ТЕЗИС

# Руководство по API

Версия 5.22

г. Самара, 2024

# Оглавление

# Введение

## О системе

СЭД ТЕЗИС – современная российская система электронного документооборота, которая подходит для компаний любого размера и отрасли, коммерческих и государственных организаций.

Внедрение СЭД ТЕЗИС помогает сделать работу удобнее и прозрачнее, ускорить документооборот и бизнес-процессы, систематизировать хранение документов.

Пользователям СЭД ТЕЗИС дает возможность:

- комфортно организовать управление задачами: постановку и контроль исполнения, получение и отчёт о ходе работы;
- упорядочить совместную работу с документами: подготовку текста, согласование, утверждение, ознакомление;
- легко находить задачи, документы, другую информацию по тексту или при помощи фильтров;
- работать с вложениями различных форматов: добавлять, следить за изменением версий, скачивать на компьютер;
- своевременно получать уведомления о необходимости выполнения действий и других важных событиях;
- оптимизировать работу канцелярии: ведение номенклатуры дел, регистрацию документов и отслеживание их движения;
- планировать работу при помощи календаря;
- визуализировать информацию в форме диаграмм для оценки ситуации и подготовки отчётов.

СЭД ТЕЗИС – кроссплатформенное решение, совместимое с широким спектром операционных систем, браузеров, офисных пакетов. Это значит, что при внедрении не требуется менять ИТ-инфраструктуру компании и переустанавливать ПО на рабочих местах пользователей.

Вход в Систему осуществляется через браузер, поэтому работать можно на любом устройстве – в офисе, дома, в командировке. Также пользователи могут работать в Системе с помощью современного мобильного приложения даже при отсутствии Интернет-соединения.

Для максимально комфортной работы можно самостоятельно настроить внешний вид СЭД ТЕЗИС – установить фото профиля и изображение на Основном экране, отображение списков и индивидуальные папки поиска.

## Мобильное приложение СЭД ТЕЗИС

Мобильное приложение системы ТЕЗИС позволяет решать следующие задачи:

- ставить задачи сотрудникам и осуществлять контроль над их выполнением;
- работать с документами и договорами Системы: создавать их, согласовывать, утверждать, накладывать резолюции и обрабатывать их, проводить ознакомление и обсуждение любых вопросов, связанных с документами и договорами;
- организовывать совещания: приглашать участников, назначать докладчиков, оформлять повестки и проводить их согласование;
- работать как со стандартными процессами системы, так и с процессами, созданными в Дизайнере процессов;
- получать уведомления о необходимости выполнить действие в Системе и информационные уведомления об изменениях в документах;
- работать с задачами и документами в офлайн-режиме, находясь без доступа к сети Интернет.

Для максимально комфортной работы можно самостоятельно настроить внешний вид СЭД ТЕЗИС – установить фото профиля и изображение на Основном экране, отображение списков и индивидуальные папки поиска.

## Правила использования

Данный документ является описанием взаимодействия между мобильным клиентом под управлением ОС IOs/Android и сервером ТЕЗИС.

Взаимодействие построено на выполнении REST-запросов со стороны мобильного клиента.

Раздел 1 содержит общую информацию.

В разделе 2 представлены данные по запросу информации о сервере.

В разделе 3 описана аутентификация.

В разделе 4 рассмотрена push-аутентификация.

В разделе 5 находится информация о синхронизации данных.

Раздел 6 описывает работу с карточками.

В разделе 7 описана работа с вложениями.

В разделе 8 описана работа с файлами вложений.

В разделе 9 рассмотрено выполнение процессных действий.

В разделе 10 представлены данные по замещению.

В разделе 11 содержит информацию по изменению аватара.

В разделе 12 описана вкладка «Обсуждения».

В разделе 13 содержит глобальный поиск карточек.

В разделе 14 представлены данные по загрузке карточек в папках поиска.

В разделе 15 описано удаление уведомлений.

В разделе 16 подробно рассмотрены используемые форматы.

Термины и сокращения, используемые в данном документе, указаны в конце документа.

# 1. Общая информация

## 1.1. Заголовки запросов

По умолчанию в заголовках всех запросов присутствуют:

```
“Accept”: “application/json”
“Accept-language”: “<locale>”
```

## 1.2. Доп. заголовки в авторизованных запросах

Для авторизованных запросов дополнительно присутствуют:

```
GET
“Authorization”: “Bearer <token>”
```

```
POST
“Authorization”: “Bearer <token>”
“Content-Type”: “application/json”
```

```
PATCH
“Authorization”: “Bearer <token>”
“Content-Type”: “application/json”
```

```
DELETE
“Authorization”: “Bearer <token>”
```

## 1.3. Возможные статусы ошибок

В ответе на любой запрос кроме базовых ошибок, таких как «404» и «500», может прийти ошибка со следующим статусом:

- «401» – пользовательская сессия истекла;
- «402» – превышено максимальное количество активных пользователей.

# 2. Информация о сервере

После ввода адреса сервера в МП выполняется запрос информации о сервере.

Кроме проверки корректности введённого адреса пользователем и определении версии сервера, данный запрос является своего рода ping-функцией перед очередной синхронизацией данных. Запрос не ресурсоёмкий и время ответа на него было ограничено 5 секундами.

Запрос:

```
GET: "/server-info"
```

**Важно!**

Данный запрос защищён basic-авторизацией, поэтому в заголовках запроса необходимо дополнительно указать:

```
"Authorization": "Basic bW9iaWxlQ2xpZW50OnNlY3JldA=="
```

Тело ответа:

```
{
api_version: string
}
```

Описание параметров:

- «**api_version**» – версия сервера (например, «5.22»).

  На основе этого параметра МП определяет какое API доступно для выполнения запросов, а какое ещё не реализовано в используемой версии сервера.

# 3. Аутентификация

После определения доступности сервера необходимо выполнить авторизацию, результатом которой будет токен, необходимый для дальнейшей работы с приложением.

## 3.1. Авторизация

Для авторизации используется протокол OAuth 2.0 с типом «Resource Owner Password Credentials Grant».

Авторизоваться можно двумя способами, указав в запросе логин/пароль или QR-код.

**Важно!**

При авторизации может прийти ошибка со следующим статусом:

- «400» – неверный логин или пароль;
- «403» – у вас нет доступа к мобильной версии.

### 3.1.1. Авторизация логином/паролем

Запрос:

```
POST: "/oauth/token"
```

**Важно!**

Данный запрос защищён basic-авторизацией, поэтому в заголовках запроса необходимо дополнительно указать:

*"Authorization": "Basic bW9iaWxlQ2xpZW50OnNIY3JIdA=="*

Так же в заголовках должен присутствовать:

*"Content-Type": "application/x-www-form-urlencoded"*

Тело запроса:

*"username=<encoded_login>&password=<encoded_password>&grant_type=password"*

Описание параметров:

- «**<encoded_login>**» – закодированный логин пользователя;
- «**<encoded_password>**» – закодированный пароль пользователя.

Тело ответа:

```
{
      access_token: string,
      token_type: string,
      expires_in: number,
      scope: string,
}
```

Описание параметров:

- «**access_token**» – токен, который затем используется в качестве параметра авторизации путём проставления в заголовки запросов;
- «**token_type**» – тип токена;
- «**expires_in**» – время жизни токена доступа в секундах (определяется на сервере через property «*thesis.mobile-rest.client.tokenExpirationTimeSec*»);
- «**scope**» – область действия токена доступа (всегда будет принимать значение «rest-api»);

### 3.1.2. Авторизация QR-кодом

Запрос:

```
POST: "/oauth/token"
```

**Важно!**

Данный запрос защищён basic-авторизацией, поэтому в заголовках запроса необходимо дополнительно указать:

```
"Authorization": "Basic bW9iaWxlQ2xpZW50OnNIY3JIdA=="
```

Так же в заголовках должен присутствовать:

```
"Content-Type": "application/x-www-form-urlencoded"
```

Тело запроса:

```
"code=<encoded_code>&grant_type=authorization_code"
```

Описание параметров:

- «**<encoded_code>**» – закодированный QR-код, считанный с камеры.

Тело ответа:

```
{
      access_token: string,
      token_type: string,
      expires_in: number,
      scope: string,
}
```

Описание параметров:

- «**access_token**» – токен доступа, который затем используется в качестве параметра авторизации путём проставления в заголовки запросов;
- «**token_type**» – тип токена доступа (всегда принимает значение «bearer»);
- «**expires_in**» – время жизни токена доступа в секундах (определяется на сервере через property «*thesis.mobile-rest.client.tokenExpirationTimeSec*»);

- «**scope**» – область действия токена доступа (всегда будет принимать значение «rest-api»).

## 3.2. Отзыв токена

Запрос:

```
POST "/oauth/revoke"
```

**Важно!**

Данный запрос защищён basic-авторизацией, поэтому в заголовках запроса необходимо дополнительно указать:

```
"Authorization": "Basic bW9iaWxlQ2xpZW50OnNIY3JIdA=="
```

Так же в заголовках должен присутствовать:

```
"Content-Type": "application/x-www-form-urlencoded"
```

Тело запроса:

```
"token=<token>"
```

Описание параметров:

- «**<token>**» – токен, который был получен при авторизации.

Тело ответа:

Тело ответа является пустым.

# 4. Push-аутентификация

В МП доступны push-уведомления.

По умолчанию они включены в настройках приложения.

В процессе работы с приложением можно включать/выключать настройку, тем самым выполняя регистрацию и отзыв токена соответственно.

По умолчанию push-уведомления включены.

После авторизации выполняется регистрация токена уведомлений (не путать с токеном авторизации приложения).

## 4.1. Регистрация

Под регистрацией подразумевается отправка сгенерированного токена уведомлений на сервер.

В приложении вызывается функция из библиотеки, отвечающей за работу с push-уведомлениями, которая возвращает токен и далее выполняется запрос с отправкой этого токена.

Запрос:

```
POST "/devices"
```

**Важно!**

Данный запрос защищён bearer-авторизацией, поэтому в заголовках запроса необходимо дополнительно указать:

```
"Authorization": "Bearer <token>"
```

Так же в заголовках должен присутствовать:

```
"Content-Type": "application/json"
```

Тело запроса:

```
{
    imei: string,
    device_id: string,
    os_id: string,
```

```
        os_version: string,
        app_id: string,
        app_version: string,
        type: string,
        notification_token: string
}
```

Описание параметров:

- «**imei**» – IMEI устройства (может быть «null»);
- «**os_id**» – имя ОС устройства;
- «**os_version**» – версия ОС устройства;
- «**app_id**» – идентификатор приложения;
- «**app_version**» – версия приложения;
- «**type**» – тип платформы (принимает значения «IOS» или «ANDROID»);
- «**notification_token**» – сгенерированный токен уведомлений.

Тело ответа:

Тело ответа является пустым.

***Примечание:***

*После успешной регистрации, push-уведомления попадают на обработку через callback-функцию, пока не будет выполнен отзыв токена уведомлений.*

## 4.2. Отзыв токена

Для прекращения получения push-уведомлений выполняется отзыв токена.

Запрос:

```
POST "/devices/unlink"
```

**Важно!**

Данный запрос защищён basic-авторизацией, поэтому в заголовках запроса необходимо дополнительно указать:

```
"Authorization": "Basic bW9iaWxlQ2xpZW50OnNIY3JIdA=="
```

Так же в заголовках должен присутствовать:

```
"Content-Type": "application/json"
```

Тело запроса:

```
{
      imei: string,
      device_id: string,
      os_id: string,
      os_version: string,
      app_id: string,
      app_version: string,
      type: string,
      notification_token: string
}
```

Описание параметров:

- «**imei**» – IMEI устройства (может быть «null»);
- «**os_id**» – имя ОС устройства;
- «**os_version**» – версия ОС устройства;
- «**app_id**» – идентификатор приложения;
- «**app_version**» – версия приложения;
- «**type**» – тип платформы (принимает значения «IOS» или «ANDROID»);
- «**notification_token**» – сгенерированный токен уведомлений.

Тело ответа:

Тело ответа является пустым.

# 5. Синхронизация данных

Синхронизация данных представляет собой набор последовательно вызывающихся функций, загружающих данные с сервера, и выполняется после:

- успешной авторизации и далее каждые 3 минуты;
- появления связи с сервером, если синхронизация не выполнялась более 3 минут.

Так как доступность сервера определяется по успешности выполнения последнего запроса, то если даже МП имеет статус «онлайн», он может быть не актуальным. Поэтому в самом начале выполняется актуализация сервера путем запроса информации о нем (см. раздел 2).

