АПЕКС API

Версия 1.3.0

Документация

Оглавление

Введение

API системы АПЕКС – это набор инструментов, которые позволяют пользователю функционально связывать систему со сторонними приложениями, а также расширять перечень доступных функций оригинального сервиса посредством механизма пользовательских скриптов.

Набор методов API в полной мере охватывает базовый функционал системы, а также ряд прикладных функций.

Все возможности АПЕКС API подробно описаны в данной документации. Тут же есть примеры, которые помогут сориентироваться в формируемых запросах и ответах.

АПЕКС API постоянно развивается и поддерживает обратную совместимость. Написанный один раз код будет корректно работать с выходом новых версий.

Объекты системы

В системе присутствуют следующие типы объектов:

- Номенклатурная позиция (nomenclature);
- Коллекция (collection);
- Файл (file);
- Параметр (parameter);
- Компонент (component).

Помимо объектов системы существует система зависимостей, позволяющих подключать к объектам системы ссылки на объекты системы. Возможны следующие типы зависимостей:

- Компонент (*component*)*;
- Связанный параметр (*linked_parameter*);
- Связанный файл (*linked_file*);
- Связанная коллекция (*linked_collection*);
- Связанная номенклатура (*linked_nomenclature*);
- Связанная зависимость (*linked_dependence*);
- Связанная ссылка (*linked_link*);
- Связанная подпись (*linked_signature*).

* Данный тип зависимости также является объектом системы, но не может существовать сам по себе и обязательно подключается к объекту системы типа «Коллекция», идентификатор которой содержится в поле «parent_obj_id».

Также посредством API возможно осуществлять доступ к дереву каталогов справочников, где присутствуют следующие объекты:

- Каталог (folder);
- Категория (category).

Чтение объекта системы/зависимости

Параметр «event» запроса установлен в значение «get_data».

Чтение объектов системы может осуществляться по id объекта и по обозначению объекта в системе. Чтение по обозначению доступно для номенклатуры, параметров и коллекций, но недоступно для компонентов и файлов.

Ниже приведён пример чтения номенклатурной позиции с идентификатором «8485».

Запрос:

```
{
    "reqArr": [
        {
            "id": "8485",
            "type": "nomenclature"
        }
    ],
    "event": "get_data"
}
```

Ответ:

```
{
    "reqArr": [
```

```
        {
            "id": "8485",
            "type": "nomenclature",
            "name": "Тестовое обозначение",
            "notation": "Тестовое наименование",
            "status": "4",
            "status_name": "В работе",
            "link_type": "25",
            "link_type_name": "Переход",
            "created_date": "2017-09-29 22:41:05",
            "change_date": "2017-09-29 22:41:05",
            "created_date_timestamp": "1506699665",
            "change_date_timestamp": "1506699665",
            "creator": "test_user",
            "description": "<h1 class=\"ql-align-center\"><strong style=\"background-color:
rgb(52, 152, 219); color: rgb(231, 76, 60);\" class=\"ql-size-huge\">ТЕСТ</strong></h1>"
        }
    ],
    "event": "get_data",
    "error": 0
}
```


Далее приведён запрос на чтение номенклатурной позиции по обозначению.

```
{
    "reqArr": [
        {
            "type": "nomenclature",
            "name": "TEST_API_NAME"
        }
    ],
    "event": "get_data"
}
```

Также имеется возможность получить все позиции справочника (номенклатура, коллекции, параметры и файлы), находящиеся в определённом каталоге. Для этого необходимо в запросе на чтение передать параметр “folder_id”, соответствующий идентификатору каталога или категории.

Далее приведён пример запроса на чтение номенклатурных позиций, находящихся в каталоге с идентификатором «7».

```
{
    "reqArr": [
        {
            "type": "nomenclature",
            "folder_id": "7"
        }
    ],
    "event": "get_data"
}
```

Для коллекций, номенклатуры и параметров существует возможность выдачи результатов выборки с учётом фильтрации по критерию наличия искомой строки, соответствующей значению поля «filter», в обозначении или наименовании позиции. Поле «filter» может быть использовано совместно с полями «folder_id», «last_id», «amount» и, в случае выборки коллекций, с полями «link_type» и «status». Поле «filter» не может использоваться в запросах с выборкой по конкретному идентификатору, т.е. совместно с полем «id».

Далее приводится пример выборки всех коллекций, доступных пользователю, содержащих в обозначении или наименовании строку «SEARCH STRING».

```
{
    "event": "get_data",
    "reqArr": [
        {
            "type": "collection",
```

```
            "filter": "SEARCH STRING"
        }
    ]
}
```

При выборке коллекций без указания идентификатора существует возможность указывать тип связи (поле «link_type») и статус (поле «status»). Запрос ниже возвращает все коллекции , с идентификатором типа связи «96», доступные текущему пользователю.

```
{
    "event": "get_data",
    "reqArr": [
        {
            "type": "collection",
            "link_type": "96"
        }
    ]
}
```

Для коллекций доступна фильтрация по дате (учитывается дата без времени), описываемая полями «date_filter_ts_start» и «date_filter_ts_end», которые представляют из себя начальную и конечную точку временного диапазона (Unix Time Stamp), на вхождение в который проверяется диапазон между датой начала и датой окончания коллекций (включительно). Далее приведён пример выборки коллекций, доступных пользователю, диапазон начала/окончания которых входит во временной отрезок 27.12.2022 - 25.01.2023.

```
{
    "event": "get_data",
    "reqArr": [
        {
            "type": "collection",
            "date_filter_ts_start": "1672099200",
            "date_filter_ts_end": "1674604800"
        }
    ]
}
```

Если задан только параметр «date_filter_ts_start» и не указан «date_filter_ts_end», то выборка осуществляется от значения параметра «date_filter_ts_start» до условной бесконечности.

Если задан только параметр «date_filter_ts_end» и не указан «date_filter_ts_end», то выборка осуществляется от времени начала эпохи Unix (01.01.1970) до значения параметра «date_filter_ts_end».

Если при выборке коллекций необходимо контролировать на вхождение в диапазон исключительно дату начала или дату окончания коллекции, стоит использовать параметр «filter_by_point», установленный в значение «start» или «end» соответственно. Далее приведён пример выборки коллекций, доступных пользователю, дата начала которых входит во временной отрезок 27.12.2022 - 25.01.2023.

```
{
    "event": "get_data",
    "reqArr": [
        {
            "type": "collection",
            "date_filter_ts_start": "1672099200",
            "date_filter_ts_end": "1674604800",
            "filter_by_point": "start"
        }
    ]
}
```

Также доступна постраничная выдача результатов выборки, которая реализуется полями «last_id» и «amount», где поле «last_id» — это идентификатор последнего объекта с прошлой страницы, т.е.

все объекты новой выборки должны иметь идентификатор меньше, чем «last_id» (используется для постраничной выдачи). Поле «amount» используется для того, чтобы указать размерность страницы, т.е. сколько элементов должно быть возвращено в результате выполнения текущего запроса.

Ниже приведён запрос на выдачу страницы из 100 коллекций, доступных текущему пользователю, с идентификатором типа связи «96» и идентификатором статуса «10», идентификатор которых меньше, чем «323723».

```
{
    "event": "get_data",
    "reqArr": [
        {
            "type": "collection",
            "link_type": "96",
            "status": "10",
            "amount": "100",
            "last_id": "323723"
        }
    ]
}
```

Если необходимо сделать выборку по нескольким статусам, либо напротив исключить из выборки позиции по признаку определённого статуса или нескольких, то в качестве содержимого поля «status» должен быть передан объект содержащий в себе поля «values» и «selection», где поле «values» представляет из себя массив с идентификаторами статусов, а поле «selection» - тип выборки со значениями «not» (исключить из выборки позиции с выбранными статусами) или «or» (включить в выборку позиции с выбранными статусами).

Далее приведён пример запроса на получение 100 штук коллекций с идентификатором типа связи «119» кроме тех, идентификаторы статусов которых установлены в значение «1» или «2».

```
{
    "event": "get_data",
    "reqArr": [
        {
            "amount": 100,
            "type": "collection",
            "link_type": "119",
            "status": {
                "selection": "not",
                "values": [
                    1,
                    2
                ]
            }
        }
    ]
}
```

Сервер возвращает объекты системы в формате JSON строки со следующей структурой полей:

Номенклатурная позиция:

- id – уникальный идентификатор позиции среди объектов своего типа;
- type – тип объекта (“*nomenclature*”);
- name – обозначение позиции (используется вместо «id»);
- notation – наименование позиции;
- created_date – дата и время создания позиции;
- change_date – дата и время изменения позиции;
- created_date_timestamp – время создания позиции (Unix Time Stamp);
- change_date_timestamp – время изменения позиции (Unix Time Stamp);

- creator – имя пользователя, создавшего позицию;
- description – содержимое блока «описание» номенклатурной позиции;
- status – статус позиции;
- link_type – тип связи позиции;

Коллекция:

- id – уникальный идентификатор позиции среди объектов своего типа;
- type – тип объекта (“*collection”*);
- name – обозначение позиции;
- notation – наименование позиции;
- link_type – тип связи позиции;
- link_type_name – обозначение типа связи коллекции;
- status – статус позиции;
- status_name – обозначение статуса коллекции;
- start_date – дата начала (формат строки: YYYY-MM-DD);
- start_time – время начала (формат строки: HH:MM:SS);
- end_date – дата окончания (формат строки: YYYY-MM-DD);
- end_time – время окончания (формат строки: HH:MM:SS);
- created_date – дата и время создания позиции;
- change_date – дата и время изменения позиции;
- created_date_timestamp - время создания коллекции (Unix Time Stamp);
- change_date_timestamp - время изменения коллекции (Unix Time Stamp);
- creator - имя пользователя, создавшего коллекцию;
- start_timestamp – время начала коллекции (Unix Time Stamp);
- end_timestamp – время окончания коллекции (Unix Time Stamp);
- description – содержимое блока «описание» коллекции;
- priority – приоритет;

Файл:

- id – уникальный идентификатор позиции среди объектов своего типа;
- type – тип объекта (*“file”*);
- name – имя файла;
- file_type – расширение файла;
- link_type – тип связи позиции;
- link_type_name – обозначение типа связи файла;
- status – статус позиции;
- status_name – обозначение статуса коллекции;
- comment – тип связи позиции;
- size – размер файла в байтах;
- created_date – дата и время создания позиции;
- change_date – дата и время изменения позиции;
- created_date_timestamp - время создания позиции (Unix Time Stamp);
- change_date_timestamp - время изменения позиции (Unix Time Stamp);
- creator - имя пользователя, создавшего позицию;
- description – содержимое блока «описание» файла;
- uploaded_file_change_date – время изменения в метаданных файла (Unix Time Stamp);
- uploaded_file_hash - контрольная сумма файла (SHA-256);

