxds.post_bundle**

2. gxds.post_bundle - отправка набора данных.

Метод вызывается с набором параметров, полученных в результате вызова метода **gxds.load_targets** и их значений**:**

```
<= [ { "target": <target_name>:str,
       "header": [ <field_value>:str ... ],
       "rowset": [ [ <cell_value>:str ... ] ... ],
       "extra": [ { "name":      <tag_name>:str,
                    "value":     <tag_value>:str } ... ],
       "cookie": <magic_cookie>:str } ]
```

Где:

target – Имя адресата для отправки

header – Значение полей заголовка

rowset – Значения таблицы

extra – Дополнительные фискальные данные, которые могут помочь в разборках. Имя открытого файла, версия программы и прочее. (#agent - название программы агента)

cookie – Значение полученное от функции gxds.load_targets

Метод возвращает результат отправки данных:

```
=> { "id": <value>:str,
     "status": <value>:str,
     "message": <value>:str,
     "reply": [ { "reply.name": <value>:str, "reply.title": <value>:str, "reply.rowset": [ <value> … ] } … ] }
```

Где:

id – Присвоенный входящий номер.

status –Статус. 'E'-Ошибка, 'W'-Предупреждение, 'O'-OK. Возможны другие буквы.

message – Текстовое сообщение от обработчика пакета.

reply – Данные возвращенных колонок. Данное поле может отсутствовать в структуре или быть пустым массивом.

reply.name – Внутреннее имя возвращенной колонки

reply.title – Имя возвращенной колонки. Для показа пользователю

reply.rowset – Массив данных возвращенной колонки. Число элементов массива равно числу переданных строк в параметре **rowset**.

3.2 Пример реализации

В данном примере описана реализация с использованием JavaScript-библиотеки **json/xml-rpc**[2].

Создание объекта для работы с сервисом:

```
var gxdsService = new rpc.ServiceProxy('/path/to/rpc/x-art/gxds', {
    asynchronous:  true,
    sanitize:          true,
    methods:          ['gxds.load_targets','gxds.post_bundle'],
    protocol:          'XML-RPC'
});
```

Получение описания цели:

```
    gxdsService.gxds.load_targets( {
            params: [
                { targets:["confirm80OUT"] }
            ],
            onSuccess: function(message) {
                var target = message[0];
                processLoadTarget(target);
            },
            onException: function(e) {
                console.error("load_targets:", e);
                return true;
            }
    } );
```

Отправка пакета данных:

```
    gxdsService.gxds.post_bundle( {
            params: [
            { "target":"confirm80OUT",
                 "header": ["<Время ожидаемой поставки>","<Дата ожидаемой поставки>",
                                "<Комментарий (60 симв.)>",
                                "<Номер заказа>",
                                "<Код доступа>"],
                 "rowset": [ ["<Артикул товара>","<Количество>","<Цена>",
                                  "<Код страны происхождения>","<N таможенной декларации>"] ],
                 "extra": [ { "name":  "#agent",
                                  "value":"<Название программы>" } ],    // Название программы агента
                "cookie":"<magic_cookie>"}                                // Значение из gxds.load_targets
            ],
            onSuccess: function(message) {
                processPostBundle(message);
            },
            onException: function(e) {
                console.error("post_bundle:", e);
                return true;
            }
    });
```

[1] Официальный сайт **XML-RPC** – http://xmlrpc.scripting.com
[2] GitLab репозиторий **json/xml-rpc** – https://gitlab.com/x-art/json-xml-rpc