Таким образом при подтверждении наличия связи с сервером начинается выполнение синхронизации данных, состоящее из следующих этапов:

1. Выгрузка данных.
2. Пересоздание сессии.
3. Загрузка данных.

Описание каждого из этапа представлено ниже.

## 5.1. Выгрузка данных

При выгрузке данных выполняется отправка действий, которые были выполнены пользователем в офлайн-режиме.

В текущей коробочной реализации это:

- удаление уведомлений (см. раздел 15);
- действия по карточкам, а именно:
  - → работа с карточками (см. раздел 6);
  - → работа с вложениями (см. раздел 7);
  - → выполнение процессных действий (см. раздел 9).

## 5.2. Пересоздание сессии

При каждой синхронизации необходимо вызывать пересоздание пользовательской сессии на сервере.

Запрос:

```
DELETE "/current-user/user-session"
```

Тело ответа:

Тело ответа является пустым.

## 5.3. Загрузка данных

Для загрузки данных, выполняется синхронизация в следующей последовательности:

1. Загрузка настроек сервера.
2. Загрузка текущего пользователя.
3. Загрузка замещаемых пользователей.
4. Загрузка метаданных типов карточек.
5. Загрузка метаданных видов карточек.
6. Загрузка метаданных процессов.
7. Загрузка метаданных типов вложений.
8. Загрузка справочников.
9. Загрузка сотрудников.
10. Загрузка пользователей.
11. Загрузка типовых резолюций.
12. Загрузка уведомлений.
13. Загрузка папок действий.
14. Загрузка папок поиска.
15. Загрузка карточек.
16. Загрузка связанных карточек.
17. Загрузка файлов вложений.

Параметры запросов и ответов, представлены ниже.

Обязательные для заполнения параметр отмечены *.

### 5.3.1. Загрузка настроек сервера

Запрос:

```
GET "/server-settings"
```

Тело ответа:

```
{
    max_upload_size_mb: number,
    request_timeout: number,
    file_request_timeout: number,
}
```

Описание параметров:

- «**max_upload_size_mb**» – максимально допустимый размер загружаемого файла, добавляемого во вложения карточки;
- «**request_timeout**» – таймаут выполнения REST-запросов;
- «**file_request_timeout**» – таймаут запроса на загрузку/выгрузку файлов.

Таймаут любого REST примерно 1,5 минуты.

Даже если выставить «request_timeout» на большее время, то таймаут всё равно сработает примерно через 1,5 минуты.

Параметр дает возможность выставлять более короткий срок (например, 1 мин) для лучшей отзывчивости приложения (более быстрому определению онлайн/офлайн).

На данный момент этот параметр не используется в МП.

### 5.3.2. Загрузка текущего пользователя

Текущий пользователь в МП представляет собой данные о пользователе из списка пользователей, а также данные о доступных ему для создания типов карточек.

Фактически в МП используется только ID пользователя и информация о доступных для создания типов карточек, поэтому в описании тела ответа будут указаны только эти параметры.

Запрос:

```
GET "/current-user"
```

Тело ответа:

```
{
    id: string,
```

```
        current_user_info: {
            available_to_create_types: string [],
}
}
```

Описание параметров:

- «**id**» – ID пользователя из справочника пользователей;
- «**current_user_info**» – дополнительная информация для текущего пользователя;
- «**available_to_create_types**» – массив имен сущностей («entity_name») типов карточек доступных для создания текущему пользователю из МП.

### 5.3.3. Загрузка замещаемых пользователей

В списке замещаемых пользователей содержится информация о пользователях, которых можно на данный момент заместить.

Запрос:

```
GET "/substitutions"
```

Тело ответа:

```
[
{
        id: string,
        substituted_user_id: string,
        start_date: string,
        end_date: string,
},
…
]
```

Описание параметров:

- «**id**» – ID объекта;
- «**substituted_user_id**» – ID пользователя из справочника пользователей доступного для замещения;
- «**start_date**» – дата начала доступности замещения;
- «**end_date**» – дата окончания доступности замещения.

### 5.3.4. Загрузка метаданных типов карточек

Все карточки в Системе делятся на типы («Задача», «Документ», «Договор» и т.д.), которые в свою очередь описываются метаданными типов и определяют какие поля должны отображаться в карточке и какие ограничения на них накладываются.

Запрос:

```
GET "/metadata/card/types"
```

Параметры запроса:

- «**offset**» – с какого элемента по счету возвращать массив в ответе;
- «**max_results**» – какой максимальный длины может быть массив в ответе;
- «**last_change_date**» – время последней синхронизации по данному запросу.

Тело ответа:

```
[
{
      id: string,
      name: string,
      entity_name: string,
      available_to_create: boolean,
      card_kinds_supported: boolean,
      properties: MetaDataTypePropertyResponse [],
      delete_ts: Nullable<string>,
},
…
]
```

Описание параметров:

- «**id**» – ID метаданных типа карточки;
- «**name**» – локализованное название типа карточки;
- «**entity_name**» – имя сущности (например, «tm$Task»);
- «**available_to_create**» – доступность создания карточки с данным типом в МП;
- «**card_kinds_supported**» – у карточки кроме типа может быть так же вид, дополнительно накладывающий ограничения видимости и обязательности на поля, которые указаны в типе.

  Если данный параметр равен «true», то у карточки обязательно имеется вид.
- «**properties**» – метаинформация о формате полей карточек данного типа.

  Описание формата «MetaDataTypePropertyResponse» представлено в п.п. 16.29.

- «**delete_ts**» – время удаления, если по какой-то причине объект был удалён в Системе.

### 5.3.5. Загрузка метаданных видов карточек

Метаданные видов карточек можно назвать подтипами метаданных типов (для типа документ видами являются «Письмо», «Инструкция» и т.п.), дополнительно накладывающих ограничения видимости и обязательности на поля, которые указаны в типе.

Запрос:

```
GET "/metadata/card/kinds"
```

Параметры запроса:

- «**offset**» – с какого элемента по счету возвращать массив в ответе;
- «**max_results**» – какой максимальный длины может быть массив в ответе;
- «**last_change_date**» – время последней синхронизации по данному запросу.

Тело ответа:

```
[
{
      id: string,
      name: string,
      type_id: string,
      available_to_create: boolean,
      process_actors_adding_disabled: boolean,
      properties: MetaDataKindPropertyResponse [],
      delete_ts: Nullable<string>,
},
…
]
```

Описание параметров:

- «**id**» – ID метаданных вида карточки;
- «**name**» – локализованное название вида карточки;
- «**type_id**» – ID метаданных типа карточки, к которому относятся метаданные вида;
- «**available_to_create**» – доступность создания карточки с данным видом в МП;
- «**process_actors_adding_disabled**» – возможность изменения ролей пользователей в процессной форме для карточек данного вида;

- «**properties**» – метаинформация о формате полей карточек данного вида.

  Описание формата «MetaDataKindPropertyResponse» представлено в п.п. 16.28.

- «**delete_ts**» – время удаления, если по какой-то причине объект был удалён в Системе.

### 5.3.6. Загрузка метаданных процессов

Метаданные процессов определяют:

- какие процессы доступны в МП;
- какие действия можно выполнять по карточкам;
- какие параметры в той или иной процессной форме необходимо показать.

Связь метаданных процессов и карточек следующая:

- если по карточке ни один процесс не запущен – в параметре «process_ids_available_to_start» будет храниться массив ID процессов доступных для запуска;
- если по карточке запущен процесс – в параметрах:
  - → «process_id» – указывается какой процесс запущен;
  - → «state» – указывается в каком процессном состоянии она находится;
  - → «actions» – указаны все процессные действия, которые доступны по карточке в текущем процессном состоянии.

Запрос:

```
GET "/metadata/processes"
```

Параметры запроса:

- «**offset**» – с какого элемента по счету возвращать массив в ответе;
- «**max_results**» – какой максимальный длины может быть массив в ответе;
- «**last_change_date**» – время последней синхронизации по данному запросу.

Тело ответа:

```
[
{
    id: string,
    process_roles: ProcessRoleResponse[],
    start_transition: Nullable<ProcessesTransitionResponse>,
    states: ProcessStateResponse[],
    custom_transitions: Nullable<ProcessTransitionResponse[]>,
```

```
        cancel_transition: Nullable<ProcessTransitionResponse>,
        duration_enabled: boolean,
        order_enabled: boolean,
        available_for_mobile_client: boolean,
        delete_ts: Nullable<string>,
    },
    …
    ]
```

Описание параметров:

- «**id**»* – ID метаданных процесса.
- «**process_roles**»* – информация о процессных ролях доступных в данном процессе.

  Описание формата «ProcessRoleResponse» представлено в п.п. 16.33.
- «**start_transition**» – описание действия запуска процесса.

  Описание формата «ProcessTransitionResponse» представлено в п.п. 16.35.
- «**states**»* – информация о возможных состояниях в процессе, где в каждом состоянии находится массив объектов описаний процессных действий доступных в этом состоянии;
- «**custom_transitions**»* – описания дополнительных специфических действий, не описанных в процессе и не привязанных к состоянию в процессе.

  Описание формата «ProcessTransitionResponse» представлено в п.п. 16.35.
- «**cancel_transition**» – описание действия отмены процесса.

  Описание формата «ProcessTransitionResponse» представлено в п.п. 16.35.
- «**duration_enabled**» – возможность видимости длительности выполнения для ролей процесса на вкладке «Роли» в карточке;
- «**order_enabled**» – возможность видимости очерёдности назначенных пользователей на роли процесса на вкладке «Роли» в карточке;
- «**available_for_mobile_client**»* – доступность процесса в МП;
- «**delete_ts**»* – время удаления, если по какой-то причине объект был удалён в Системе.

### 5.3.7. Загрузка метаданных типов вложений

Каждое вложение в карточке, состоит из самого файла и сущности, содержащей информацию об этом файле, которая заполняется при добавлении вложения и далее отображается на вкладке «Вложения».

Каждому вложению присваивается тип вложения, который можно изменить (по умолчанию это «Вложение»).

Метаданные типов вложений описывают эти типы, а их загрузка выполняется по запросу.

Запрос:

```
GET "/attachment-types"
```

Тело ответа:

```
[
{
      id: string,
      name: string,
      is_default: boolean,
      code: string,
      available_to_create: boolean,
      delete_ts: Nullable<string>,
},
…
]
```

Описание параметров:

- «**id**»* – ID типа вложения;
- «**name**»* – локализованное название типа вложения;
- «**is_default**»* – тип, который должен быть проставлен по умолчанию при добавлении нового вложения;
- «**code**»* – код типа вложения;
- «**available_to_create**»* – доступность данного типа вложения для выбора;
- «**delete_ts**»* – время удаления, если по какой-то причине объект был удалён в Системе.

## 5.3.8. Загрузка справочников

В Системе имеется много различных справочников, тиках как «Организация», «Приоритет», «Сотрудники» и т.д., значения которых могут выступать в роли значений полей карточек или процессных форм.

Для того, чтобы можно было офлайн использовать значения справочников (например, при редактировании карточки) необходимо загрузить их на устройство.

Каждый справочник имеет свою структуру, поэтому для каждого справочника необходимо реализовывать свой запрос и заводить свою таблицу.

Однако в большинстве случаев в МП достаточно только наименование значений и их идентификаторы, поэтому для таких справочников имеется два общих запроса. По первому загружается метаинформация по всем доступным для загрузки справочникам. По второму запросу загружаются значения справочников.

## 5.3.9. Загрузка метаданных справочников

Загрузка метаданных справочников выполняется по запросу.

Запрос:

```
GET "/metadata/references"
```

Тело ответа:

```
[
{
      id: string,
      name: string,
      entity_name: string,
},
…
]
```

Описание параметров:

- «**id**»* – ID справочника;
- «**name**»* – локализованное название справочника;
- «**entity_name**»* – имя сущности (например, «tm$Priority»).

## 5.3.10. Загрузка значений справочников

После загрузки метаданных справочников выполняется загрузка значений каждого справочника.

Запрос:

```
GET "/referencies/<reference_id>"
```

Параметры URL:

- «**reference_id**» – ID справочника.

Параметры запроса:

- «**offset**» – с какого элемента по счету возвращать массив в ответе;
- «**max_results**» – какой максимальный длины может быть массив в ответе;
- «**last_change_date**» – время последней синхронизации по данному запросу.