Параметр:

- id – уникальный идентификатор позиции среди объектов своего типа;
- type – тип объекта (*“parameter”*);
- name – обозначение позиции;
- notation – наименование позиции;
- parameter_type – тип параметра;
- created_date – дата и время создания позиции;
- change_date – дата и время изменения позиции;
- created_date_timestamp - время создания позиции (Unix Time Stamp);
- change_date_timestamp - время изменения позиции (Unix Time Stamp);
- creator - имя пользователя, создавшего позицию;
- description – содержимое блока «описание» параметра;

Компонент:

- id – уникальный идентификатор позиции среди объектов своего типа;
- type – тип объекта (“*component”*);
- nomenclature_id – идентификатор номенклатурной позиции, на базе которой был создан компонент;
- nomenclature_name – обозначение номенклатуры в справочнике номенклатуры, на базе которой был создан компонент;
- nomenclature_notation – наименование номенклатуры в справочнике номенклатуры, на базе которой был создан компонент;
- parent_obj_id – идентификатор коллекции в которой содержится данный компонент;
- parent_id – идентификатор родительского объекта типа «компонент» в дереве компонентов;
- level – уровень иерархии компонента в дереве компонентов (присутствует только для «event» = «get_obj_blocks»);
- link_type – тип связи позиции;
- link_type_name – обозначение типа связи;
- status – статус позиции;
- status_name – обозначение статуса;
- amount – количество;
- unit – единица измерения;
- unit_name – обозначение единицы измерения;
- comment – комментарий;
- price – цена;
- start_date – дата начала (формат строки: YYYY-MM-DD);
- start_time – время начала (формат строки: HH:MM:SS);
- end_date – дата окончания (формат строки: YYYY-MM-DD);
- end_time – время окончания (формат строки: HH:MM:SS);
- start_timestamp – время начала (Unix Time Stamp);
- end_timestamp – время окончания (Unix Time Stamp);
- created_date_timestamp - время создания компонента (Unix Time Stamp);
- change_date_timestamp - время изменения компонента (Unix Time Stamp);
- creator - имя пользователя, создавшего компонент;
- description – содержимое блока «описание» компонента;
- priority – приоритет;

Далее приведены поля, возвращаемые для зависимостей:

Связанный параметр:

- id – уникальный идентификатор позиции среди объектов своего типа;
- type – тип зависимости (*"*linked_parameter*"*);
- parameter_id – идентификатор параметра в справочнике параметров, на базе которого был создан связанный параметр;
- parent_obj_id – идентификатор объекта, к которому прикреплён данный параметр;
- parent_obj_type - тип родительского объекта (nomenclature, collection, file, component);
- param_id – техническое поле, дублирующее поле «parameter_id»;
- parent_id – идентификатор родительского объекта типа «связанный параметр» в дереве связанных параметров;
- s_value – значение текстового параметра (тип Text);
- d_value – значение числового параметра (тип Float);
- tolerance_plus – допуск +;
- tolerance_minus – допуск -;
- comment – комментарий;
- formula_id – расчётный ID;
- formula – формула;
- created_date_timestamp – время создания связанного параметра (Unix Time Stamp);
- change_date_timestamp – время изменения связанного параметра (Unix Time Stamp);
- creator – имя пользователя, создавшего связанный параметр;
- parameter_name – обозначение базового параметра;
- parameter_notation – наименование базового параметра;
- parameter_type – тип базового параметра;

Связанный файл

- id – уникальный идентификатор позиции среди объектов своего типа;
- type – тип зависимости (*"*linked_file*"*);
- file_id – идентификатор файла в справочнике файлов, на базе которого был создан связанный файл;
- parent_obj_id – идентификатор объекта, к которому прикреплён данный файл;
- parent_obj_type - тип родительского объекта (nomenclature, collection, file, component);
- created_date_timestamp – время создания связанного файла (Unix Time Stamp);
- change_date_timestamp – время изменения связанного файла (Unix Time Stamp);
- creator – имя пользователя, создавшего связанный файл;
- file_name – имя базового файла;
- file_type – тип базового файла;
- file_size – размер базового файла;
- file_link_type – id типа связи базового файла;
- file_comment – комментарий базового файла;
- link_type_name – обозначение типа связи базового файла;

Связанная коллекция

- id – уникальный идентификатор позиции среди объектов своего типа;
- type – тип объекта (*"linked_collection"*);

- collection_id - идентификатор коллекции в справочнике коллекций, на базе которой была создана связанная коллекция;
- parent_obj_id – идентификатор объекта, к которому прикреплёна данная коллекция;
- parent_obj_type - тип родительского объекта (nomenclature, collection, component);
- assignment_link_type – назначение позиции (тип связи);
- created_date_timestamp – время создания связанной коллекции (Unix Time Stamp);
- change_date_timestamp – время изменения связанной коллекции (Unix Time Stamp);
- collection_name – обозначение базовой коллекции;
- collection_notation – наименование базовой коллекции;
- collection_link_type – тип связи базовой коллекции;
- collection_status – статус базовой коллекции;
- collection_start_date – дата начала базовой коллекции;
- collection_start_time – время начала базовой коллекции;
- collection_end_date – дата окончания базовой коллекции;
- collection_end_time – время окончания базовой коллекции;
- start_timestamp – время начала базовой коллекции (Unix Time Stamp);
- end_timestamp – время окончания базовой коллекции (Unix Time Stamp);
- link_type_name – обозначение типа связи базовой коллекции;
- status_name – обозначение статуса базовой коллекции;

Связанная номенклатура

- id – уникальный идентификатор позиции среди объектов своего типа;
- type – тип зависимости (*"*linked_nomenclature*"*);
- nomenclature_id – идентификатор номенклатуры в справочнике номенклатуры, на базе которой была создана связанная номенклатура;
- parent_obj_id – идентификатор объекта, к которому прикреплена данная номенклатура;
- parent_obj_type - тип родительского объекта (nomenclature, collection, file, component);
- link_type – тип связи;
- link_type_name – обозначение типа связи;
- comment – комментарий;
- created_date_timestamp – время создания связанной номенклатуры (Unix Time Stamp);
- change_date_timestamp – время изменения связанной номенклатуры (Unix Time Stamp);
- creator – имя пользователя, создавшего прикреплённую номенклатуру;
- nomenclature_name – обозначение базовой номенклатуры;
- nomenclature_notation – наименование базовой номенклатуры;

Связанная зависимость

- id – уникальный идентификатор позиции среди объектов своего типа;
- type – тип зависимости (*"linked_dependence"*);
- component_id – идентификатор компонента, на базе которого создана зависимость;
- parent_obj_id – идентификатор объекта, к которому прикреплён данный параметр;
- creator – имя пользователя, создавшего зависимость;
- created_date_timestamp – время создания зависимости (Unix Time Stamp);
- change_date_timestamp – время изменения зависимости (Unix Time Stamp);
- nomenclature_name – обозначение базовой номенклатуры;
- nomenclature_notation – наименование базовой номенклатуры;
- nomenclature_id – id базовой номенклатуры;

- component_parent_id – идентификатор родительского объекта типа «компонент» в дереве компонентов;
- component_link_type – тип связи базового компонента;
- component_status – статус базового компонента;
- component_amount – количество базового компонента;
- component_unit – единица измерения базового компонента;
- component_comment – комментарий базового компонента;
- component_price – цена базового компонента;
- component_start_date – дата начала базового компонента;
- component_start_time – время начала базового компонента;
- component_end_date – дата окончания базового компонента;
- component_end_time – время окончания базового компонента;
- component_start_timestamp – время начала базового компонента (Unix Time Stamp);
- component_end_timestamp – время окончания базового компонента (Unix Time Stamp);
- link_type_name – обозначение типа связи базового компонента;
- status_name – обозначение статуса базового компонента;
- unit_name – обозначение единицы измерения базового компонента;

Связанная ссылка

- id – уникальный идентификатор позиции среди объектов своего типа;
- type – тип зависимости (*"*linked_link*"*);
- link – ссылка;
- parent_obj_id – идентификатор объекта, к которому прикреплена данная ссылка;
- parent_obj_type - тип родительского объекта (nomenclature, collection, file, component);
- comment – комментарий;
- created_date_timestamp – время создания ссылки (Unix Time Stamp);
- change_date_timestamp – время изменения ссылки (Unix Time Stamp);
- creator – имя пользователя, создавшего ссылку;

Связанная подпись

- id – уникальный идентификатор позиции среди объектов своего типа;
- type – тип зависимости (*"linked_signature"*);
- parent_obj_id – идентификатор объекта, к которому прикреплена данная подпись;
- parent_obj_type - тип родительского объекта (nomenclature, collection, file, component);
- creator_id – id пользователя, создавшего связанную подпись;
- name – наименование подписи;
- signer_name – имя подписывающего;
- signer_id – id подписывающего пользователя;
- signer_type – тип подписывающего пользователя (пользователь – «user»/группа – «group»);
- signed_by_id – id подписавшего пользователя;
- signed_date_timestamp – время подписания;
- created_date_timestamp – время создания прикреплённой подписи;
- last_change_date_timestamp – время изменения прикреплённой подписи;
- comment – комментарий;

Чтение содержимого блоков объекта

Параметр «event» запроса установлен в значение «get_obj_blocks».

Сервер возвращает содержимое блоков объекта системы в формате JSON строки.

Тело запроса для объекта состоит из двух обязательных полей

- type – тип объекта;
- id – id объекта;

Далее приведён запрос на чтение всех блоков коллекции.

```
{
    "reqArr": [
        {
            "type": "collection",
            "id": 278208
        }
    ],
    "event": "get_obj_blocks"
}
```

Далее приведён ответ сервера на чтение всех блоков коллекции.

```
{
    "event": "get_obj_blocks",
    "reqArr": [
        {
            "type": "collection",
            "id": 278208,
            "blocks": {
                "linked_components": [
                    {
                        "id": "2475841",
                        "type": "linked_component",
                        "nomenclature_id": "107308",
                        "parent_id": "0",
                        "link_type": "42",
                        "status": "10",
                        "amount": "123",
                        "unit": "20",
                        "comment": "test comment",
                        "parent_obj_id": "278208",
                        "price": "456.00",
                        "start_date": "2022-09-01",
                        "start_time": "04:15:00",
                        "end_date": "2022-09-30",
                        "end_time": "04:30:00",
                        "creator": "apeks",
                        "start_timestamp": "1661980500",
                        "end_timestamp": "1664487000",
                        "nomenclature_name": "TEST NOMENCLATURE 1",
                        "nomenclature_notation": "NOTATION",
                        "link_type_name": "Материал",
                        "status_name": "В работе",
                        "level": 2,
                        "unit_name": "кг",
                        "priority": 0
                    }
                ],
                "linked_parameters": [
                    {
                        "id": "8133",
                        "type": "linked_parameter",
                        "nom_id": "278208",
                        "parent_obj_id": "278208",
                        "parent_obj_type": "collection",
                        "param_id": "478",
                        "parent_id": "0",
                        "s_value": "",
```

```
                    "d_value": "123",
                    "tolerance_plus": "1",
                    "tolerance_minus": "2",
                    "comment": "test comment",
                    "param_name": "API_TEST_PARAM_NAME",
                    "param_notation": "API_TEST_PARAM_NOTATION",
                    "param_type": "Float",
                    "formula_id": "A",
                    "formula": "A+B",
                    "creator": "apeks",
                    "created_date_timestamp": "1571672546",
                    "change_date_timestamp": "1571672546"
                }
            ],
            "linked_files": [
                {
                    "id": "285191",
                    "type": "linked_file",
                    "file_id": "5232",
                    "creator": "apeks",
                    "file_name": "test.pdf",
                    "file_type": "pdf",
                    "file_size": "515820",
                    "file_link_type": "0",
                    "file_comment": "",
                    "link_type_name": "",
                    "parent_obj_id": "383",
                    "parent_obj_type": "collection",
                    "created_date_timestamp": "1571672546",
                    "change_date_timestamp": "1571672546"
                }
            ],
            "linked_links": [
                {
                    "id": "165",
                    "type": "linked_link",
                    "link": "www.test_link.ru",
                    "comment": "test comment",
                    "creator": "apeks",
                    "parent_obj_id": "383",
                    "parent_obj_type": "collection",
                    "created_date_timestamp": "1571672546",
                    "change_date_timestamp": "1571672546"

                }
            ],
            "linked_collections": [
                {
                    "id": "583992",
                    "type": "linked_collection",
                    "collection_id": "278209",
                    "assignment_link_type": "118",
                    "creator": "apeks",
                    "collection_name": "TEST COLLECTION",
                    "collection_notation": "NOTATION",
                    "collection_link_type": "63",
                    "collection_status": "10",
                    "collection_start_date": "2022-09-02",
                    "collection_start_time": "03:00:00",
                    "collection_end_date": "2022-09-30",
                    "collection_end_time": "03:45:00",
                    "start_timestamp": "1662062400",
                    "end_timestamp": "1664484300",
                    "link_type_name": "Технология",
                    "status_name": "В работе",
                    "parent_obj_id": "383",
                    "parent_obj_type": "collection",
                    "created_date_timestamp": "1571672546",
                    "change_date_timestamp": "1571672546",
                    "collection_priority": 0

                }
            ],
```

```
                "linked_nomenclature": [
                    {
                        "id": "212736",
                        "type": "linked_nomenclature",
                        "nomenclature_id": "107308",
                        "link_type": "69",
                        "comment": "test comment",
                        "creator": "apeks",
                        "nomenclature_name": "TEST NOMENCLATURE 1",
                        "nomenclature_notation": "NOTATION",
                        "link_type_name": "Деталь",
                        "parent_obj_id": "383",
                        "parent_obj_type": "collection",
                        "created_date_timestamp": "1571672546",
                        "change_date_timestamp": "1571672546"
                    }
                ],
                "rights": [
                    {
                        "user": "apeks",
                        "reader": "1",
                        "associate": "1",
                        "creator": "1"
                    }
                ],
                "using": [
                    {
                        "id": "360",
                        "type": "collection",
                        "name": "TEST COLLECTION NAME",
                        "notation": "TEST COLLECTION NOTATION",
                        "block_name": "linked_nomenclature",
                        "linked_id": "13148"
                    }
                 ],
                "linked_signatures": [
                    {
                        "id": "84",
                        "type": "linked_signature",
                        "creator_id": "1",
                        "name": "SIGNATURE NAME",
                        "signer_name": "FULL NAME",
                        "signer_id": "195",
                        "signer_type": "user",
                        "signed_by_id": "0",
                        "creator": "apeks",
                        "signed_date_timestamp": null,
                        "created_date_timestamp": "1663429266",
                        "last_change_date_timestamp": "1663429266",
                        "comment": "test comment",
                        "signed_by": "",
                        "signer": "apeks"
                    }
                ]
            }
        }
    ],
    "error": 0
}
```


Стоит учитывать, что по умолчанию содержимое блоков «Использование» («using») и «Обсуждение» («discussion») не включается в результат выдачи блоков (если параметр «block_type» не указывает их явно). Для выдачи блока «Обсуждение» в числе остальных блоков необходимо использовать параметр «show_discussion_flag», установленный в значение «1», а для выдачи блока «Использование» - параметр «show_using_flag», установленный в значение «1». Ниже приведён пример такого запроса.

```
{
    "event": "get_obj_blocks",
    "show_using_flag": 1,
```

```
    "reqArr": [
        {
            "type": "collection",
            "id": "383"
        }
    ]
}
```

Также имеется возможность получения одного конкретного блока. Для этого используется необязательный параметр «block_type» – тип блока («linked_parameter», «linked_component», «linked_file», «linked_link», «linked_collection», «linked_nomenclature», «linked_dependence», «linked_signature», «right», «using», «discussion») указывается вне массива «reqArr» на одном уровне с параметром «event». Если данный параметр не указан, то будут получены все блоки объекта.

Далее приведён запрос на чтение блока компонентов коллекции с идентификатором «278208».

```
{
    "event": "get_obj_blocks",
    "block_type": "linked_component",
    "reqArr": [
        {
            "type": "collection",
            "id": 278208
        }
    ]
}
```

Далее приведён ответ сервера, содержащий блок компонентов коллекции с идентификатором «278208».

```
{
    "event": "get_obj_blocks",
    "block_type": "linked_component",
    "reqArr": [
        {
            "type": "collection",
            "id": 278208,
            "blocks": {
                "linked_components": [
                    {
                        "id": "2475841",
                        "type": "linked_component",
                        "nomenclature_id": "107308",
                        "parent_id": "0",
                        "link_type": "42",
                        "status": "10",
                        "amount": "123",
                        "unit": "20",
                        "comment": "test comment",
                        "parent_obj_id": "278208",
                        "price": "456.00",
                        "start_date": "2022-09-01",
                        "start_time": "04:15:00",
                        "end_date": "2022-09-30",
                        "end_time": "04:30:00",
                        "start_timestamp": "1661980500",
                        "end_timestamp": "1664487000",
                        "nomenclature_name": "TEST NOMENCLATURE 1",
                        "nomenclature_notation": "NOTATION",
                        "link_type_name": "Материал",
                        "status_name": "В работе",
                        "unit_name": "кг",
                        "creator": "apeks",
                        "priority": 0
                    }
                ]
            }
        }
    ],
```

```
    "error": 0
}
```


Далее приведён запрос на чтение блока «Обсуждение» номенклатурной позиции с идентификатором «1527».


```
{
    "reqArr": [
        {
            "id": 1527,
            "type": "nomenclature"
        }
    ],
    "event": "get_obj_blocks",
    "block_type": "discussion"
}
```


Далее приведён ответ сервера, содержащий блок «Обсуждение» номенклатурной позиции с идентификатором «1527».


```
{
    "reqArr": [
        {
            "id": 1527,
            "type": "nomenclature",
            "error": 0,
            "blocks": {
                "discussion": [
                    {
                        "id": "28",
                        "created_date": "2024-03-16 22:25:01",
                        "created_date_string": "16.03.2024 в 22:25",
                        "created_date_timestamp": "1710602701",
                        "text": "ANSWER",
                        "user_login": "user1",
                        "data": "null",
                        "edited": "0"
                    },
                    {
                        "id": "27",
                        "created_date": "2024-03-16 22:24:55",
                        "created_date_string": "16.03.2024 в 22:24",
                        "created_date_timestamp": "1710602695",
                        "text": "HELLO",
                        "user_login": "user2",
                        "data": "null",
                        "edited": "0"
                    }
                ]
            }
        }
    ],
    "event": "get_obj_blocks",
    "error": 0,
    "block_type": "discussion"
}
```


Для каждого из блоков объекта возвращается массив объектов, содержащих определённый набор полей в зависимости от типа блока.

Для блока «Использование» («using»):

- id – идентификатор объекта, использующего целевой объект;
- type – тип объекта, использующего целевой объект;
- name – обозначение объекта, использующего целевой объект;
- notation – наименование объекта, использующего целевой объект;
- block_name – тип блока объекта, в котором используется целевой объект;

- linked_id – идентификатор зависимости (связанной номенклатурной позиции, связанной коллекции и т.д.);
- s_value (только для блока «использование» параметра) - строковое значение для строковых параметров;
- d_value (только для блока «использование» параметра) - числовое значение для числовых параметров;

Для блока «Обсуждение» («discussion»):

- id – уникальный идентификатор комментария;
- created_date – дата создания комментария (формат строки: YYYY-MM-DD HH:MM:SS);
- created_date_string – дата создания комментария (формат строки: DD.MM.YYYY в* HH:MM);
- created_date_timestamp – дата создания комментария (Unix Time Stamp);
- text – текст комментария;
- user_login – имя пользователя, оставившего комментарий;
- data – дополнительная информация о сообщении в формате JSON;
- edited – флаг редактирования комментария (установлен в «1», если комментарий редактировался автором);

* Зависит выбранного языка интерфейса у пользователя, выполняющего запрос.

Чтение определённого блока объекта с возможностью фильтрации

Параметр «event» запроса установлен в значение «get_obj_block».

В настоящее время метод работает с объектами типа «collection» и типом блока «linked_component».

Данный метод позволяет получить содержимое конкретного блока объекта или выборки объектов с возможностью фильтрации как выборки родительских объектов, так и содержимого блока.

Обязательный параметр «block_type» – тип блока («linked_parameter», «linked_component», «linked_file», «linked_link», «linked_collection», «linked_nomenclature», «linked_dependence», «linked_signature», «right», «using») указывается вне массива «reqArr» на одном уровне с параметром «event».