Тело ответа:

```
[
{
      id: string,
      value: string,
      delete_ts: Nullable<string>,
},
…
]
```

Описание параметров:

- «**id**»* – ID значения справочника;
- «**value**»* – локализованное название значения справочника;
- «**delete_ts**»* – время удаления, если по какой-то причине объект был удалён в Системе.

## 5.3.11. Загрузка сотрудников

Наименований значений справочника сотрудников недостаточно в МП, поэтому загрузка выполняется по отдельному запросу.

Запрос:

```
GET “/employees”
```

Параметры запроса:

- «**offset**» – с какого элемента по счету возвращать массив в ответе;
- «**max_results**» – какой максимальный длины может быть массив в ответе;
- «**last_change_date**» – время последней синхронизации по данному запросу.

Тело ответа:

```
[
{
      id: string,
      name: string,
```

```
        first_name: Nullable<string>,
        middle_name: Nullable<string>,
        last_name: Nullable<string>,
        position: Nullable<string>,
        department: Nullable<string>,
        phone: Nullable<string>,
        mobile_phone: Nullable<string>,
        email: Nullable<string>,
        organization_id: Nullable<string>,
        avatar_id: Nullable<string>,
        delete_ts: Nullable<string>
},
…
]
```

Описание параметров:

- «**id**»* – ID сотрудника;
- «**name**»* – локализованное полное наименование сотрудника;
- «**first_name**»* – локализованное имя сотрудника;
- «**middle_name**»* – локализованное отчество сотрудника;
- «**last_name**»* – локализованная фамилия сотрудника;
- «**position**»* – локализованное название должности сотрудника;
- «**department**»* – локализованное название подразделения сотрудника;
- «**phone**»* – номер телефона сотрудника;
- «**mobile_phone**»* – мобильный номер телефона сотрудника;
- «**email**»* – e-mail сотрудника;
- «**organization_id**»* – ID значения справочника организации, к которой относится сотрудник, который синхронизируется в синхронизации справочников;
- «**avatar_id**»* – ID аватара сотрудника.

  Параметр необходимо загрузить по отдельному запросу:

  ```
  GET "/files/<avatar_id>"
  ```

  Телом ответа является загруженный аватар сотрудника в виде «string».

  Так же стоит отметить, что указывать следующий заголовок запроса не нужно:

  ```
  "Accept": "application/json"
  ```

- «**delete_ts**» – время удаления, если по какой-то причине объект был удалён в Системе.

### 5.3.12. Загрузка пользователей

Наименований значений справочника пользователей так же недостаточно в МП, поэтому загрузка выполняется по отдельному запросу.

Запрос:

```
GET "/users"
```

Параметры запроса:

- «**offset**» – с какого элемента по счету возвращать массив в ответе;
- «**max_results**» – какой максимальный длины может быть массив в ответе;
- «**last_change_date**» – время последней синхронизации по данному запросу.

Тело ответа:

```
[
{
      id: string,
      name: string, first_name Nullable<string>,
      middle_name: Nullable<string>,
      last_name: Nullable<string>,
      position: Nullable<string>,
      active: boolean,
      sec_roles: Nullable<string []>,
      employee_id: Nullable<string>,
      organization_id: Nullable<string>,
      delete_ts: Nullable<string>,
},
…
]
```

Описание параметров:

- «**id**»* – ID пользователя;
- «**name**»* – локализованное полное наименование пользователя;
- «**first_name**»* – локализованное имя пользователя;
- «**middle_name**»* – локализованное отчество пользователя;
- «**last_name**»* – локализованная фамилия пользователя;

- «**position**»* – локализованное название должности пользователя;
- «**active**»* – активен пользователь или нет.

  Если пользователь не активен – это означает, что на данный момент пользователь временно не доступен для выбора (например, он в отпуске).
- «**sec_roles**»* – массив имён ролей, которые есть у пользователя;
- «**employee_id**»* – ID значения сотрудника, относящегося к данному пользователю (синхронизируется в синхронизации сотрудников);
- «**organization_id**»* – ID значения справочника организации, к которой относится пользователя (синхронизируется в синхронизации справочников);
- «**delete_ts**»* – время удаления, если по какой-то причине объект был удалён в Системе.

### 5.3.13. Загрузка типовых резолюций

Наименований значений справочника типовых резолюций недостаточно в МП, поэтому загрузка выполняется по отдельному запросу.

Запрос:

```
GET "/typical-resolutions"
```

Параметры запроса:

- «**offset**» – с какого элемента по счету возвращать массив в ответе;
- «**max_results**» – какой максимальный длины может быть массив в ответе;
- «**last_change_date**» – время последней синхронизации по данному запросу.

Тело ответа:

```
[
{
      id: string,
      name: string,
      text: string,
      delete_ts: Nullable<string>,
},
…
]
```

Описание параметров:

- «**id**»* – ID типовой резолюции;

- «**name**»* – локализованное краткое описание текста резолюции;
- «**text**»* – локализованный текст резолюции;
- «**delete_ts**»* – указывается время удаления, если по какой-то причине объект был удалён в Системе.

### 5.3.14. Загрузка уведомлений

Уведомления, которые отображаются в МП на вкладке «Уведомления» синхронизируются по запросу.

Запрос:

```
GET "/notifications"
```

Параметры запроса:

- «**offset**» – с какого элемента по счету возвращать массив в ответе;
- «**max_results**» – какой максимальный длины может быть массив в ответе;
- «**last_change_date**» – время последней синхронизации по данному запросу.

Тело ответа:

```
[
{
      id: string,
      create_ts: string,
      notification_type: string,
      card_id: Nullable<string>,
      message: string,
      title: string,
      delete_ts: Nullable<string>,
},
…
]
```

Описание параметров:

- «**id**»* – ID уведомления;
- «**create_ts**»* – время создания уведомления;
- «**notification_type**»* – тип уведомления.

  Тип уведомления указывает при каком условии оно появилось.

Например, это может быть переход на замещение или выполнение процессного действия по карточке:

- «action» – действие по карточке;
- «overdue» – просроченное действие по карточке;
- «comment» – новый комментарий в карточке;
- «substitution» – информация о замещении;
- «simple» – остальные уведомления.

- «**card_id**»* – ID карточки, результатом действия по которой является данное уведомление;
- «**message**»* – сообщение уведомления;
- «**title**» – заголовок сообщения (появился с версии 5.22);
- «**delete_ts**»* – время удаления, если по какой-то причине объект был удалён в Системе.

## 5.3.15. Загрузка папок действий

Загрузка папок действий, в которые приходят назначенные карточки выполняется по запросу.

Запрос:

```
GET "/app-folders"
```

Тело ответа:

```
[
{
      id: string,
      name: string,
      empty_folder_text: string,
      sort_order: number,
      quantity: number,
      parent_folder: Nullable<string>,
      emphasized: boolean,
},
…
]
```

Описание параметров:

- «**id**»* – ID папки действий;

- «**name**»* – локализованное наименование папки действий;
- «**empty_folder_text**»* – текст, который будет отображаться в папке, если в ней отсутствуют карточки;
- «**sort_order**»* – очередность папки действий;
- «**quantity**»* – количество карточек в папке действий (на данный момент в МП не используется, так как подсчет выполняется в МП среди загруженных карточек);
- «**parent_folder**»* – ID родительской папки действий, в которой данная папка является подпапкой (если родительской папки нет, то значение равно – “”);
- «**emphasized**»* – отметка о том, что в папке есть новые назначенные карточки, т.е. карточки, которые пользователь ещё не открывал после назначения.

## 5.3.16. Загрузка папок поиска

Папки поиска не кэшируются в приложении и доступны только в режиме «онлайн», а их загрузка выполняется по запросу.

Запрос:

```
GET “/search-folders”
```

Тело ответа:

```
[
{
      id: string,
      name: string,
      sort_order: number,
      parent_folder: Nullable<string>,
},
…
]
```

Описание параметров:

- «**id**»* – ID папки поиска;
- «**name**»* – локализованное наименование папки поиска;
- «**sort_order**»* – очередность папки поиска;
- «**parent_folder**»* – ID родительской папки поиска, в которой данная папка является подпапкой (если родительской папки нет, то значение равно – “”).

## 5.3.17. Загрузка карточек

Метаданные типов и видов карточек описывают возможные структуры карточек в Системе, а сами карточки – это уже созданные объекты на основе этих метаданных.

Актуализация карточек выполняется в два этапа:

1. Выполняется запрос, в котором передаётся информация о текущем состоянии МП и приходит ответ с ID карточек.

   В запросе указывается какие карточки уже загружены, какие из них отображаются в папках действий, какие имеют пометку «Важное» и какие не привязаны к папкам и не имеют пометку «Важное», но загружены ранее и необходимы в МП (например карточки, по которым есть уведомления на вкладке «Уведомления» или которые отображаются в списке «История»).

   В ответе приходит информация с ID карточек, которые появились, пропали или изменились в той или иной папке действия, поменяли статус пометки «Важное», а также какие карточки изменились, которые необходимы в МП.

2. На основе ответа предыдущего запроса формируется массив ID карточек, которые необходимо загрузить в МП (которые необходимо обновить или вовсе отсутствуют в МП) и далее выполняется загрузка этих карточек циклично вторым запросом, в ответе которого приходит массив запрашиваемых карточек.

### 5.3.17.1. Загрузка информации с ID карточек

Запрос:

```
POST "/cards-sync"
```

Тело запроса:

```
{
      last_change_date: string,
      folders: CardFoldersInfoRequest [],
      important_card_ids: string [],
      unattached_card_ids: string [],
}
```

Описание параметров:

- «**last_change_date**»* – время последней синхронизации;
- «**folders**»* – массив объектов, определяющий какие карточки к каким папкам действий привязаны.

  Описание формата «CardFoldersInfoRequest» представлено в п.п. 16.10.

- «**important_card_ids**»* – массив ID карточек, которые имеют пометку «Важное»;
- «**unattached_card_ids**»* – массив ID карточек, которые не привязаны к папкам действий, не имеют пометку «Важное», но загружены ранее и необходимы в МП.

Тело ответа:

```
{
      folder_cards_differences: FolderCardDiffResponse [],
      important_cards_difference: ImportantCardDiffResponse,
      unattached_cards_difference: UnattachedCardDiffResponse,
}
```

Описание параметров:

- «**folder_cards_differences**»* – массив объектов, определяющий какие карточки появились/изменились или пропали из той или иной папки действий.

  Описание формата «FolderCardDiffResponse» представлено в п.п. 16.23.

- «**important_cards_difference**»* – объект, определяющий какие карточки сменили статус пометки «Важное», а также имеют пометку «Важное», но были изменены.

  Описание формата «ImportantCardDiffResponse» представлено в п.п. 16.27.

- «**unattached_cards_difference**»* – объект, определяющий какие карточки были изменены среди тех, которые были указаны в теле запроса как необходимые для МП и были изменены.

  Описание формата «UnattachedCardDiffResponse» представлено в п.п. 16.50.

## 5.3.17.2. Загрузка массива карточек

Запрос:

```
POST "/cards/load-request"
```

Тело запроса:

```
string []
```

Тело ответа:

```
{
      cards: CardResponse [],
      not_found: string [],
}
```

Описание параметров:

- «**cards**»* – массив объектов карточек (ради чего выполняется синхронизация карточек).

  Описание формата «CardFoldersInfoRequest» представлено в п.п. 16.16.

- «**not_found**»* – массив ID карточек, которые по какой-то причине не были найдены на сервере.

## 5.3.18. Загрузка связанных карточек

После основной загрузки карточек, МП определяет какие карточки необходимо догрузить и загружает по тому же запросу, что и в основной загрузке во втором этапе.

**Пример:**

Есть требование, что карточки, связанные с карточками в папках действий (например, значение в поле «Основание»), должны быть доступны офлайн.

Их невозможно загрузить в основной синхронизации по той причине, что неизвестно еще какие карточки являются связанными.

Поэтому после основной синхронизации, МП выполняет поиск ID карточек среди только что загруженных в их параметрах.

В результате формируется массив ID карточек, который загружается по тому же запросу, что и в основной синхронизации во втором этапе.

## 5.3.19. Загрузка файлов вложений

Загрузкой файлов вложений в процессе синхронизации выполняется для возможности их просмотра в офлайн.

По умолчанию автозагрузка файлов отключена, так как это может быть ресурсозатратно.

Включается в настройках приложения в пункте «Автозагрузка».