Тело запроса состоит из следующих полей:

- type – тип родительского объекта («nomenclature», «collection», «parameter», «file», «component»);
- id – id родительского объекта;
- link_type_id – идентификатор типа связи родительского объекта (используется для массовой выборки родительских объектов, если не указан id родительского объекта);
- link_type_name – обозначение типа связи родительского объекта (используется для массовой выборки родительских объектов, если не указаны id родительского объекта и «link_type_id»);
- status_id – идентификатор статуса родительского объекта (используется для массовой выборки родительских объектов, если не указан id родительского объекта);
- status_name – обозначение статуса родительского объекта (используется для массовой выборки родительских объектов, если не указаны id родительского объекта и «link_type_id»);

- block_amount – количество элементов блока, которое должно быть возвращено в результате выполнения текущего запроса;
- block_last_id – это идентификатор последнего элемента блока с прошлой страницы, т.е. все объекты новой выборки должны иметь идентификатор меньше, чем «last_id» (используется для постраничной выдачи);
- block_status – идентификатор статуса элемента блока, либо объект описания выборки по статусам;
- block_filter – подстрока для выборки по обозначению или наименованию элементов блока;
- block_date_filter_ts_start – начало диапазона фильтрации элементов блока по дате (Unix Time Stamp);
- block_date_filter_ts_end – конец диапазона фильтрации элементов блока по дате (Unix Time Stamp);

Выдача записей осуществляется в порядке от новых к старым.

Далее приведён пример получения 100 штук компонентов, содержащихся внутри коллекций со связью «Заказ». Идентификатор статуса компонентов не должен соответствовать значениям «11» и «12». В обозначении или наименовании компонента (номенклатурной позиции, на базе которой он был создан) должна присутствовать подстрока «promt». Диапазон дат начала и окончания компонента должен пересекаться с диапазоном, находящимся внутри временных меток Unix Time Stamp «1687539600» и «1687971600».

Запрос:

```
{
    "event": "get_obj_block",
    "block_type": "linked_component",
    "reqArr": [
        {
            "type": "collection",
            "link_type_name": "Заказ",
            "block_amount": 100,
            "block_status": {
                "selection": "not",
                "values": [
                    "11",
                    "12"
                ]
            },
            "block_filter": "promt",
            "block_date_filter_ts_start": 1687539600,
            "block_date_filter_ts_end": 1687971600
        }
    ]
}
```

Ответ представляет из себя стандартную выдачу содержимого блока объекта по типу метода «get_obj_blocks».

Чтение истории объекта

Параметр «event» запроса установлен в значение «get_history».

Данный метод позволяет получить записи журнала истории объекта. История объекта хранит информацию о действиях, совершаемых пользователями над объектом, и записями в блоках

объекта, такими как создание/правка/удаление объекта, создание/правка/удаление записи внутри одного из блоков объекта.

Тело запроса для объекта состоит из следующих полей:

- type – тип объекта;
- id – id объекта;
- amount – количество элементов, которое должно быть возвращено в результате выполнения текущего запроса;
- last_id – это идентификатор последнего объекта с прошлой страницы, т.е. все объекты новой выборки должны иметь идентификатор меньше, чем «last_id» (используется для постраничной выдачи);
- history_id – уникальный идентификатор отдельно взятой записи лога. Может быть использован только вместо приведённых выше полей и возвращает содержимое одной конкретной записи журнала при условии наличия прав на объект системы, к которому данная запись относится.

Тело ответа сервера помимо полей запроса включает в себя поле «history», представляющее из себя массив записей. Каждая запись в свою очередь содержит следующие поля:

- id – идентификатор записи журнала истории;
- user_id – идентификатор пользователя, совершившего действие;
- timestamp – время действия (Unix Time Stamp);
- date_format – отформатированная строка, содержащая дату и время действия;
- obj_id – id объекта, к которому относится запись журнала;
- obj_type – тип объекта, к которому относится запись журнала;
- linked_obj_id – идентификатор связанного объекта (если действие совершено в отношении связанной с объектом записи, т.е. содержимого блоков объекта);
- linked_obj_type – тип связанного объекта (если действие совершено в отношении связанной с объектом записи, т.е. содержимого блоков объекта);
- linked_obj_original_id – идентификатор оригинала справочной позиции связанного объекта (если действие совершено в отношении связанной с объектом записи, т.е. содержимого блоков объекта);
- linked_obj_original_type – тип оригинала справочной позиции связанного объекта (если действие совершено в отношении связанной с объектом записи, т.е. содержимого блоков объекта);
- multi_count – количество позиций, к которым было применено групповое действие (0 – действие негрупповое);
- event_code – код типа события (0 – удаление, 1 – создание, 2 – редактирование, 3 – предоставление прав);
- user – логин пользователя, совершившего действие;
- event_text – текстовое представление записи журнала.

Выдача записей осуществляется в порядке от новых к старым.

Далее приведён пример получения двух записей истории для коллекции с идентификатором «383», выдача должна осуществляться начиная с идентификатора записи «2343» (будут возвращены записи с идентификатором меньше 2343).

Запрос:

```
{
    "event": "get_history",
```

```
    "reqArr": [
        {
            "id": 383,
            "type": "collection",
            "amount": 2,
            "last_id": 2343
        }
    ]
}
```


Ответ:


```
{
    "event": "get_history",
    "reqArr": [
        {
            "id": 383,
            "type": "collection",
            "amount": 10,
            "last_id": 691971,
            "history": [
                {
                    "id": "691970",
                    "user_id": "10",
                    "timestamp": "1684945307",
                    "date_format": "24-05-2023 23:21:47",
                    "obj_id": "383",
                    "obj_type": "collection",
                    "linked_obj_id": "3224035",
                    "linked_obj_type": "component",
                    "linked_obj_original_id": "754",
                    "linked_obj_original_type": "nomenclature",
                    "multi_count": "0",
                    "event_code": "0",
                    "user": "test_user_1",
                    "event_text": "Удалён компонент \"Test Component 1\""
                },
                {
                    "id": "691968",
                    "user_id": "9",
                    "timestamp": "1684945293",
                    "date_format": "24-05-2023 23:21:33",
                    "obj_id": "383",
                    "obj_type": "collection",
                    "linked_obj_id": "1820687",
                    "linked_obj_type": "component",
                    "linked_obj_original_id": "706",
                    "linked_obj_original_type": "nomenclature",
                    "multi_count": "0",
                    "event_code": "2",
                    "user": "test_user_2",
                    "event_text": "Изменён компонент \" Test Component 2\""
                }
            ],
            "error": 0
        }
    ],
    "error": 0
}
```


Чтение содержимого блока компонентов коллекции с учётом фильтрации

Параметр «event» запроса установлен в значение «get_components_filter».

Сервер возвращает содержимое блока компонентов коллекции с учётом фильтрации в формате JSON строки в формате аналогичном запросу «get_obj_blocks».

Для осуществления фильтрации могут быть использованы следующие параметры:

- Номенклатура, на базе которой созданы компоненты;
- Единица измерения компонента;
- Массив прикреплённых к компоненту параметров и соответствующие им значения;

Тело запроса для объекта состоит из следующих полей:

- type – тип объекта (должен быть установлен в «collection»);
- id – id коллекции;
- name – обозначение коллекции (используется вместо «id»);
- nomenclature_id – id номенклатуры, на базе которой должны быть созданы отобранные компоненты;
- nomenclature_name – обозначение номенклатуры, на базе которой должны быть созданы отобранные компоненты (используется вместо «nomenclature_id»);
- unit_id – id единицы измерения для отбора компонентов;
- unit_name – обозначение единицы измерения для отбора компонентов (используется вместо «unit_id»);
- params – массив параметров и их значений, которые должны быть прикреплены к компоненту для попадания в выборку (имеет следующий вид: [{«Параметр_1»: «Значение «},{«Параметр_2»: «Значение»},…,{«Параметр_N»: «Значение»}]);

Далее приведён запрос на чтение компонентов коллекции с учётом следующих фильтров:

- Родительская коллекция с обозначением «Тестовая коллекция»;
- Номенклатура, на базе которой созданы компоненты, должна иметь обозначение «Тестовая номенклатура»;
- Единица измерения компонента «кг»;
- Компонента должна иметь два прикреплённых параметра «Тестовый параметр 1» и «Тестовый параметр 2» со значениями «123» и «456» соответственно;

```
{
    "reqArr": [
        {
            "type": "collection",
            "name": "Тестовая коллекция",
            "nomenclature_name": "Тестовая номенклатура",
            "unit_name": "кг",
            "params": [
                {
                    "Тестовый параметр 1": "123"
                },
                {
                    "Тестовый параметр 2": "456"
                }
            ]
        }
    ],
    "event": "get_components_filter"
}
```

Редактирование объекта системы

Параметр «event» запроса установлен в значение «edit_data».

Сервер осуществляет редактирование выбранного объекта системы

Тело запроса для объекта состоит из трёх и более полей (в зависимости от количество реквизитов объекта, которые необходимо отредактировать)

- type – тип объекта;
- id – id объекта;
- name – обозначение объекта для (может использоваться только вместо «id», но не для редактирования обозначения);
- new_name – используется для редактирования обозначения позиции – поля «name», т.к. само поле «name», являясь уникальным может использоваться вместо «id» для указания на позицию для осуществления редактирования;
- <поле_объекта 1> – любое из полей объекта, доступное для редактирования;
- <поле_объекта 2> – любое из полей объекта, доступное для редактирования;
- <поле_объекта N> – любое из полей объекта, доступное для редактирования;

Далее приведён запрос для редактирования поля количества у компонента с id равным «256486», количество для этого компонента будет установлено в «123».

{"reqArr":[{"type":"component","id":"256486","amount":"123" }],"event":"edit_data"}

Удаление объекта системы

Параметр «event» запроса установлен в значение «delete_data».

Сервер осуществляет удаление выбранного объекта системы

Тело запроса для объекта состоит из двух обязательных полей:

- type – тип объекта;
- id – id объекта;
- name – обозначение объекта (используется вместо «id»);

Далее приведён запрос для удаления номенклатурной позиции с id равным «21».

```
{
    "reqArr": [
        {
            "id": "21",
            "type": "nomenclature"
        }
    ],
    "event": "delete_data"
}
```

Полное удаление объекта системы

Параметр «event» запроса установлен в значение «delete_data_complete».

Данный метод позволяет выполнить полное удаление объекта системы, не смотря на наличие содержимого в его блоках.

В настоящее время полное удаление доступно только для компонентов (тип объекта «*component*»).

Тело запроса для объекта состоит из двух полей:

- type – тип объекта;
- id – id объекта;

Компонент нельзя удалить в случае наличия дочерних компонентов, не включённых в этот же запрос на удаление. В таком случае для компонентов, дочерние компоненты у которых не включены в запрос на удаление, будет возвращена ошибка иерархии (код «11»).

Если же дочерние компоненты всех уровней находятся в одном запросе на удаление с родительским, то все они будут успешно удалены.

Также родительский компонент не получится удалить, если у текущего пользователя отсутствуют права на один из дочерних компонентов. В таком случае для родительского компонента будет возвращена ошибка прав (код «1»).

Далее приведён пример полного удаления компонентов с идентификаторами «2475835» и «2475836»:

Запрос:

```
{
    "reqArr": [
        {
            "id": "2475835",
            "type": "component"
        },
        {
            "id": "2475836",
            "type": "component"
        }
    ],
    "event": "delete_data_complete"
}
```

Ответ:

```
{
    "reqArr": [
        {
            "id": "2475835",
            "type": "component",
            "error": 0
        },
        {
            "id": "2475836",
            "type": "component",
            "error": 0
        }
    ],
    "event": "delete_data_complete",
    "error": 0
}
```

Чтение доступных статусов

Параметр «event» запроса установлен в значение «get_statuses».

Чтение доступных единиц измерения

Параметр «event» запроса установлен в значение «get_units».

Чтение доступных типов связи

Параметр «event» запроса установлен в значение «get_link_types».

Чтение дерева каталогов и категорий пользователя

Параметр «event» запроса установлен в значение «get_ref_tree».

Тело запроса для объекта состоит из одного поля:

- ref_type – тип справочника («nomenclature», «collection», «parameter», «file»);

Далее приведён ответ сервера, содержащий структуру категорий и каталогов для справочника коллекций. В описанном случае вся структура справочника коллекций состоит из категории «TEST CATEGORY» и одного содержащегося в ней каталога «FIRST FOLDER».


```
{
	"reqArr": [
		{
			"id": "27",
			"parent_id": "0",
			"type": "category",
			"name": "TEST CATEGORY",
			"category": "27"
		},
		{
			"id": "28",
			"parent_id": "27",
			"type": "folder",
			"name": "FIRST FOLDER",
			"category": "27"
		}
	],
	"event": "get_ref_tree",
	"ref_type": "collection",
	"error": 0
}
```


Запись данных

Параметр «event» запроса установлен в значение «new_data».

В ответ на запрос записи сервер отправляет JSON структуру с подтверждением, содержащую идентификаторы вновь созданных объектов.

Далее приведены запрос на создание номенклатурной позиции и ответ сервера. Создаётся номенклатурная позиция с обозначением «TEST_API_NOM_NAME» и наименованием «TEST_API_NOM_NOTATION», которая будет помещена в каталог справочника с именем «test_api_folder», находящийся в папке «Входящие» пользователя.

Запрос:


```
{
    "reqArr": [{
        "type": "nomenclature",
        "name": "TEST_API_NOM_NAME",
        "notation": "TEST_API_NOM_NOTATION",
        "folder_name": "test_api_folder"
    }],
    "event": "new_data"
}
```


Ответ:


```
{
    "reqArr": [
        {
            "type": "nomenclature",
            "name": "TEST_API_NOM_NAME",
            "notation": "TEST_API_NOM_NOTATION",
```

```
            "folder_name": "test_api_folder",
            "error": 0,
            "id": 102024
        }
    ],
    "event": "new_data",
    "error": 0
}
```


Параметр «folder_name» представляет из себя имя каталога, который будет создан (если таковой отсутствует) в каталоге, определённом для новых записей данного справочника в настройках текущего профиля. Если же в настройках текущего профиля конкретный каталог не указан, то каталог с именем, соответствующим параметру «folder_name», будет создан в папке «Входящие» данного справочника.

Также имеется возможность прямого определения каталога для создания новых позиций. Для этих целей необходимо передать в теле запроса параметр «folder_id». Параметр «folder_id» имеет более высокий приоритет, чем параметр «folder_name».

На объекты системы, создаваемые через API, по умолчанию не действует функционал подписки. Чтобы подписчики пользователя получили доступ к создаваемому объекту, необходимо при создании записи использовать поле «create_for_subs», установленное в значение «1». Работа с подпиской при создании доступна для типов объекта «nomenclature», «collection», «file», «parameter».

Для записи данных необходимо отправить данные в следующем формате:

Номенклатурная позиция:

- type – тип объекта («*nomenclature*»);
- name - обозначение новой позиции;
- notation - наименование новой позиции;
- folder_id – идентификатор каталога, в который будет помещена пользовательская ссылка на позицию;
- folder_name – наименование каталога во "Входящие", в который будет помещена пользовательская ссылка на позицию;

Коллекция:

- type – тип объекта («*collection*»);
- name – обозначение позиции;
- notation – наименование позиции;
- link_type – тип связи позиции;
- status – статус позиции;
- start_date – дата начала (формат строки: YYYY-MM-DD);
- start_time – время начала (формат строки: HH:MM:SS);
- end_date – дата окончания (формат строки: YYYY-MM-DD);
- end_time – время окончания (формат строки: HH:MM:SS);
- folder_id – идентификатор каталога, в который будет помещена пользовательская ссылка на позицию;
- folder_name - наименование каталога во "Входящие";
- priority – приоритет;

Параметр:

- type – тип объекта («*parameter*»);
- name - обозначение новой позиции;
- notation - наименование новой позиции;
- parameter_type - тип параметра. Значения "Float" или "Text";
- folder_id – идентификатор каталога, в который будет помещена пользовательская ссылка на позицию;
- folder_name - наименование каталога во "Входящие";

Файл:

- type – тип объекта («*file*»);
- form_file_name – имя поля формы, содержащего файл, соответствующий текущему создаваемому объекту системы с типом «file»;
- comment – комментарий;
- link_type – тип связи позиции;
- folder_id – идентификатор каталога, в который будет помещена пользовательская ссылка на позицию;
- folder_name - наименование каталога во "Входящие";

Данные серверу отправляются в виде формы (заголовок «Content-Type: form/multipart»), а каждый из отправляемых файлов должен быть передан как поле формы с именем, соответствующим полю «form_file_name» для текущего объекта.

Если в запросе создаётся лишь один файл, то поле «form_file_name» в JSON структуре объекта можно не передавать, т.к. будет автоматически взят первый из передаваемых в запросе файлов (полей формы, содержащих файлы).

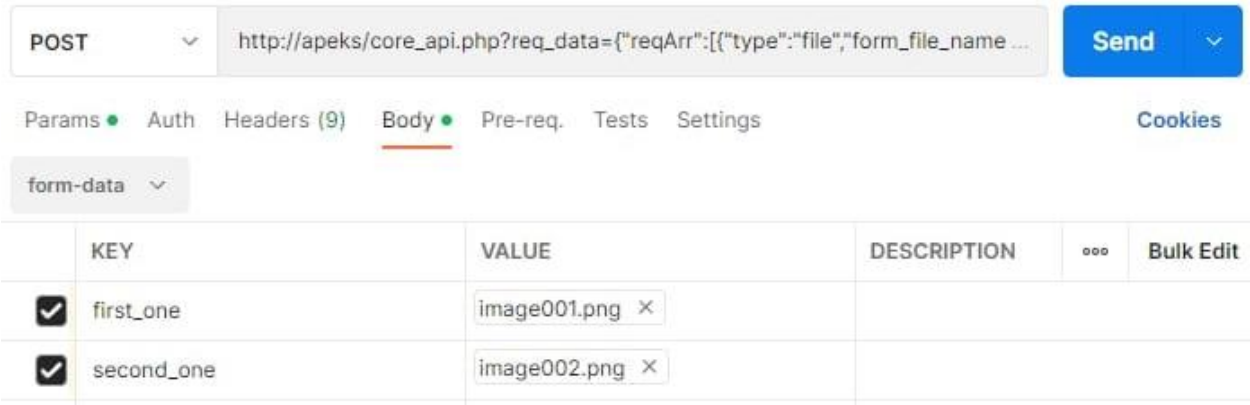
Если запрос подразумевает создание нескольких файлов в системе, а поле «form_file_name» не было передано, то к каждому из объектов будет загружен один и тот же (первый по списку) файл.

Далее приведен пример создания файла в системе:

Запрос:

```
{
    "reqArr": [
        {
            "type": "file",
            "form_file_name": "first_one",
            "link_type": 42,
            "comment": "test_comment_2",
            "folder_name": "test123"
        },
        {
            "type": "file",
            "form_file_name": "second_one",
            "link_type": 63,
            "comment": "test_comment_1",
            "folder_id": 17516
        }
    ],
    "event": "new_data"
}
```

Иллюстрация заполненных полей формы в утилите Postman:



Ответ:


```
{
    "reqArr": [
        {
            "type": "file",
            "form_file_name": "first_one",
            "link_type": 42,
            "comment": "test_comment_2",
            "folder_name": "test123",
            "error": 0,
            "id": 5253
        },
        {
            "type": "file",
            "form_file_name": "second_one",
            "link_type": 63,
            "comment": "test_comment_1",
            "folder_id": 17516,
            "error": 0,
            "id": 5254
        }
    ],
    "event": "new_data",
    "error": 0
}
```


Компонент:

- type – тип объекта («*component*»);
- nomenclature_id – идентификатор номенклатурной позиции, на базе которой будет создан компонент;
- parent_obj_id – идентификатор коллекции в которой будет создан данный компонент;
- parent_id – идентификатор родительского объекта типа «компонент» в дереве компонентов;
- link_type – тип связи позиции;
- status – статус позиции;
- amount – количество;
- unit – единица измерения;
- comment – комментарий;
- price – цена;
- start_date – дата начала (формат строки: YYYY-MM-DD);
- start_time – время начала (формат строки: HH:MM:SS);
- end_date – дата окончания (формат строки: YYYY-MM-DD);
- end_time – время окончания (формат строки: HH:MM:SS);
- priority – приоритет;

Связанный параметр:

- type – тип объекта («linked_parameter»);
- parameter_id - id параметра в системе;
- parent_obj_id - id родительского объекта (коллекции, компонента и т.д.);
- parent_obj_type - тип родительского объекта (nomenclature, collection, file, component);
- parent_id - id родительского прикреплённого параметра для организации иерархии. 0 - для помещения в корень;
- s_value - строковое значение для строковых параметров;
- d_value - числовое значение для числовых параметров;
- tolerance_plus – допуск +;
- tolerance_minus – допуск -;
- comment – комментарий;

Связанный файл

- type – тип зависимости («linked_file»);
- file_id – идентификатор файла в справочнике файлов, на базе которого был создан связанный файл;
- parent_obj_id – идентификатор объекта, к которому прикреплён данный файл;
- parent_obj_type - тип родительского объекта (nomenclature, collection, file, component);

Связанная коллекция:

- type – тип объекта (“*linked_collection”*);
- collection_id - id коллекции в системе;
- parent_obj_id - id родительского объекта (коллекции, компонента и т.д.);
- parent_obj_type - тип родительского объекта (nomenclature, collection, file, component);
- assignment_link_type – назначение позиции (тип связи);

Связанная номенклатура:

- type – тип объекта (“*linked_nomenclature”*);
- nomenclature_id - id номенклатуры в системе;
- parent_obj_id - id родительского объекта (коллекции, компонента и т.д.);
- parent_obj_type - тип родительского объекта (nomenclature, collection, file, component);
- comment – комментарий;
- link_type – тип связи позиции;

Связанная зависимость:

- type – тип объекта (“*linked_dependence”*);
- component_id - id компонента в системе;
- parent_obj_id - id родительского объекта (коллекции, компонента и т.д.);
- assignment_link_type – назначение позиции (тип связи);

Связанная ссылка:

- type – тип объекта (“linked_link*”*);
- link – текст ссылки;
- parent_obj_id - id родительского объекта (коллекции, компонента и т.д.);
- parent_obj_type - тип родительского объекта (nomenclature, collection, file, component);
- comment – комментарий.