После включения настройки, при следующей синхронизации и после этапа загрузки карточек, вызывается запуск загрузки файлов вложений. Данная загрузка выполняется асинхронно в фоне, поэтому если за 3 минуты все файлы не успели загрузиться, то параллельно начинает выполняться очередная синхронизация данных. По окончанию очередной синхронизации данных снова вызывается загрузка файлов

вложений, а так как она уже выполняется, то происходит актуализация списка загружаемых файлов, и загрузка файлов продолжает выполняться в фоне.

Формат загрузки файлов описан в п.п 8.1.

# 6. Работа с карточками

> **Важно!**
>
> Параметры полей карточек и процессных форм похожи.
>
> Однако сами поля в карточках и в процессных имеют разных смысл.
>
> Поля процессных форм более специфические по своему характеру.

## 6.1. Загрузка карточки

Кроме возможности загрузки массива карточек, так же есть возможность загрузки одной карточки.

Данный запрос полезен, если выполняется переход:

- в связанную карточку, которая не загружена;
- в карточку из уведомления которая тоже не загружена.

Запрос:

```
GET "/cards/<card_id>"
```

Параметры URL:

- «**card_id**» – ID карточки.

Тело ответа:

Телом ответа является «CardResponse», формат которого описан в п.п. 5.3.17.

## 6.2. Инициализация полей новой карточки

Для возможности инициализации значений полей и «permissions» новой карточки при создании, если приложение онлайн, выполняется запрос, который возвращает объект карточки с заполненными «по умолчанию» полями.

Этот новый объект не сохраняется ни на стороне сервера, ни в МП и по факту предоставляет только значения «properties» и «permissions», которыми предзаполняются поля создаваемой карточки.

Запрос:

```
GET "/card_initial_states/<type_id>/<kind_id>"
```

Параметры URL:

- «**type_id**» – ID метаданных типа карточки;
- «**kind_id**» – ID метаданных вида карточки (необязательный).

Тело ответа:

Телом ответа является «CardResponse», формат которого описан в п.п. 16.16.

## 6.3. Онлайн редактирование полей карточки

При создании новой карточки или редактировании существующей, в МП открывается экран редактирования полей карточки.

Доступность выполнения того или иного действия над полем определяют метаданные типов и видов полей карточек.

### 6.3.1. Получение списка возможных значений для ссылочного поля

В некоторых случаях, возможные значения для ссылочного поля необходимо грузить с сервера.

Для таких случаев в метаданных типов полей карточек есть параметр «load_options_from_server».

Если значение параметра равно «true», то список возможных значений необходимо загружать с сервера, а не предлагать пользователю выбрать любое значение из справочника.

Данный параметр принимает значение «true»:

- когда список возможных значений имеет некие условия фильтрации на стороне сервера, которые в силу особенностей реализации невозможно применить в МП;
- если справочник для данного поля не загружен в МП.

Так же, если «load_options_from_server» равен «true», то в офлайн поле автоматически становится не редактируемым, так как будет невозможно загрузить список возможных значений с сервера.

Загрузка списка возможных значений с сервера выполняется по следующему запросу:

```
POST "/card-property-options/search"
```

Тело запроса:

```
{
      type_id: string,
      property_name: string,
      item_id: Nullable<string>,
      property_entity_name: Nullable<string>,
      search_string: string,
      properties: CardPropertyRequest [],
}
```

Описание параметров:

- «**type_id**»* – ID метаданных типа карточки;
- «**property_name**»* – имя поля карточки;
- «**item_id**»* – ID значения поля карточки с типом «composition» (проставляется если данное поле является вложенным полем в composition-поле);
- «**property_entity_name**»* – имя выбранного типа сущности поля.

  Является уточняющим параметром и проставляется, когда значение поля может принимать один из нескольких типов, т.е. если в метаданных типа поля карточки указан параметр «types_to_select».
- «**search_string**»* – строка поиска;
- «**properties**»* – информация о текущем состоянии значений полей редактируемой карточки. Так же в массив должны быть добавлены атрибуты «entity_name», «card_id» и «kind_id» с их значениями.

  Описание формата «CardPropertyRequest» представлено в п.п. 16.13.

Тело ответа:

```
{
      id: string,
      value: string
}
```

Описание параметров:

- «**id**»* – ID значения справочника;
- «**value**»* – локализованное название значения справочника.

### 6.3.2. Возможность динамического изменения состояния полей

В некоторых случаях при изменении значения одного из полей необходимо изменить значения и «permissions» полей, зависящих от данного.

Для таких случаев в метаданных типов полей карточек есть параметр «handle_change_event_on_server».

Если значение параметра равно «true», то после изменения значения данного поля, и наличия онлайн, выполняется запрос на сервер, ответом на который является информация о том значения и «permissions» каких полей надо так же так же заменить.

Запрос:

```
POST "/card-events/property-change"
```

Тело запроса:

```
{
      entity_name: string,
      changed_property: string,
      item_id: Nullable<string>,
      operation: Nullable<string>,
      value: null | boolean | number | string | string[] |
PropertyValueRequest | PropertyValueRequest[],
      card_id: Nullable<string>,
      component_states: PropertyPermissionsRequest [],
      property_valies: CardPropertyRequest [],
}
```

Описание параметров:

- «**entity_name**»* – имя типа сущности поля;
- «**changed_property**»* – имя поля карточки.

  Если данное поле является вложенным полем в поле с типом «composition», то имя поля указывается через точку (например, «meetingQuestions.question»).
- «**item_id**»* – ID значения поля карточки с типом «composition».

  Проставляется, если данное поле является вложенным полем в поле с типом «composition».
- «**operation**»* – название операции, произведённой над значением поля с типом «composition».

  Проставляется, если действие выполняется над значением поля с типом «composition» и принимает значения «create», «update» и «remove».
- «**value**»* – значение поля карточки.

Описание формата «PropertyValueRequest» представлено в п.п. 16.38.

- «**card_id**»* – ID редактируемой карточки (равен «null» если это новая карточка);
- «**component_states**»* – информация видимости/редактируемости полей редактируемой карточки.

  Описание формата «PropertyPermissionsRequest» представлено в п.п. 16.36.
- «**property_values**»* – информация о текущем состоянии значений полей редактируемой карточки.

  Описание формата «CardPropertyRequest» представлено в п.п. 16.13.

Тело ответа:

```
{
    component_states: PropertyPermissionsResponse [],
    property_values: CardPropertyResponse [],
}
```

Описание параметров:

- «**component_states**»* – информация видимости/редактируемости полей карточки, зависимых от изменённого поля.

  Описание формата «PropertyPermissionsResponse» представлено в п.п. 16.37.
- «**property_values**»* – информация значений полей карточки, зависимых от изменённого поля.

  Описание формата «CardPropertyResponse» представлено в п.п. 16.14.

## 6.4. Создание карточки

В случае если это новая карточка, сохранение изменений и создание новой карточки происходит после выполнения изменений полей карточки.

Запрос:

```
POST "/cards"
```

Тело запроса:

```
{
    card_id: string,
    type_id: string,
    kind_id: Nullable<string>,
    properties: CardPropertyRequest [],
    parent_card_id: string,
    assignment_id: string,
```

```
}
```

Описание параметров:

- «**card_id**»* – ID создаваемой карточки;
- «**type_id**»* – ID метаданных типа создаваемой карточки;
- «**kind_id**»* – ID метаданных вида карточки;
- «**properties**»* – информация значений полей создаваемой карточки.

  Описание формата «CardPropertyRequest» представлено в п.п. 16.13.
- «**parent_card_id**» – ID карточки основания, если карточка создается на основании другой карточки;
- «**assignment_id**» – ID назначения процессного действия из «CardActionResponse».

Тело ответа:

Телом ответа является созданная карточка «CardResponse», формат которого описан в п.п. 16.16.

> ⚠ **Важно!**
>
> В случае ошибки валидации полей на стороне сервера, телом ответа будет являться объект, описанный в п.п. 6.6.

## 6.5. Редактирование карточки

После изменения полей карточки, происходит сохранение изменений.

Запрос:

```
POST: "/action-execution-requests"
```

Тело запроса:

```
{
      card: CardResponse,
      entity_name: string,
      action_id: string,
      skip_concurrent_modification_check: string,
      sync_id: string,
```

```
    action_parameters: EditCardParams,
}
```

Описание параметров:

- «**card**»* – объект карточки, в том виде в котором она была загружена в МП.

  Описание формата «CardResponse» представлено в п.п. 16.16.

- «**entity_name**»* – имя сущности (например, «tm$Task»);
- «**action_id**»* – ID выполняемого действия, в данном случае это «update»;
- «**skip_concurrent_modification_check**»* – параметр, определяющий игнорирование изменений данной карточки на сервере.

  Например, если после синхронизации с сервером, карточка была изменена на сервере и изменения не успели синхронизироваться, то будет конфликт синхронизации. Но, если данный параметр будет выставлен в «true», сервер проигнорирует изменения на сервере и применит изменения с МП.

- «**sync_id**»* – ID действия, который используется для избежания дублирования выполнения запроса при плохом соединении с сервером;
- «**action_parameters**»* – информация значений полей карточки с учётом редактирования.

  Описание формата «EditCardParams» представлено в п.п. 16.21.

Тело ответа:

```
{
    card_payload: CardResponse,
    difference: Nullable<ActionExecutionDifferenceResponse>,
    successful: boolean,
    resolvable: boolean,
    error_message: string,
    code: number,
}
```

Описание параметров:

- «**card_payload**»* – новый актуальный объект карточки.

  Описание формата «CardResponse» представлено в п.п. 16.16.

- «**difference**»* – объект, описывающий конфликт синхронизации действия.

  В случае если параметр «successful» равен «false» и «code» равен «», и принимающий значение «null» в других случаях.

  Описание формата «ActionExecutionDifferenceResponse» представлено в п.п. 16.1.

- «**successful**»* – успешность выполнения действия;
- «**resolvable**»* – возможность разрешения конфликта синхронизации путём игнорирования изменений на сервере (учитывается если параметр «successful» равен «false»);
- «**error_message**»* – текст ошибки в случае, если параметр «successful» равен «false»;
- «**code**»* – статус выполненного действия (равен «200», в случае успешного выполнения).

**Важно!**

В случае ошибки валидации полей на стороне сервера телом ответа будет являться объект, описанный в п.п. 6.6.

***Примечание:***

*Идентичные форматы запроса и ответа, за исключением параметра «action_parameters» описаны в п.п. 9.4.*

## 6.6. Валидация полей карточки

Если на стороне сервера будут обнаружены ошибки валидации полей при создании (см. п.п. 6.4) или редактировании карточки (см. п.п. 6.5), то в ответ придёт следующий ответ с ошибкой.

Тело ответ:

```
{
    errors: CardValidationErrorResponse [],
    code: number,
}
```

Описание параметров:

- «**errors**»* – набор сообщений с ошибками к полям карточки.

  Описание формата «CardValidationErrorResponse» представлено в п.п. 16.17.

- «**code**»* – код ошибки (в данном случае – «412»).

## 6.7. Пометка карточки «Важное»

Пометку карточки «Важное» можно изменить.

Запрос:

```
PATCH "/cards/<card_id>"
```

Параметры URL:

- «**card_id**» – ID карточки

Тело запроса:

```
{
      important: boolean,
}
```

Описание параметров:

- «**important**»* – параметр, определяющий на какой статус надо изменить пометку «Важное».

Тело ответа:

Телом ответа является строка, не используемая в МП.

# 7. Работа с вложениями

## 7.1. Добавление вложения

При добавлении нового вложения в карточку, т.е. выборе файла вложения, заполнения параметров вложения и нажатии на кнопку «Сохранить», выполняется следующий запрос на сервер.

Запрос:

```
POST "/cards/<card_id>/attachments"
```

Параметры URL:

- «**card_id**» – ID карточки.

Тело запроса:

```
{
      name: string,
      file_name: string,
      ext: string,
      comment: Nullable<string>,
      attachment_type: string,
      attachment_id: string,
      origin_attachment_id: string,
      file_id: string,
}
```

Описание параметров:

- «**name**»* – название вложения;
- «**file_name**»* – название файла вложения;
- «**ext**»* – расширение файла вложения;
- «**comment**»* – комментарий к добавляемому вложению;
- «**attachment_type**»* – ID метаданных типа вложения;
- «**attachment_id**»* – ID создаваемого вложения;
- «**origin_attachment_id**» – ID вложения, новое вложение которому является версией;

- «**file_id**» – ID файла вложения, указывается при копировании существующего вложения из карточки основания.

Тело ответа:

Телом ответа является строка, не используемая в МП.