Категория:

- type – тип объекта (“category”);
- ref_type – тип справочника (nomenclature, collection, parameter, file);
- name – имя новой категории.

Каталог:

- type – тип объекта (“folder”);
- ref_type – тип справочника (nomenclature, collection, parameter, file);
- name – имя нового каталога;
- parent_id – идентификатор родительского каталога или категории.

Клонирование позиции

Клонирование позиции позволяет создать копию справочной позиции (реквизиты и содержимое блоков) для номенклатуры и коллекции (значение поля “type”:“nomenclature”, значение поля “type”:“collection”).

Параметр «event» запроса установлен в значение «clone_data».

Тело запроса для объекта состоит из следующих полей:

- type – тип объекта;
- id – id объекта;
- folder_id – id каталога справочника для создания копии (необязательно, по умолчанию клон размещается в соответствующей папке внутри каталога «входящие», либо в соответствии с настройками в профиле пользователя);
- is_template – (только для коллекций) флаг шаблона, если установлен в «1», из исходной коллекции не копируются прикреплённые коллекции с назначением "Шаблон", а также из первой коллекции с назначением "Шаблон" берётся тип связи для новой коллекции (необязательно, по умолчанию установлен в «0»);
- suffix – суффикс, добавляемый в конец обозначения позиции вместо стандартной строки с датой и временем создания клона, может пригодиться при одновременном создании нескольких клонов одной позиции, когда суффикс с временем создания не будет делать обозначение уникальным, что помешает созданию последующих клонов (необязательно).

В ответе от сервера в теле каждого из клонируемых объектов присутствует поле «clone_id», содержащее в себе значение идентификатора созданной позиции-клона.

Далее приведён пример клонирования коллекции с идентификатором «6» с помещением её в каталог с идентификатором «89127»:

Запрос:

```
{
    "reqArr": [
        {
            "id": 6,
            "type": "collection",
            "folder_id": 89127
        }
    ],
    "event": "clone_data"
}
```

Ответ:

```
{
    "reqArr": [
        {
            "id": 6,
            "type": "collection",
            "error": 0,
            "folder_id": 89127,
            "clone_id": 278195
        }
    ],
    "event": "clone_data",
    "error": 0
}
```

Стоит учитывать, что по умолчанию блок «права» не копируется при клонировании объекта. Для клонирования с учётом блока «права» необходимо использовать параметр «clone_rights_flag», установленный в значение «1». Ниже приведён пример такого запроса.

```
{
    "event": "clone_data",
    "reqArr": [
        {
            "type": "nomenclature",
            "id": "107622"
        }
    ],
    "clone_rights_flag": 1
}
```

Замена компонентов местами

Метод позволяет поменять два компонента местами с учётом всех зависимостей.

Параметр «event» запроса установлен в значение «swap_components».

Тело запроса для объекта состоит из двух полей:

- first_component_id – идентификатор первого компонента для замены;
- second_component_id – идентификатор второго компонента для замены;

Далее приведён пример замены местами компонентов с идентификаторами «2345» и «9876»:

```
{
    "reqArr": [
        {
            "first_component_id": "2345",
            "second_component_id": "9876"
        }
    ],
    "event": "swap_components"
}
```

Чтение версии API

Параметр «event» запроса установлен в значение «get_version».

Ответ сервера содержит в себе строки с номером текущей версии API («version») и датой её релиза («date»), а также массив «version_arr», содержащий в себе целочисленные значения, представляющие собой номер версии API и размещённые в соответствующем порядке.

Далее приведён пример чтения версии API:

Запрос:

```
{
    "event": "get_version"
}
```

Ответ:

```
{
    "event": "get_version",
    "error": 0,
    "version": "1.1.38",
    "version_arr": [
        1,
        1,
        38
    ],
    "date": "12.09.2022"
}
```

Получение данных пользователя/группы

Параметр «event» запроса установлен в значение «get_user_data».

Тело запроса для пользователя состоит из поля «user» – логин пользователя/группы;

Сервер возвращает данные пользователя/группы в формате JSON строки со следующей структурой полей:

- «id» – идентификатор пользователя или группы;
- «user» – логин пользователя или группы;
- «type» – тип пользователя (группа «group» или пользователь «user»);
- «first_name» – имя (для пользователя);
- «second_name» – фамилия (для пользователя);
- «father_name» – отчество (для пользователя);
- «e_mail» – e-mail (для пользователя);
- «phone» – телефон (для пользователя);
- «full_name» – полное имя группы (для группы);

Чтобы получить данные о текущем пользователе, необходимо указать «me» в качестве логина пользователя (поле «user»). Также для получении информации только о текущем пользователе можно не передавать никакие поля кроме «event».

Получить можно лишь те реквизиты пользователя, которые были сделаны общедоступными соответствующим пользователем в настройках профиля.

Далее приведён пример получения данных пользователя «test_user».

Запрос:

```
{
    "event": "get_user_data",
    "reqArr": [
        {
            "user": "test_user"
        }
    ]
}
```

Ответ:

```
{
    "event": "get_user_data",
    "reqArr": [
```

```
        {
            "user": " test_user",
            "id": 315,
            "type": "user",
            "first_name": "Кирилл",
            "second_name": "Иванов",
            "error": 0
        }
    ],
    "error": 0
}
```

Так же имеется возможность получения данных пользователя/группы по идентификатору. В таком случае тело запроса должно содержать поля «id» и «type», соответствующие идентификатору пользователя/группы и типу (группа «group» или пользователь «user»), соответственно.

Далее приведён пример запроса на получение данных пользователя с идентификатором «315»:

```
{
    "event": "get_user_data",
    "reqArr": [
        {
            "id": 315,
            "type": "user"
        }
    ]
}
```

Получение списка групп пользователя

Параметр «event» запроса установлен в значение «get_user_groups».

Метод позволяет получить список групп, в которых находится текущий пользователь, либо любой другой пользователь системы (в случае запроса от имени администратора).

Далее приведён пример запроса на получение групп текущего пользователя.

Запрос:

```
{
    "event": "get_user_groups"
}
```

Ответ:

```
{
    "event": "get_user_groups",
    "error": 0,
    "reqArr": [
        {
            "id": 154,
            "user": "test_user",
            "groups": [
                {
                    "id": 36,
                    "name": "group_1",
                    "full_name": "group_1 full name",
                    "is_admin": 1
                },
                {
                    "id": 40,
                    "name": "group_2",
                    "full_name": "group_2 full name",
                    "is_admin": 0
                },
                {
                    "id": 41,
                    "name": "group_3",
```

```
                    "full_name": "group_3 full name",
                    "is_admin": 0
                }
            ],
            "error": 0
        }
    ]
}
```

Группы, членом которых является пользователь, перечислены в массиве «groups». Каждым элементом данного массива является группа, представленная в виде объекта с полями:

- «id» – идентификатор группы;
- «name» – имя группы;
- «full_name» – полное имя группы;
- «is_admin» – флаг наличия прав администратора у пользователя в данной групе(«1» или «0»);

Для получения списка групп конкретного пользователя (отличного от текущего) необходимо указать его имя (поле «user») или идентификатор (поле «id»).

Если имя/идентификатор пользователя отличаются от имени/идентификатора текущего пользователя, запрос необходимо выполнять с правами администратора. В противном случае будет возвращена ошибка прав.

Далее приведён пример запроса на получение групп пользователя с идентификатором «188»:

```
{
    "event": "get_user_groups",
    "reqArr": [
        {
            "id": 188
        }
    ]
}
```

Далее приведён пример запроса на получение групп пользователя с именем «test_user_006»:

```
{
    "event": "get_user_groups",
    "reqArr": [
        {
            "user": "test_user_006"
        }
    ]
}
```

Получение списка пользователей группы

Параметр «event» запроса установлен в значение «get_group_users».

Метод позволяет получить список пользователей группы, в которой текущий пользователь имеет членство, либо список пользователей любой другой группы (в случае запроса от имени администратора).

Тело запроса для объекта состоит из одного поля :

- id – идентификатор группы;

или

- name – имя группы;

Далее приведён пример запроса на получение пользователей группы test_group.

Запрос:

```
{
    "event": "get_group_users",
    "reqArr": [
        {
            "name": "test_group"
        }
    ]
}
```

Ответ:

```
{
    "event": "get_group_users",
    "reqArr": [
        {
            "name": "test_group",
            "id": "43",
            "users": [
                {
                    "id": 220,
                    "user": "test_user_1",
                    "is_admin": 1
                },
                {
                    "id": 207,
                    "user": "test_user_2",
                    "is_admin": 0
                },
                {
                    "id": 208,
                    "user": "test_user_3",
                    "is_admin": 0
                }
            ],
            "error": 0
        }
    ],
    "error": 0
}
```

Если запрос выполняется для группы, в которой текущий пользователь не состоит, а также текущий пользователь не является администратором, будет возвращена ошибка прав.

Выдача прав на объект

Для работы с правами объектов системы используется параметр запроса «event» установленный в значение «share_data». Данным способом возможно создавать, редактировать и удалять права на объект системы для пользователя или группы.

Запрос для работы с правами объекта системы может быть выполнен для следующих типов объектов системы:

- Номенклатурная позиция – значение поля «type» установлено в «nomenclature»;
- Коллекция – значение поля «type» установлено в «collection»;
- Файл – значение поля «type» установлено в «file»;
- Параметр – значение поля «type» установлено в «parameter».

Тело запроса для объекта состоит из следующих полей:

- type – тип объекта;

- id – id объекта;
- user – логин пользователя или группы;
- reader – права читателя;
- associate – права соавтора;
- creator – права автора;

Далее приведён пример выдачи прав автора на коллекцию с идентификатором «3845» для пользователя/группы «test_user»:

```
{
        "event": "share_data",
        "reqArr": [
                {
                        "type": "collection",
                        "id": 3845,
                        "user": "test_user",
                        "reader": 0,
                        "associate": 0,
                        "creator": 1
                }
        ]
}
```

Получить список пользователей, имеющих права на объект возможно, считав все блоки объекта, либо блок прав отдельно с помощью запроса с параметром «event», установленным в значение «get_obj_blocks».