Следом выполняется отправка файла вложения, которая описана в п.п. 8.2.

## 7.2. Удаление вложения

Вложение из карточки можно удалить.

Запрос:

```
DELETE "/cards/<card_id>/attachments/<attachment_id>"
```

Параметры URL:

- «**card_id**» – ID карточки;
- «**attachment_id**» – ID вложения.

Тело ответа:

Телом ответа является строка, не используемая в МП.

## 7.3. Пометка вложения как «Основное»

Пометку вложения «Основное» можно изменить.

Запрос:

```
PATCH "/cards/<card_id>/attachments/<attachment_id>"
```

Параметры URL:

- «**card_id**» – ID карточки;
- «**attachment_id**» – ID вложения.

Тело запроса:

```
{
    main: boolean,
}
```

Описание параметров:

- «**main**»* – параметр, определяющий на какой статус надо изменить пометку «Основное».

Стоит отметить, что если параметр равен «true», и какая-либо другая версия вложения уже имеет отметку «Основное», то эта пометка будет снята.

Тело ответа:

Телом ответа является строка, не используемая в МП.

# 8. Работа с файлами вложений

## 8.1. Загрузка файла вложения

Для каждого файла выполняется потоковый запрос при:

- синхронизации файлов вложений;
- клике на вложения в карточке, файлы которых не загружены.

Запрос:

```
"/files<file_id>"
```

Параметры URL:

- «**file_id**» – ID файла в Системе.

Заголовки запроса следующие:

```
"Authorization": "Bearer <token>"
"Accept-language": "<locale>"
```

Вызываемый метод в МП так же принимает параметр «filePath», который указывает куда сохранить файл и callback-функции, которые периодически актуализируют статус загрузки файла.

## 8.2. Выгрузка файла вложения

При добавлении нового вложения (см. п.п. 7.1), после запроса добавления нового вложения, выполняется выгрузка файла вложения на сервер.

Запрос:

```
"/cards/<card_id>/attachments/<attachment_id>/file"
```

Параметры URL:

- «**card_id**» – ID карточки;
- «**attachment_id**» – ID вложения.

Заголовки запроса следующие:

```
"Accept": "application/json"
```

```
“Accept-language”: “<locale>”
“Authorization”: “Bearer <token>”
“Content-Type”: “application/octet-stream”,
“Content-Length”: “<file_size>”
```

Вызываемый метод в МП так же принимает параметр «filePath», в котором содержится путь к отправляемому файлу.

# 9. Выполнение процессных действий

**Важно!**

Параметры полей карточек и процессных форм похожи.

Однако сами поля в карточках и в процессных имеют разных смысл.

Поля процессных форм более специфические по своему характеру.

## 9.1. Выполнение «pre conditional handlers»

Под «pre conditional handlers» подразумевается параметр «pre_conditional_handlers» из «ProcessTransitionResponse» описанного в п.п. 16.35, который представляет собой массив ID скриптов.

Каждый скрипт необходимо выполнить на стороне сервера для определения возможности выполнения процессного действия.

Таким образом, при клике на процессное действие, до перехода в процессную форму, выполняется поочерёдно запрос по каждому ID.

**Важно!**

Процессные действия, у которых есть «pre_conditional_handlers», не доступны офлайн.

Запрос 1 (если о карточке не запущен процесс):

*POST "/cards/<card_id>/process/pre-conditional-handlers"*

Параметры URL:

- «**card_id**» – ID карточки.

Запрос 2 (если по карточке запущен процесс):

*POST "/cards/<card_id>/<process_id>/pre-conditional-handlers"*

Параметры URL:

- «**card_id**» – ID карточки;
- «**process_id**» – ID метаданных процесса.

В случае если по карточке еще не был запущен процесс – используется первый запрос, иначе – второй.

Тело запроса:

```
{
      handler_id: string,
      assignment_id: Nullable<string>,
      state: Nullable<string>,
      action_name: string,
}
```

Описание параметров:

- «**handler_id**»* – ID скрипта;
- «**assignment_id**»* – ID назначения процессного действия из «CardActionResponse»;
- «**state**»* – текущее процессное состояние карточки из «CardResponse»;
- «**action_name**»* – имя процессного действия «name» из «ProcessTransitionResponse» (см. п.п. 16.35).

Тело ответа:

```
{
      successful: boolean,
      message: string,
      card_update_needed: boolean,
}
```

Описание параметров:

- «**successful**»* – успешность выполнения действия.

  Если равен «false», то запросы на выполнение дальнейших скриптов прекращаются и пользователя выводится ошибка.

- «**message**» – текст ошибки.

  Не используется, если «successful» равен «true».

- «**card_update_needed**» – необходимость повторной загрузки карточки в МП после успешного выполнения текущего скрипта.

## 9.2. Загрузка ролей карточки

После нажатия на кнопку процессного действия и выполнения «pre conditional handlers» (если они есть), если действием является запуск какого-либо процесса и в случае онлайн, выполняется дополнительный запрос на сервер на получение ролей карточки, которыми предзаполняется форма запуска процесса поверх «roles» по процессу из «CardResponse».

Запрос:

```
GET "/cards/<card_id>/default-actors/<process_id>"
```

Параметры URL:

- «**card_id**» – ID карточки;
- «**process_id**» – ID метаданных процесса.

Тело ответа:

Ответом является «CardRoleResponse», формат которого описан в п.п. 16.15.

## 9.3. Подписание карточки

В процессной форме есть кнопка «Подписать» выполняющая подписание КриптоПро хэшей вложений и полей карточки.

За доступность подписания в принципе отвечает параметр «sign_enabled», а за доступность подписания дополнительно полей карточки отвечает параметр «sign_card» из «FormSignatureSettingsResponse», описанный в п.п. 16.26.

За обязательность подписания отвечает параметр «sign_required» из «CardActionResponse», описанный в п.п. 16.6.

### 9.3.1. Доп. запрос перед загрузкой хэшей карточки

Если в «CardActionResponse» присутствует параметр «prepare_signature_action», то перед выполнением запроса получения хэшей выполняется дополнительный запрос, необходимый для доп. обработки на сервере.

Запрос:

```
POST "/cards/<card_id>/<endpoint>"
```

Параметры URL:

- «**card_id**» – ID карточки;

- «**endpoint**» – параметр «mapping» из «PrepareSignatureActionResponse» (см. п.п. 16.32).

Тело запроса:

```
{
      [key: string]: any,
}
```

Описание параметров:

- «**[key: string]**»* – объект запроса представляет собой «map», где ключами являются названия полей процессной формы, которые указаны в «requestFields» из «PrepareSignatureActionResponse» (см. п.п. 16.32).

Тело ответа:

Телом ответа является строка, не используемая в МП.

### 9.3.2. Запрос на получение хешей

После предварительного запроса выполняется сам запрос на получение хэшей, которые необходимо подписать.

Запрос:

```
GET "/cards/<card_id>/sign_data"
```

Параметры URL:

- «**card_id**» – ID карточки.

Параметры запроса:

- «**algorithm**» – алгоритм сертификата;
- «**edmRefuse**» – параметр «edm_refuse» из «CardActionResponse» (см. п.п. 16.6).

Тело ответа:

```
{
      card_id: string,
      signed_properties: string [],
      card_hash: string,
      attachment_hash_map: {[attachment_id: string]: string},
}
```

Описание параметров:

- «**card_id**»* – ID карточки;

- «**signed_properties**»* – массив имен полей карточки, которые были учтены при формировании хэша полей карточки;
- «**card_hash**» – хэш полей карточки в формате «*.base64» для подписания.

  Если параметр «sign_card» не равен «true», то данный параметр не используется.
- «**attachment_hash_map**»* – объект, представляющий «map», в котором ключ это ID вложения, а значение хэш этого вложения в формате «*.base64» для подписания.

После получения хэшей карточки и их подписания в МП, результат подписания сохраняется как поле карточки с именем «sign_data» для дальнейшей отправки на сервер с полями формы при нажатии на кнопку «Сохранить» в процессной форме.

## 9.4. Выполнение процессного действия

Для выполнения самого процессного действия по карточке выполняется запрос.

Запрос:

```
POST "/action-execution-requests"
```

Тело запроса:

```
{
    card: CardResponse,
    entity_name: string,
    action_id: string,
    skip_concurrent_modification_check: string,
    sync_id: string,
    action_parameters: StartProcessParams | StartResolutionProcessParams | FinishAssignmentProcessParams | CustomProcessParams | CancelProcessParams,
}
```

Описание параметров:

- «**card**»* – объект карточки, в том виде в котором она была загружена в МП.

  Описание формата «CardResponse» представлено в п.п. 16.16.
- «**entity_name**»* – имя сущности (например, «tm$Task»);
- «**action_id**»* – ID выполняемого действия (например, «startProcess», «finishAssignment», «cancelProcess»);
- «**skip_concurrent_modification_check**»* – параметр, определяющий игнорирование изменений данной карточки на сервере.

Если после синхронизации с сервером, карточка была изменена на сервере и изменения не успели синхронизироваться – будет конфликт синхронизации. Но, если данный параметр будет выставлен в «true», сервер проигнорирует изменения на сервере и применит изменения с МП.

- «**sync_id**»* – ID действия.

  Используется для избежания дублирования выполнения запроса при плохом соединении с сервером.

- «**action_parameters**»* – информация о деталях процессного действия.

  Тип параметра зависит от значения «action_id».

  Описание формата «StartProcessParams» представлено в п.п. 16.44.

  Описание формата «StartResolutionProcessParams» представлено в п.п. 16.45.

  Описание формата «FinishAssignmentProcessParams» представлено в п.п. 16.22.

  Описание формата «CustomProcessParams» представлено в п.п. 16.20.

  Описание формата «CancelProcessParams» представлено в п.п. 16.5.

Тело ответа:

```
{
    card_payload: CardResponse,
    difference: Nullable<ActionExecutionDifferenceResponse>,
    successful: boolean,
    resolvable: boolean,
    error_message: string,
    code: number,
}
```

Описание параметров:

- «**card_payload**»* – новый актуальный объект карточки;
- «**difference**»* – конфликт синхронизации действия.

  В случае если параметр «successful» равен «false» и «code» равен «409» и принимающий значение «null» в других случаях.

- «**successful**»* – успешность выполнения действия;
- «**resolvable**»* – возможность разрешения конфликта синхронизации путём игнорирования изменений на сервере (учитывается если параметр «successful» равен «false»);
- «**error_message**»* – текст ошибки в случае, если параметр «successful» равен «false»;

- «**code**»* – статус выполненного действия (равен «200», в случае успешного выполнения).

***Примечание:***

*Идентичные форматы запроса и ответа, за исключением параметра «action_parameters» описаны в п.п. 6.5.*

# 10. Замещение

Как при замещении пользователя, так и при отмене замещения выполняется запрос на сервер.

После успешного выполнения все данные в МП удаляются и выполняется синхронизация как при первичной синхронизации после авторизации.

Данная логика необходима поскольку что в МП загружаются только данные для работы текущего пользователя.

По этой причине замещение доступно только онлайн.

## 10.1. Выполнение замещения

На создание замещения выполняется запрос.

Запрос:

```
POST "/substitutions/substitute"
```

Тело запроса:

Телом запроса является ID пользователя, обёрнутый в json строку.

Тело ответа:

Телом ответа является значение «Boolean», характеризующее успешность выполнение действия.

## 10.2. Отмена замещения

На отмену замещения выполняется запрос.

Запрос:

```
POST "/substitutions/unsubstitute"
```

Тело запроса:

Телом запроса является пустой объект {}.

Тело ответа:

Телом ответа является значение «Boolean», характеризующее успешность выполнение действия.

# 11. Изменение аватара

## 11.1. Добавление аватара

При добавлении аватара не важно имеется ли уже аватар у текущего пользователя или нет и выполняется потоковый запрос.

Запрос:

```
"/current-user/photo"
```

Вызываемый метод в МП так же принимает параметр «filePath», в котором содержится путь к отправляемому файлу.

## 11.2. Удаление аватара

При удалении аватара вызывается следующий запрос.

Запрос:

```
DELETE "/current-user/photo"
```

Ответ:

Телом ответа является строка, не используемая в МП.

# 12. Вкладка «Обсуждения»

## 12.1. Добавление комментария

В МП, на вкладке «Обсуждения», в режиме онлайн, можно добавлять новые комментарии.

Запрос:

```
POST "/card-comments"
```

Тело запроса:

```
{
      card_id: string,
      parent_id: Nullable<string>,
      comment: string,
}
```

Описание параметров:

- «**card_id**»* – ID карточки, в которую добавляется комментарий;
- «**parent_id**»* – ID комментария основания, если сообщение является ответом на другое сообщение и «null» в противном случае;
- «**comment**»* – новый комментарий.

Тело ответа:

В теле ответа приходит «CardCommentResponse» описанный в п.п. 16.9.

***Примечание:***

*Далее в МП ответ игнорируется и выполняется запрос загрузки актуальной карточки (см. п.п. 6.1) с дальнейшим сохранением.*

*Такая реализация позволяет исключить возможность некорректного состояния карточки.*

## 12.2. Загрузка комментариев

Синхронизация в МП выполняется раз в 3 минуты.

При просмотре какой-либо карточки пользователю было бы не удобно оставлять комментарий и потом ждать пока не выполнится повторная синхронизация для проверки не написал ли кто-то ответ.

Поэтому для решения данной проблемы есть запрос, который выполняется каждые 5 секунд пока карточка открыта.

Запрос:

```
GET "/card-comments/<card_id>"
```

Параметры URL:

- «**card_id**» – ID карточки.

Тело ответа:

В теле ответа приходит массив «CardCommentResponse» описанный в п.п. 16.9.

***Примечание:***

*Далее если массив сообщений не совпадает по длине с массивом сообщений карточки на МП, выполняется запрос загрузки актуальной карточки с дальнейшим сохранением, описанный в п.п. 6.1.*

*Такая реализация позволяет исключить возможность некорректного состояния карточки.*

# 13. Глобальный поиск карточек

В меню «Папки» МП, над папками действий есть текстовое поле ввода глобального поиска карточек.

В режиме «онлайн» выполняется запрос на сервер.

В режиме «офлайн» выполняется поиск локально.

> ⚠ **Важно!**
>
> Исходя из особенностей реализации в МП выполняется два запроса.
>
> Первый запрос выполняется для определения общего числа подходящих карточек и параметр «max_results» равен «1», чтобы не перегружать запрос.
>
> Второй запрос выполняются с адекватным «max_results», для загрузки и отображения результатов.

Запрос:

```
GET "/cards/search"
```

Параметры запроса:

- «**offset**» – с какого элемента по счету возвращать массив в ответе;
- «**max_results**» – какой максимальный длины может быть массив в ответе;
- «**last_change_date**» – время последней синхронизации по данному запросу.

Тело ответа:

```
{
      card_search_results: SearchCardResponse [],
      total_results: number,
}
```

Описание параметров:

- «**card_search_results**» – массив карточек с описанием того, по каким признакам они подходят под условие поиска.

  Описание формата «SearchCardResponse» представлено в п.п. 16.44.

- «**total_results**» – общее количество карточек, подходящее под условие поиска.

# 14. Загрузка карточек в папках поиска

Папки поиска в МП доступны только онлайн.

Карточки по данным папкам не участвуют в синхронизации и загружаются запросом в момент открытия папки.

Запрос:

```
GET "/search-folders/<folder_id>/cards"
```

Параметры URL:

- «**folder_id**» – ID папки поиска.

Параметры запроса:

- «**offset**» – с какого элемента по счету возвращать массив в ответе;
- «**max_results**» – какой максимальный длины может быть массив в ответе;
- «**last_change_date**» – время последней синхронизации по данному запросу.

Тело ответа:

```
{
      cards: CardResponse [],
      total_results: number,
}
```

Описание параметров:

- «**cards**» – массив объектов карточки.

  Описание формата «CardResponse» представлено в п.п. 16.16.

- «**total_results**» – общее количество карточек в данной папке поиска.

# 15. Удаление уведомлений

При открытии карточек, уведомления по этим карточкам становятся не актуальными и удаляются в вебе.

В МП есть запрос, по которому можно удалить уведомление на сервере.

Запрос:

```
DELETE "/notifications/<notification_id>"
```

Параметры URL:

- «**notification_id**» – ID уведомления.

Тело ответа:

Телом ответа является строка, не используемая в МП.

# 16. Форматы

Форматы, используемые в API МП:

1. ActionExecutionDifferenceResponse.
2. ActionFormParamResponse.
3. AttachmentPermissionsResponse.
4. ActionFormValueResponse.
5. CancelProcessParams.
6. CardActionResponse.
7. CardAssignmentResponse.
8. CardAttachmentResponse.
9. CardCommentResponse.
10. CardFoldersInfoRequest.
11. CardOriginatorResponse.
12. CardPermissionsResponse.
13. CardPropertyRequest.
14. CardPropertyResponse.
15. CardRoleResponse.
16. CardResponse.
17. CardValidationErrorResponse.
18. CompositionPropertyPermissionsRequest.
19. CompositionPropertyPermissionsResponse.
20. CustomProcessParams.
21. EditCardParams.
22. FinishAssignmentProcessParams.
23. FolderCardDiffResponse.
24. FormFieldResponse.
25. FormPropertyRequest.
26. FormSignatureSettingsResponse.
27. ImportantCardDiffResponse.
28. MetaDataKindPropertyResponse.

29. MetaDataTypePropertyResponse.
30. NewRolesValueRequest.
31. OldRolesValueRequest.
32. PrepareSignatureActionResponse.
33. ProcessRoleResponse.
34. ProcessStateResponse.
35. ProcessTransitionResponse.
36. PropertyPermissionsRequest.
37. PropertyPermissionsResponse.
38. PropertyValueRequest.
39. PropertyValueResponse.
40. RolesPropertyRequest.
41. SearchCardResponse.
42. SearchHitResponse.
43. SignPropertyRequest.
44. StartProcessParams.
45. StartResolutionProcessParams.
46. TabPermisionsResponse.
47. TransitionDialogResponse.
48. TransitionFormResponse.
49. TransitionNotificationResponse.
50. UnattachedCardDiffResponse.

Описание параметров каждого формата представлено в пунктах ниже.

Форматы расположены в алфавитном порядке.

Обязательные для заполнения параметр отмечены *.

## 16.1. ActionExecutionDifferenceResponse

Формат:

```
{
      changed_properties: string [],
      added_attachment_ids: string [],
      removed_attachment_ids: string [],
```

```
    new_comment_ids: string [],
}
```

Описание параметров:

- «**changed_properties**»* – поля карточки, которые были изменены на сервере после последней синхронизации МП с сервером;
- «**added_attachment_ids**»* – ID вложений, которые были добавлены в карточку на сервере после последней синхронизации МП с сервером;
- «**removed_attachment_ids**»* – ID вложений, которые были удалены из карточки на сервере после последней синхронизации МП с сервером;
- «**new_comment_ids**»* – ID комментариев, которые были добавлены в карточку на сервере после последней синхронизации МП с сервером.

## 16.2. ActionFormParamResponse

Формат:

```
{
    name: string,
    [key: string]: any,
}
```

Описание параметров:

- «**name**»* – имя поля процессной формы, в котором планируется использовать параметры;
- «**[key: string]**»* – остальные параметры объекта, которые должны быть учтены в поле процессной формы.

  Если названия параметров совпадают с названиями параметров поля процессной формы, то они автоматически заменяются без какой-либо доработки со стороны МП.

## 16.3. AttachmentPermissionsResponse

Формат:

```
{
    id: string,
    remove_enabled: boolean,
    new_version_enabled: boolean,
    main_version_enabled: boolean,
```

```
}
```

Описание параметров:

- «**id**»* – ID вложения;
- «**remove_enabled**» – возможность удаления вложения;
- «**new_version_enabled**» – возможность добавления новой версии вложения;
- «**main_version_enabled**» – возможность пометки вложения как «Основное».

## 16.4. ActionFormValueResponse

Формат:

```
{
      name: string,
      value: null | boolean | number | string | CardRoleResponse
[],
}
```

Описание параметров:

- «**name**»* – имя поля процессной формы, которое необходимо проинициализировать;
- «**value**»* – значение, которым необходимо проинициализировать поле процессной формы (формат значения должен соответствовать типу поля).

  Описание формата «CardRoleResponse» представлено в п.п. 16.15.

## 16.5. CancelProcessParams

Формат:

```
{
      handler_id: string,
      action_name: string,
      process_id: string,
      form_commit_data:            (FormPropertyRequest            |
RolesPropertyRequest | SignPropertyRequest) [],
}
```

Описание параметров:

- «**handler_id**»* – ID обработчика процессного действия на стороне сервера.

Возможные значения:

- → «id» из TransitionFormResponse;
- → «handler_id» из TransitionDialogResponse;
- → «handler_id» из TransitionNotificationResponse.

- «**action_name**»* – имя процессного действия «name» из «ProcessTransitionResponse» (см. п.п. 16.35) запускаемого процесса;
- «**process_id**»* – ID метаданных запускаемого процесса;
- «**form_commit_data**»* – массив значений полей формы запуска процесса.

  Может быть пустым массивом, если форма в назначении отсутствует и вместо неё отображается диалоговое окно или нотификация.

  Описание формата «FormPropertyRequest» представлено в п.п. 16.25.

  Описание формата «RolesPropertyRequest» представлено в п.п. 16.40.

  Описание формата «SignPropertyRequest» представлено в п.п. 16.43.

## 16.6. CardActionResponse

Формат:

```
{
      name: string,
      assignment_id: Nullable<string>,
      form_params: Nullable<ActionFormParamResponse []>,
      form_values: Nullable<ActionFormValueResponse []>,
      process_ids_available_to_start: string [],
      card_types: string [],
      proc_id: string,
      sign_required: boolean,
      edm_refuse: boolean,
      prepare_signature_action:
Nullable<PrepareSignatureActionResponse>,
}
```

Описание параметров:

- «**name**»* – имя процессного действия.

  Соответствующее «name» из «ProcessTransitionResponse» (см. п.п. 16.35) или равное *“START_PROCESS_ACTION”/”CANCEL_PROCESS_ACTION”*.

- «**assignment_id**»* – ID назначения процессного действия.

  Используется в теле запроса на выполнение процессного действия.

- «**form_params**» – массив объектов, с помощью которых можно подменять любые параметры полей процессной формы.

  Например, есть поле «Комментарий» с именем «name = “comment”, в процессной форме.

  Его параметры определены в метаданных процесса объектом «FormFieldResponse» (см. п.п. 16.24) и там прописано «required = false».

  Если необходимо, чтобы только у этой карточки в данной процессной форме поле было обязательно для заполнения, то в данном массиве объектов следует передать объект, у которого «name = “comment”», а «required = true».

  Кроме этого, можно передавать любые специфические параметры с сервера, которых нет в «FormFieldResponse» (см. п.п. 16.24) и через доработку МП учитывать эти параметры с учётом требований.

  Описание формата «ActionFormParamResponse» представлено в п.п. 16.2.

- «**form_values**» – массив объектов, с помощью которого можно инициализировать поля процессной формы.

  Описание формата «ActionFormValueResponse» представлено в п.п. 16.4.

- «**process_ids_available_to_start**» – ID процессов доступных для запуска.

  Используется в процессе «Резолюция» на этапе «Обработка резолюции» в действии «Запустить».

- «**card_types**» – ID типов карточек доступных для создания карточки на основании.

  Используется в процессе «Резолюция» на этапе «Обработка резолюции» в действии «Создать».

- «**proc_id**» – ID процесса.

  Используется в случае, если «name» равно “*START_PROCESS_ACTION*”/”*CANCEL_PROCESS_ACTION*”.

- «**sign_required**» – обязательность подписания в данном процессном действии;

- «**edm_refuse**» – URL-запроса хэшей карточки для подписания;

- «**prepare_signature_action**» – объект, используемый при выполнении подписания.

  Если присутствует – перед подписанием выполняется дополнительный запрос, endpoint которого указан в «mapping» данного объекта.

  Описание формата «PrepareSignatureActionResponse» представлено в п.п. 16.32.

## 16.7. CardAssignmentResponse

Формат:

```
{
      id: string,
      create_ts: string,
      finish_due: Nullable<string>,
      finished: Nullable<string>,
      user: string,
      user_id: string,
      process: string,
      state: string,
      result: Nullable<string>,
      comment: Nullable<string>,
}
```

Описание параметров:

- «**id**»* – ID назначения в журнале действий;
- «**create_ts**»* – время создания назначения в журнале действий;
- «**finish_due**»* – время, к которому необходимо завершить текущее назначение;
- «**finished**»* – время завершения назначения;
- «**user**»* – имя пользователя, на которого необходимо выполнить данное назначение;
- «**user_id**»* – ID пользователя, которому необходимо выполнить данное назначение;
- «**process**»* – локализованное название процесса, к которому относится данное назначение;
- «**state**»* – локализованное название состояния, к которому относится данное назначение;
- «**result**»* – локализованное название результата выполнения действия по данному назначению;
- «**comment**»* – комментарий, оставленный в результате выполнения данного назначения.