Получение ссылки на объект

Метод позволяет получить строку, представляющую из себя ссылку на объект системы.

Параметр «event» запроса установлен в значение «get_data_url».

Тело запроса для объекта состоит из двух полей:

- type – тип объекта;
- id – id объекта;

В ответном сообщении сервером возвращается поле «url», которое представляет собой ссылку на объект системы.

Далее приведён пример получения ссылки для номенклатурной позиции с идентификатором «107274».

Запрос:

```
{
    "reqArr": [
        {
            "id": "107274",
            "type": "nomenclature"
        }
    ],
    "event": "get_data_url"
}
```

Ответ:

```
{
    "reqArr": [
        {
            "id": "107274",
            "type": "nomenclature",
            "url": "http://<АДРЕС СИСТЕМЫ>/#nom_params?nom_id=107274",
            "error": 0
```

```
        }
    ],
    "event": "get_data_url",
    "error": 0
}
```

Получение информации о размещении объекта

Метод позволяет получить список всех каталогов, доступных текущему пользователю, в которых расположен данный объект.

Параметр «event» запроса установлен в значение «get_data_location».

Тело запроса для объекта состоит из двух полей:

- type – тип объекта;
- id – id объекта;

В ответ на запрос сервер возвращает массив «location», содержащий в себе перечень каталогов пользователя, в которых встречается искомый объект.

Каждый элемент массива «location» описывает собой ссылку на объект в справочнике, доступную пользователю, и содержит следующие поля:

- parent_id – id родительского каталога, в котором находится ссылка на объект;
- category – id категории, в которой содержится родительский каталог;
- addr – строка, содержащая путь к объекту в справочнике;
- user_login – логин пользователя (или группы) – обладателя указанных далее прав;
- creator – права автора;
- associate – права соавтора;
- reader – права читателя;
- id – идентификатор ссылки на объект;
- ref_link_creator – пользователь, создавший ссылку на объект в справочнике с идентификатором «id»;
- user_cat – пользователь, создавший категорию с идентификатором «category».

Далее приведён пример получения сведений о размещении номенклатурной позиции с идентификатором «107310».

Запрос:

```
{
    "event": "get_data_location",
    "reqArr": [
        {
            "type": "nomenclature",
            "id": 107310
        }
    ]
}
```

Ответ:

```
{
    "event": "get_data_location",
    "reqArr": [
        {
            "type": "nomenclature",
            "id": 107310,
            "location": [
```

```
                {
                    "user_login": "test_user",
                    "creator": 1,
                    "associate": 1,
                    "reader": 1,
                    "parent_id": 876,
                    "category": 2,
                    "id": 204887,
                    "ref_link_creator": "test_user",
                    "user_cat": "test_user",
                    "addr": "category_123/test_folder/test123"
                },
                {
                    "user_login": "test_user",
                    "creator": 1,
                    "associate": 1,
                    "reader": 1,
                    "parent_id": 204849,
                    "category": 306,
                    "id": 204888,
                    "ref_link_creator": "test_user",
                    "user_cat": "test_user",
                    "addr": "category_777/"
                }
            ]
        }
    ],
    "error": 0
}
```


Проверка существования объекта

Метод позволяет получить сведения о существовании объекта в системе по его идентификатору или обозначению.

Параметр «event» запроса установлен в значение «check_data».

Запрос может быть выполнен для следующих типов объектов системы:

- Номенклатурная позиция – значение поля «type» установлено в «nomenclature»;
- Коллекция – значение поля «type» установлено в «collection»;
- Параметр – значение поля «type» установлено в «parameter».

Тело запроса для объекта состоит из следующих полей:

- type – тип объекта;
- id – id объекта;
- name – обозначение объекта (если не указан «id»).

В ответном сообщении сервером возвращается поле «exist», которое установлено в «1» в случае существования объекта в системе, либо установлено в «0» в случае его отсутствия.

Также, если объект существует, возвращаются соответствующие поля «name», «notation» и «id» объекта системы.

Далее приведён пример проверки существования в системе номенклатурной позиции с обозначением «TEST_NAME».

Запрос:

```
{
    "event": "check_data",
    "reqArr": [
        {
```

```
            "type": "nomenclature",
            "name": "TEST_NAME"
        }
    ]
}
```

Ответ:

```
{
    "event": "check_data",
    "reqArr": [
        {
            "type": "nomenclature",
            "name": "TEST_NAME",
            "exist": 1,
            "id": "107310",
            "notation": "TEST_NOTATION",
            "error": 0
        }
    ],
    "error": 0
}
```

Проверка прав пользователя на объект

Метод позволяет получить сведения о наличии/отсутствии и типе прав конкретного пользователя на конкретный объект системы.

Чтобы получить данные сведения через API, необходимо иметь как минимум права уровня «читатель» на объект.

Параметр «event» запроса установлен в значение «get_user_rights».

Запрос может быть выполнен для следующих типов объектов системы:

- Номенклатурная позиция – значение поля «type» установлено в «nomenclature»;
- Коллекция – значение поля «type» установлено в «collection»;
- Файл – значение поля «type» установлено в «file»;
- Параметр – значение поля «type» установлено в «parameter»;
- Компонент – значение поля «type» установлено в «component».

Тело запроса для объекта состоит из следующих полей:

- type – тип объекта;
- id – id объекта;
- user – логин пользователя или группы;
- full_rights – флаг выдачи расширенной информации о правах (опционально);

В ответ на запрос сервер возвращает следующие поля:

- creator – права автора;
- associate – права соавтора;
- reader – права читателя;
- group – флаг прав через группу;
- category – флаг прав через категорию;
- admin – флаг прав через учётную запись администратора.

Далее приведён пример получения прав пользователя «test_user» на коллекцию с идентификатором «74549».

Запрос:

```
{
    "event": "get_user_rights",
    "reqArr": [
        {
            "type": "collection",
            "id": 74549,
            "user": "test_user"
        }
    ]
}
```

Ответ:

```
{
    "event": "get_user_rights",
    "reqArr": [
        {
            "type": "collection",
            "id": 74549,
            "user": "test_user",
            "creator": 0,
            "associate": 0,
            "reader": 1,
            "group": 1,
            "category": 0,
            "admin": 0,
            "error": 0
        }
    ],
    "error": 0
}
```

Чтобы получить расширенную информацию о правах пользователя на объект системы, необходимо установить значение поля «full_rights» в «1» (по умолчанию установлено в «0»). В таком случае в ответ для целевого объекта будет дополнительно включено три поля, представляющие из себя объекты, содержащие значения прав, полученных путём обладания личными правами на объект («self_rights»), полученных благодаря членству в группах («group_rights»), полученных благодаря доступу к категориям («category_rights»).

Далее приведён пример запроса статуса полных прав пользователя «test_user» на коллекцию с идентификатором «120112».

Запрос:

```
{
    "event": "get_user_rights",
    "reqArr": [
        {
            "type": "collection",
            "id": 120112,
            "user": "test_user",
            "full_rights": 1
        }
    ]
}
```

Ответ:

```
{
    "event": "get_user_rights",
    "reqArr": [
        {
            "type": "collection",
            "id": 120112,
            "user": "test_user",
            "full_rights": 1,
            "creator": 0,
```

```
            "associate": 1,
            "reader": 1,
            "group": 1,
            "category": 1,
            "admin": 0,
            "error": 0,
            "self_rights": {
                "creator": 0,
                "associate": 0,
                "reader": 0
            },
            "group_rights": {
                "creator": 0,
                "associate": 1,
                "reader": 0
            },
            "category_rights": {
                "creator": 0,
                "associate": 0,
                "reader": 1
            }
        }
    ],
    "error": 0
}
```


Также метод «get_user_rights» позволяет получить информацию о наличии прав на редактирование объекта, которые могут быть обусловлены блокирующими статусами системы (например «Завершено» и «Блокировка»), перечень которых настраивается администратором в случае локальной установки системы, а также другими настройками системы.

Чтобы получить информацию о правах на редактирование необходимо установить значение поля «edit_rights» в «1» (по умолчанию установлено в «0»). В таком случае в ответе поле «edit_rights» будет содержать в себе всю информацию о правах на редактирование для указанного пользователя.

Поле «edit_rights» содержит в себе следующие поля:

- blocked_for_me – результирующий флаг запрета редактирования. Если установлен в «1», то изменение объекта недоступно для пользователя (рассчитывается на основании настроек системы и флагов, следующих далее);
- blocking_status – флаг наличия блокирующего статуса у объекта. Если установлен в «1», значит объект имеет блокирующий статус;
- edit_blocking_status_by_creator_system_flag – индикатор системного флага, позволяющего авторам редактировать объекты с блокирующими статусами. Если установлен в «1», то авторы имеют право редактировать объекты с блокирующими статусами;
- edit_completed_user_flag – персональный флаг редактирования завершённых, получаемый из настроек профиля;
- status – статус объекта;

Далее приведён пример запроса прав текущего пользователя на редактирование номенклатурной позиции с идентификатором «1527».

Запрос:


```
{
    "reqArr": [
        {
            "id": 1527,
            "type": "nomenclature",
            "edit_rights": 1
        }
    ],
```

```
    "event": "get_user_rights",
}
```

Ответ:

```
{
    "reqArr": [
        {
            "id": 1527,
            "type": "nomenclature",
            "error": 0,
            "edit_rights": {
                "blocking_status": 0,
                "status": 0,
                "edit_blocking_status_by_creator_system_flag": 1,
                "edit_completed_user_flag": 1,
                "blocked_for_me": 0
            },
            "creator": 1,
            "associate": 0,
            "reader": 0,
            "group": 0,
            "category": 0,
            "admin": 1
        }
    ],
    "event": "get_user_rights",
    "error": 0
}
```

Установка/снятие статуса администратора для текущего пользователя

Метод позволяет установить или снять статус администратора для текущего пользователя.

Метод доступен после включения соответствующего параметра в конфигурации системы.

Тело запроса состоит из одного поля:

- admin – флаг присвоения/снятия статуса администратора («1» или «0»);

Если поле «admin» установлено в значение «1», то текущему пользователю будет присвоен статус администратора, а если в «0», то статус администратора будет снят.

Далее приведён пример установки статуса администратора для текущего пользователя.