## 16.8. CardAttachmentResponse

Формат:

```
{
      id: string,
      file_id: string,
      create_ts: string,
      name: string,
      file_name: string,
      comment: Nullable<string>,
      ext: string,
      creator: string,
      main: boolean,
      version_of: string,
      attachment_type: string,
      version: number,
}
```

Описание параметров:

- «**id**»* – ID вложения;
- «**file_id**»* – ID файла вложения;
- «**create_ts**»* – время создания вложения;
- «**name**»* – название вложения;
- «**file_name**»* – название файла вложения;
- «**comment**»* – комментарий к вложению;
- «**ext**»* – расширение файла вложения;
- «**creator**»* – ID пользователя, который добавил вложение в карточку;
- «**main**»* – наличие пометки «Основное» у вложения;
- «**version_of**»* – ID первой версии вложения.

  Если данное вложение является первой версией – то значение будет равно “”.
- «**attachment_type**»* – ID метаданных типа вложения.
- «**version**»* – версия вложения.

## 16.9. CardCommentResponse

Формат:

```
{
      id: string,
      create_ts: string,
      state: Nullable<string>,
      outcome: Nullable<string>,
      parent_id: Nullable<string>,
      comment_type: string,
      comment: string,
      sender_id: string,
      substituted_sender_id: Nullable<string>,
}
```

Описание параметров:

- «**id**»* – ID комментария;
- «**create_ts**»* – время добавления комментария;
- «**state**»* – локализованное процессное состояние карточки, в котором она была в момент добавления комментария;
- «**outcome**»* – локализованное название результата перехода по процессу (заполняется если комментарий системный по процессу);
- «**parent_id**»* – ID родительского комментария (заполняется если данный комментарий является ответом на другой комментарий);
- «**comment_type**»* – тип комментария.

  Может принимать значения:

  → «C» – сообщение, оставленное пользователем на вкладке обсуждение;

  → «P» – процессное сообщение, оставленное автоматически по процессу;

  → «A» – сообщение о добавлении вложения;

  → «D» – сообщение об удалении вложения.
- «**comment**»* – комментарий;
- «**sender_id**»* – ID пользователя, который добавил данный комментарий;
- «**substituted_sender_id**»* – ID замещаемого пользователя, который добавил комментарий (если комментарий был добавлен под замещением).

## 16.10. CardFoldersInfoRequest

Формат:

```
{
      id: string,
      card_ids: string [],
}
```

Описание параметров:

- «**id**» – ID папки действий;
- «**card_ids**» – массив ID карточек, которые привязаны к данной папке действий.

## 16.11. CardOriginatorResponse

Формат:

```
{
      type: string,
      value: string,
      user_id: Nullable<string>,
}
```

Описание параметров:

- «**type**»* – тип создателя (физ. лицо или юр. лицо).

  Доступные значения:

  → «string»;

  → «person».

- «**value**»* – локализованное название создателя (заполняется, если «type = «string»);
- «**user_id**»* – ID пользователя (заполняется, если «type = «person»).

## 16.12. CardPermissionsResponse

Формат:

```
{
      card_tabs: TabPermisionsResponse [],
      properties: PropertyPermissionsResponse [],
      attachments: AttachmentPermissionsResponse [],
```

```
    new_attachment_enabled: boolean,
}
```

Описание параметров:

- «**card_tabs**»* – информация видимости/редактируемости вкладок карточки.

  Описание формата «TabPermisionsResponse» представлено в п.п. 16.46.

- «**properties**»* – информация видимости/редактируемости полей карточки.

  Описание формата «PropertyPermissionsResponse» представлено в п.п. 16.37.

- «**attachments**»* – информация редактируемости вложений карточки.

  Описание формата «AttachmentPermissionsResponse» представлено в п.п. 16.3.

- «**new_attachment_enabled**» – возможность добавления новых вложений в карточку.

## 16.13. CardPropertyRequest

Формат:

```
{
    property: string,
    value: null | boolean | number | string | string[] | PropertyValueRequest | PropertyValueRequest[],
}
```

Описание параметров:

- «**property**»* – имя поля карточки;
- «**value**»* – значение поля карточки.

  Описание формата «PropertyValueRequest» представлено в п.п. 16.38.

## 16.14. CardPropertyResponse

Формат:

```
{
    property: string,
    value: null | boolean | number | string | PropertyValueResponse | PropertyValueResponse [],
}
```

Описание параметров:

- «**property**»* – имя поля карточки;
- «**value**»* – значение поля карточки.

  Формат значения должен соответствовать типу поля.

  Описание формата «PropertyValueResponse» представлено в п.п. 16.39.

## 16.15. CardRoleResponse

Формат:

```
{
      id: string,
      code: string,
      user_id: Nullable<string>,
      process_id: string,
      proc_name: string,
      role_name: string,
      sort_order: Nullable<number>,
      duration: Nullable<number>,
      time_unit: Nullable<string>,
      editable: boolean,
}
```

Описание параметров:

- «**id**»* – ID роли карточки;
- «**code**»* – код процессной роли (например, «10-Initiator»);
- «**user_id**»* – ID пользователя, назначенного на данную роль (принимает значение «null» если пользователь на роль не назначен);
- «**process_id**»* – ID метаданных процесса, к которому относится данная роль;
- «**proc_name**»* – локализованное название процесса, к которому относится данная роль;
- «**role_name**»* – локализованное название роли;
- «**sort_order**» – очерёдность выполнения назначения, если в процессе доступно последовательное выполнение назначений;
- «**duration**» – выполнения назначения данной роли, если в процессе доступно указание времени выполнения;
- «**time_unit**» – единицы времени выполнения назначения данной роли, если в процессе доступно указание времени выполнения;

- «**editable**» – возможность редактирования данной роли.

## 16.16. CardResponse

Формат:

```
{
      id: string,
      type_id: string,
      kind_id: Nullable<string>,
      version: number,
      creator: string,
      create_ts: string,
      card_originator_info: Nullable<CardOriginatorResponse>,
      list_item_info: Nullable<string []>,
      additional_header_info: Nullable<string []>,
      important: boolean,
      description: string,
      state: Nullable<string>,
      loc_state: string,
      process_id: Nullable<string>,
      process_ids_available_to_start: Nullable<string []>,
      actions: Nullable<CardActionResponse []>,
      cancel_process_enabled: boolean,
      properties: CardPropertyResponse [],
      permissions: Nullable<CardPermissionsResponse>,
      attachments: Nullable<CardAttachmentResponse []>,
      comments: CardCommentResponse [],
      assignments: Nullable<CardAssignmentResponse []>,
      roles: Nullable<CardRoleResponse []>,
      registered: boolean,
      edm_formalized: boolean,
}
```

Описание параметров:

- «**id**»* – ID карточки;
- «**type_id**»* – ID метаданных типа карточки;
- «**kind_id**»* – ID метаданных вида карточки;
- «**version**» – версия карточки (на данный момент не используется в МП);
- «**creator**» – имя создателя карточки (на данный момент не используется в МП);
- «**create_ts**»* – время создания карточки;

- «**card_originator_info**»* – объект, определяющий создателя карточки.

  Может быть любым субъектом (физ. лицом или юр. лицом).

  Из него берётся информация для отображения создателя карточки в списке карточек.

  Описание формата «CardOriginatorResponse» представлено в п.п. 16.11.

- «**list_item_info**»* – массив строк, в которых можно передать любую текстовую информацию, и она отобразится в списке карточек;
- «**additional_header_info**»* – массив строк, в которых можно передать любую текстовую информацию, и она отобразится в карточке под заголовком;
- «**important**»* – наличие пометки «Важное» у карточки;
- «**description**»* – описание карточки, которое отображается в заголовке карточки;
- «**state**»* – текущее процессное состояние карточки;
- «**loc_state**»* – локализованное название текущего процессного состояния карточки;
- «**process_id**»* – ID метаданных процесса, по которому была запущена карточка (равен «null» или “” если по процессу не запускалась);
- «**process_ids_available_to_start**»* – ID метаданных процессов, доступных для запуска (равен «null» если какой-либо процесс по данной карточке уже запущен);
- «**actions**»* – список доступных процессных действий по карточке.

  Если процесс по карточке не запущен, то список доступных процессных действий строится по «process_ids_available_to_start», а в данном параметре могут находиться только вспомогательные объекты, у которых «name» = “*START_PROCESS_ACTION*”.

  Описание формата «CardActionResponse» представлено в п.п. 16.6.

- «**cancel_process_enabled**»* – возможность отмены процесса;
- «**properties**»* – значения полей карточки.

  Описание формата «CardPropertyResponse» представлено в п.п. 16.14.

- «**permissions**»* – разрешения вкладок карточки и их содержимого.

  Описание формата «CardPermissionsResponse» представлено в п.п. 16.12.

- «**attachments**»* – вложения в карточке.

  Описание формата «CardAttachmentResponse» представлено в п.п. 16.8.

- «**comments**»* – комментарии в карточке.

Описание формата «CardCommentResponse» представлено в п.п. 16.9.

- «**assignments**»* – журнал назначений в карточке.

  Описание формата «CardAssignmentResponse» представлено в п.п. 16.7.

- «**roles**»* – предзаполненные процессные роли карточки для того или иного процесса.

  Описание формата «CardRoleResponse» представлено в п.п. 16.15.

- «**registered**» – является ли документ зарегистрированным (учитывается в процессной форме при назначении ролей);

- «**edm_formalized**» – является ли документ формализованным (учитывается в процессной форме при назначении ролей).

## 16.17. CardValidationErrorResponse

Формат:

```
{
      message: string,
      properties: string [],
      item_id: string,
}
```

Описание параметров:

- «**message**»* – текст ошибки;

- «**properties**»* – массив имен полей, к которым относится эта ошибка.

  Если поле с ошибкой является вложенным в значение composition-поле, то имя поля будет указано через точку (например, «meetingQuestions.question»).

- «**item_id**» – ID значения поля с типом «composition».

  Указывается, если поле с ошибкой является вложенным в composition-поле.

## 16.18. CompositionPropertyPermissionsRequest

Формат:

```
{
      id: string,
      editable: boolean,
      removable: boolean,
      properties: Nullable< PropertyPermissionsResponse []>,
```

```
}
```

Описание параметров:

- «**id**»* – ID ссылочного значения поля карточки;
- «**editable**»* – возможность редактирования значения поля карточки с типом «composition»;
- «**removable**» – возможность удаления значения поля карточки с типом «composition»;
- «**properties**» – информация видимости/редактируемости полей значения поля карточки с типом «composition».

  Описание формата «PropertyPermissionsResponse» представлено в п.п. 16.37.

## 16.19. CompositionPropertyPermissionsResponse

Формат:

```
{
      id: string,
      editable: boolean,
      removable: boolean,
      properties: Nullable< PropertyPermissionsResponse []>,
}
```

Описание параметров:

- «**id**»* – ID ссылочного значения поля карточки;
- «**editable**»* – возможность редактирования значения поля карточки с типом «composition»;
- «**removable**» – возможность удаления значения поля карточки с типом «composition»;
- «**properties**» – не обязательный массив объектов, представляющий собой информацию видимости/редактируемости полей значения «composition» поля карточки.

  Описание формата «PropertyPermissionsResponse» представлено в п.п. 16.37.

## 16.20. CustomProcessParams

Формат:

```
{
      handler_id: Nullable<string>,
      state: Nullable<string>,
      assignment_id: Nullable<string>,
      action_name: string,
      form_commit_data:            (FormPropertyRequest            |
RolesPropertyRequest | SignPropertyRequest) [],
}
```

Описание параметров:

- «**handler_id**»* – ID обработчика процессного действия на стороне сервера.

  Возможные значения:

  → «id» из TransitionFormResponse;

  → «handler_id» из TransitionDialogResponse;

  → «handler_id» из TransitionNotificationResponse.

- «**state**»* – текущее процессное состояние карточки из «CardResponse»;
- «**assignment_id**»* – ID назначения процессного действия из «CardActionResponse»;
- «**action_name**»* – имя процессного действия «name» из «ProcessTransitionResponse» запускаемого процесса;
- «**form_commit_data**»* – значения полей формы запуска процесса.