Запрос:

```
{
    "event": "set_user_admin",
    "admin": 1
}
```

Ответ:

```
{
    "event": "set_user_admin",
    "admin": 1,
    "error": 0
}
```

Пользовательские скрипты

Пользователь имеет возможность расширять функционал системы, не прибегая к сторонним приложениям, используя пользовательские скрипты, представляющие из себя файлы с расширением «.js», загруженные в систему посредством стандартного интерфейса и содержащие

в себе программный код на языке JavaScript, реализующий необходимый прикладной функционал. Тип связи для таких файлов должен быть установлен в «Скрипт»

Пользовательский скрипт должен быть реализован в виде процедуры с именем «userfunc», которая ничего не возвращает, а на вход принимает параметры «reqData» и «context». Параметр «reqData» представляет из себя объект, содержащий массив «reqArr», который в свою очередь содержит объекты системы, состоящие из двух полей: «id» и «type», где «id» – уникальный идентификатор позиции среди объектов своего типа, а «type» – тип объекта системы. Помимо указанного массива «reqData» содержит свойство «event», которое изначально имеет значение «null», а впоследствии устанавливается пользователем при передаче запроса серверу с целью выбора действия, которое сервер должен произвести над переданными объектами системы. Также «reqData» и каждый из элементов массива «reqArr», имеют поле с именем «error», содержащее код ошибки выполнения запроса сервером. Коды ошибок можно найти в приложении к данному руководству.

Параметр «context» представляет из себя объект, содержащий в себе информацию о контексте вызова скрипта, а именно, с какой страницы системы он был вызван (поле «page»), а также, для случая вызова скрипта со страниц, инициализация которых связана с каким-либо объектом (объектами), - массив соответствующих объектов («items»). Если скрипт вызывается со страницы, которую невозможно однозначно сопоставить с каким-либо объектом системы, то массив «items» передаётся в пользовательский скрипт пустым.

Возможные значения поля «page» параметра «context»:

- cat_nom_edit – страница справочника номенклатуры;
- collections – страница справочника коллекций;
- parameters – страница справочника параметров;
- ref_files – страница справочника файлов;
- nom_params – страница номенклатурной позиции;
- collection_params – страница коллекции;
- param_params – страница параметра;
- file_params – страница файла;
- orders – страница режима «Заказы»;
- schedule – страница режима «График»;
- execution – страница режима «Выполнение»;
- show_tasks – страница обзора заданий по выделенным компонентам;
- parties – страница режима «Партии».

Далее приводится JSON представление параметра «context» для случая вызова скрипта со страницы коллекции с идентификатором «383».

```
{
    "items": [
        {
            "id": 383,
            "type": "collection"
        }
    ],
    "page": "collection_params"
}
```

Далее приводится JSON представление параметра «context» для случая вызова скрипта со страницы обзора заданий по выделенным компонентам, идентификаторы которых представляют из себя список «584081, 584084, 586323».

```
{
    "items": [
```

```
        {
            "id": 584081,
            "type": "component"
        },
        {
            "id": 584084,
            "type": "component"
        },
        {
            "id": 586323,
            "type": "component"
        }
    ],
    "page": "show_tasks"
}
```

Далее приводится JSON представление параметра «context» для случая вызова скрипта со страницы режима «График».

```
{
    "items": [],
    "page": "schedule"
}
```

Ниже приведён пример реализации пользовательского скрипта на базе функции «userfunc», которая выводит в браузере текстовое сообщение, содержащее данные о позициях, к которым применяется скрипт в формате JSON.

```
function userfunc(reqData, context) {
    alert(JSON.stringify(reqData));
}
```

Данный пользовательский скрипт выведет в браузере сообщение, содержащее объект «reqData», для объекта(ов) системы, к которому(ым) данный пользовательский скрипт был применён. Ниже приведён пример выводимого таким скриптом сообщения:

```
{
    "reqArr": [
        {
            "id": "21",
            "type": "nomenclature",
            "error": "0"
        }
    ],
    "event": null,
    "error": 0
}
```

Объект «reqData» предназначен для предоставления пользовательскому скрипту информации о объекте или объектах, к которым он должен быть применён. Также возможна последующая передача данного объекта серверу для выполнения каких-либо действий с объектами системы.

Все необходимые проверки на соответствие/несоответствие входных данных алгоритму пользовательского скрипта должны быть произведены пользователем непосредственно в реализации скрипта.

Запрос серверу

Запрос серверу можно осуществить посредством функции «apiDoRequest». Функция имеет один входной параметр «reqData», формат которого аналогичен формату данных, передающихся пользовательскому скрипту системой, рассмотренному выше. Данная особенность позволяет без лишних сложностей передавать данные из системы на сервер, попутно внося в них необходимые изменения. Функция возвращает модифицированную переменную «reqData», массив «reqArr»

внутри которой содержит уже не только поля «id» и «type», но и прочие поля, присущие для данного типа объекта системы.

Запрос на чтение данных объекта системы:

Для того, чтобы запросить у системы поля одного из объектов, необходимо вызвать функцию «apiDoRequest», передав ей в виде параметра переменную «reqData» с полем «event» равным «get_data». Ниже приведён пример переменной «reqData» для запроса полей объекта системы типа «номенклатура» с идентификатором «21».

```
{
    "reqArr": [
        {
            "id": "21",
            "type": "nomenclature",
            "error": "0"
        }
    ],
    "event": "get_data",
    "error": 0
}
```

Массив «reqArr» может содержать неограниченное количество объектов системы разного типа, для которых необходимо осуществить выдачу информации. Пример запроса для нескольких объектов системы указан ниже.

```
{
    "reqArr": [
        {
            "id": "21",
            "type": "nomenclature",
            "error": "0"
        },
        {
            "id": "106",
            "type": "collection",
            "error": "0"
        },
        {
            "id": "49",
            "type": "nomenclature",
            "error": "0"
        },
        {
            "id": "745",
            "type": "file",
            "error": "0"
        }
    ],
    "event": "get_data",
    "error": 0
}
```

Поле «error» в отправляемых данных не играет роли и требует обработки лишь при получении ответа от сервера (сервер отвечает в этом же формате, но добавляет к содержимому элементов массива “reqArr”, помимо полей «id» и «уре», прочие поля, свойственные данному типу объекта системы). Далее приведён пример ответа от сервера на запрос, указанный выше.

```
{
    "reqArr": [
        {
            "id": "21",
            "type": "nomenclature",
            "name": "TEST POSITION #1",
            "notation": "NOTATION OF TEST POSITION #1",
            "created_date": "2017-05-28 04:00:09",
            "last_change_date": "2017-05-28 04:00:09",
```

```
            "error": "0"
        }
    ],
    "event": "get_data",
    "error": "0"
}
```

Процедура пользовательского скрипта, осуществляющего вывод данной JSON строки в виде сообщения браузера выглядит следующим образом:

```
function userfunc(reqData)
{
    reqData.event = "get_data";
    alert(JSON.stringify(apiDoRequest(reqData)));
}
```

Для вызова одного пользовательского скрипта внутри другого пользовательского скрипта можно воспользоваться функцией «apiLoadScript», принимающей три параметра: «scriptFileId», «reqData» и «context». Содержимое, передаваемое с параметром reqData, аналогично параметру функции «userfunc», параметр «scriptFileId» принимает внутрисистемный справочный идентификатор файла, представляющего собой пользовательский скрипт («id» объект типа «file» со связью «Скрипт»), параметр «context» является необязательным и работает аналогично одноимённому параметру функции «userfunc», предоставляя пробросить контекст выполнения родительского скрипта в дочерний, либо подменить контекст выполнения дочернего скрипта при необходимости. Ниже приведён пример проброса контекста из родительского скрипта в дочерний с идентификатором «5687» (дочерний скрипт выводит в консоль родительский контекст).

Родительский скрипт:

```
function userfunc(reqData, context)
{
    apiLoadScript(5687, reqData, context);
}
```

Дочерний скрипт:

```
function userfunc(reqData, context)
{
    console.log(context);
}
```

Коды ошибок

0 – Ошибки отсутствуют;

1 – Ошибка прав;

2 – Ошибка получения данных;

3 – Ошибка данных запроса;

4 – Ошибка создания объекта;

5 – Ошибка авторизации;

6 – Ошибка подписки (неприменимо к локальной конфигурации системы);

7 – Ошибка ограничений пользователя;

8 – Ошибка удаления (запись не пуста);

9 – Системная ошибка;

10 – Ошибка обновления данных;

11 – Ошибка иерархии;

12 – Действие запрещено настройками системы;

13 – Ничего не найдено (объект не существует).

Приложение

Далее указано несколько дополнительных инструментов, которые могут оказаться полезны при разработке приложений на базе АПЕКС API.

Получение изображения QR кода со ссылкой на объект системы

Для того, чтобы получить изображение с расширением «png», представляющее из себя QR код с ссылкой на объект системы, необходимо отправить запрос следующего вида:

http://<*АДРЕС СИСТЕМЫ*>/getqr.php?obj_type=<*тип объекта системы*>&obj_id=<*идентификатор объекта системы*>&size=<*размер изображения (1-10)*>

Чтобы сгенерировать QR код на основе пользовательской строки, необходимо использовать параметр «text»:

http://<*АДРЕС СИСТЕМЫ*>/getqr.php?text=<*пользовательская строка*>&size=<*размер изображения (1-10)*>

Получение изображения штрихкода формата PDF417 со ссылкой на объект системы

Для того, чтобы получить изображение с расширением «png», представляющее из себя штрихкод формата PDF417 со ссылкой на объект системы, необходимо отправить запрос следующего вида:

http://<*АДРЕС СИСТЕМЫ*>/getpdf417.php?obj_type=<*тип объекта системы*>&obj_id=<*идентификатор объекта системы*>&size=<*размер изображения (1-20)*>&padding=<*размер белого поля вокруг изображения в пикселях (0-50*)>&ratio=<вертикальный размер одной ячейки штрихкода (1-10)>

Получение изображение аватара пользователя

Для того, чтобы получить изображение, представляющее из себя аватар пользователя, необходимо отправить запрос следующего вида:

http://<*АДРЕС СИСТЕМЫ*>/userpic.php?login=<*логин пользователя*>

Скачивание файла из системы

Далее приведены варианты запросов для скачивания файла из системы.

Для загрузки по id справочника:

http://<*АДРЕС СИСТЕМЫ*>/getfile.php?**f_id_r**=<*id файла в справочнике*>

Для загрузки по id прикреплённого файла:

http://<*АДРЕС СИСТЕМЫ*>/getfile.php?**f_id_l**=<*id прикреплённого к объекту файла*>

Для загрузки эскиза объекта системы (выгружается файл из блока файлов объекта, имеющий тип связи «эскиз»):

http://<*АДРЕС СИСТЕМЫ*>/getfile.php?obj_id=<*идентификатор объекта системы*>&obj_type=<*тип объекта системы*>