  Может быть пустым массивом, если форма в назначении отсутствует и вместо неё отображается диалоговое окно или нотификация.

  Описание формата «FormPropertyRequest» представлено в п.п. 16.25.

  Описание формата «RolesPropertyRequest» представлено в п.п. 16.40.

  Описание формата «SignPropertyRequest» представлено в п.п. 16.43.

## 16.21. EditCardParams

Формат:

```
{
      properties: CardPropertyRequest [],
}
```

Описание параметров:

- «**properties**»* – новые значения полей карточки.

Описание формата «CardPropertyRequest» представлено в п.п. 16.13.

## 16.22. FinishAssignmentProcessParams

Формат:

```
{
      handler_id: string,
      state: string,
      assignment_id: string,
      action_name: string,
      form_commit_data:            (FormPropertyRequest            |
RolesPropertyRequest | SignPropertyRequest) [],
}
```

Описание параметров:

- «**handler_id**»* – ID обработчика процессного действия на стороне сервера.

  Возможные значения:

  → «id» из TransitionFormResponse;

  → «handler_id» из TransitionDialogResponse;

  → «handler_id» из TransitionNotificationResponse.

- «**state**»* – текущее процессное состояние карточки из «CardResponse»;
- «**assignment_id**»* – ID назначения процессного действия из «CardActionResponse»;
- «**action_name**»* – имя процессного действия «name» из «ProcessTransitionResponse» запускаемого процесса;
- «**form_commit_data**»* – значения полей формы запуска процесса.

  Может быть пустым массивом, если форма в назначении отсутствует и вместо неё отображается диалоговое окно или нотификация.

  Описание формата «FormPropertyRequest» представлено в п.п. 16.25.

  Описание формата «RolesPropertyRequest» представлено в п.п. 16.40.

  Описание формата «SignPropertyRequest» представлено в п.п. 16.43.

## 16.23. FolderCardDiffResponse

Формат:

```
{
      folder_id: string,
      updated_and_new_card_ids: string [],
      missing_card_ids: string [],
      removed_card_ids: string [],
}
```

Описание параметров:

- «**folder_id**»* – ID папки действий;
- «**updated_and_new_card_ids**»* – ID карточек, которые были изменены или появились в данной папке действий;
- «**missing_card_ids**»* – ID карточек, которые пропали из папки действий;
- «**removed_card_ids**»* – ID карточек, которые находились в папках действий и были удалены в Системе.

## 16.24. FormFieldResponse

Формат:

```
{
      name: string,
      loc_name: string,
      type: string,
      required: boolean,
      readonly: boolean,
      multiple: boolean,
      multiline: boolean,
      required_roles: string,
      visible_roles: string,
      duration_enabled: boolean,
      order_enabled: boolean,
      sec_roles: string [],
      options: {name: string, loc_name: string} [],
      enum: {key: string, value: string} [],
      self_assign: boolean,
}
```

Описание параметров:

- «**name**»* – имя поля (например, «comment»);
- «**loc_name**»* – локализованное название поля;
- «**type**»* – тип поля.

  Возможные значения:

  → простой тип («boolean», «int», «string» и т.д.);

  → ссылочный («userSelect», «cardRolesSelect» и т.д.).
- «**required**» – обязательность заполнения поля (то есть процессное действие нельзя будет выполнить если есть хоть одно незаполненное поле с признаком «required» равное «true»);
- «**readonly**» – возможность редактирования поля в процессной форме;
- «**multiple**» – возможность множественного значения в поле с типом «userSelect»;
- «**multiline**» – многосторонность текстового поля (в МП игнорируется и считается что всегда равен «true»);
- «**required_roles**» – используется в поле с типом «cardRolesSelect» и определяет коды процессных ролей обязательных к заполнению.

  Коды ролей перечислены в строке через «,» и «|».

  Считаются доступными к заполнению даже если отсутствуют в «visible_roles».
- «**visible_roles**» – используется в поле с типом «cardRolesSelect» и определяет коды процессных ролей доступных к заполнению.

  Коды ролей перечислены в строке через «,»”.
- «**duration_enabled**» – используется в поле с типом «cardRolesSelect» и определяет возможность задания длительности выполнения для процессных ролей;
- «**order_enabled**» – используется в поле с типом «cardRolesSelect» и определяет возможность задание очерёдности назначенных пользователей на процессные роли;
- «**sec_roles**» – используется в поле с типом «userSelect» и определяет имена ролей доступа, которые необходимы пользователю, добавляемому в значение данного поля;
- «**options**» – используется в поле с типом «radioButtonSet» и определяет возможные значения данного поля;
- «**enum**» – используется в поле с типом «enum» и определяет возможные значения данного поля;

- «**self_assign**» – используется в поле с типом «userSelect» и определяет возможность добавления самого себя в значение поля.

## 16.25. FormPropertyRequest

Формат:

```
{
    name: string,
    value: string | number | boolean | null,
}
```

Описание параметров:

- «**name**»* – имя поля карточки;
- «**value**»* – значение поля карточки.

## 16.26. FormSignatureSettingsResponse

Формат данных:

```
{
    sign_enabled: boolean,
    sign_card: boolean,
}
```

Описание параметров:

- «**sign_enabled**»* – возможность подписания КриптоПро в данной процессной форме;
- «**sign_card**»* – возможность подписания полей карточки КриптоПро (по умолчанию, если параметр равен «false», подписываются только вложения карточки).

## 16.27. ImportantCardDiffResponse

Формат данных:

```
{
    updated_and_new_card_ids: string [],
    missing_card_ids: string [],
    removed_card_ids: string [],
}
```

Описание параметров:

- «**updated_and_new_card_ids**»* – ID карточек, которые имеют пометку «Важное» и были изменены или получили пометку «Важное»;
- «**missing_card_ids**»* – ID карточек, у которых пропала пометка «Важное»;
- «**removed_card_ids**»* – ID карточек, которые имели пометку «Важное» и были удалены в Системе.

## 16.28. MetaDataKindPropertyResponse

Формат данных:

```
{
    property: string,
    required: boolean,
}
```

Описание параметров:

- «**property**»* – имя поля, например «comment»;
- «**required**»* – обязательность заполнения поля при создании/редактировании карточки.

## 16.29. MetaDataTypePropertyResponse

Формат данных:

```
{
    property: string,
    type: string,
    required: boolean,
    readonly: boolean,
    cardinality: string,
    additional_info: boolean,
    loc_name: {[index: string]: string},
    load_options_from_server: boolean,
    types_to_select: [{name: string, loc_name: string}, …],
    offline_disabled: boolean,
    handle_change_event_on_server: boolean,
    max_length: number,
    multiline: boolean,
    enum: [{key: string, value: string}, …],
    detailed_info: boolean,
```

```
    composition_properties: MetaDataTypePropertyResponse [],
    nested_entity: boolean,
    name_pattern: string,
}
```

Описание параметров:

- «**property**»* – имя поля (например, «comment»);
- «**type**»* – тип поля.

  Возможные значения:

  → простой тип («boolean», «int», «string» и т.д.);

  → ссылочный («df$Employee», «sec$User», «wf$Card» и т.д.).
- «**required**»* – обязательность заполнения поля.

  В режиме создания/редактирования карточки изменения нельзя сохранить, если есть хоть одно незаполненное поле с признаком «required» равное «true».
- «**readonly**»* – возможность отображения данного поля в режиме редактирования.

  Если равен «true», то в режиме создания/редактирования поле не отображается.

  Например, это может быть какое ни будь системное поле.
- «**cardinality**»* – является ли значение поля массивом или нет.

  Доступные значения:

  → «one»;

  → «many».
- «**additional_info**»* – в какой группе отображать поле на вкладке детали «Основная информация» (при открытии карточки группа развернута) или «Дополнительная информация» (при открытии карточки группа идет после основной группы и свернута);
- «**loc_name**»* – локализованное название поля;
- «**load_options_from_server**» – используется при создании/редактировании карточки.

  Указывается в полях с ссылочным типом.

  Равен «true» если для поля задано условие фильтрации и тогда доступные значения формируются сервером, а редактирование поля только в режиме онлайн (см. более подробное описание в разделе 6).
- «**types_to_select**» – используется при создании/редактировании карточки.

Параметр «type» может быть обобщающим (например, «df$Correspondent»), тогда в «types_to_select» необходимо уточнить какие именно типы доступны для выбора).

- «**offline_disabled**» – используется при создании/редактировании карточки/

  Если равен «true», то параметр становится недоступным для редактирования.

- «**handle_change_event_on_server**» – используется при создании/редактировании карточки.

  Значение равное «true» указывает на наличие обработчика изменений значений на стороне сервера.

- «**max_length**» – максимальное количество символов в значении поля с примитивным типом данных («int», «string» и т.п.);

- «**multiline**»* – определяет многосторонность текстового поля (в МП игнорируется и считается что всегда равен «true»);

- «**enum**» – необходим для поля с типом «enum» и определяет возможные значения поля;

- «**detailed_info**» – возможность просмотра значений поля с типом «composition» в отдельном окне;

- «**composition_properties**» – метаинформация о формате значений поля с типом «composition».

  Если параметр отсутствует, то просмотреть значения данного поля в отдельном окне будет невозможно.

  Описание формата «MetaDataTypePropertyResponse» представлено в п.п. 16.29.

- «**nested_entity**» – определяет является поле «composition» или нет;

- «**name_pattern**» – позволяет сформировать имя нового значения поля с типом «composition» в списке значений при создании/редактировании карточки.

## 16.30. NewRolesValueRequest

Формат:

```
{
      id: string,
      user_id: string,
      code: string,
      sort_order: Nullable<string>,
      duration: Nullable<number>,
      time_unit: Nullable<string>,
}
```

Описание параметров:

- «**id**»* – ID роли карточки;
- «**user_id**»* – ID пользователя;
- «**code**»* – код процессной роли (например, «10-Initiator»);
- «**sort_order**» – очерёдность выполнения назначения (если в процессе доступно последовательное выполнение назначений);
- «**duration**» – время выполнения назначения данной роли (если в процессе доступно указание времени выполнения);
- «**time_unit**» – единицы времени выполнения назначения данной роли (если в процессе доступно указание времени выполнения).

## 16.31. OldRolesValueRequest

Формат:

```
{
      id: string,
      user_id: Nullable<string>,
}
```

Описание параметров:

- «**id**»* – ID роли карточки;
- «**user_id**»* – ID пользователя.

## 16.32. PrepareSignatureActionResponse

Формат:

```
{
      mapping: string,
      requestFields: string [],
      readOnlyFields: string [],
}
```

Описание параметров:

- «**mapping**»* – endpoint по которому необходимо выполнить запрос перед подписанием;
- «**requestFields**»* – массив имен полей процессной формы, значения которых надо передать в теле запроса по endpoint;

- «**readOnlyFields**»* – массив имен полей процессной формы, которые должны стать недоступными для редактирования после выполнения запроса по endpoint и подписания.

## 16.33. ProcessRoleResponse

Формат данных:

```
{
      code: string,
      loc_name: string,
      sec_role: Nullable<string>,
      multiple_users: boolean,
      order_filling_type: Nullable<string>,
}
```

Описание параметров:

- «**code**»* – код процессной роли (например, «10-Initiator»);
- «**loc_name**»* – локализованное название процессной роли;
- «**sec_role**»* – имя роли доступа, которую должен иметь пользователь, назначаемый на данную процессную роль;
- «**multiple_users**»* – возможность множественного назначения на данную роль;
- «**order_filling_type**» – возможность параллельного назначения на данную роль.

  Возможные значения:

  → «P» – параллельное назначение;

  → «S» – последовательное назначение.

  Если «multiple_users» равен «false», то параметр ни на что не влияет.

## 16.34. ProcessStateResponse

Формат данных:

```
{
     name: string,
     transitions: ProcessTransitionResponse[],
}
```

Описание параметров:

- «**name**»* – имя состояния в процессе;

- «**transitions**»* – список описаний процессных действий доступных в этом состоянии.

  Описание формата «ProcessTransitionResponse» представлено в п.п. 16.35.

## 16.35. ProcessTransitionResponse

Формат данных:

```
{
      name: string,
      loc_name: string,
      action_id: string,
      remove_card_from_app_folder: boolean,
      success: Nullable<boolean>,
      pre_conditional_handlers: Nullable<{id: string}[]>,
      notification_action: TransitionNotificationResponse,
      confirm_dialog: TransitionDialogResponse,
      form: TransitionFormResponse,
}
```

Описание параметров:

- «**name**»* – имя процессного действия;
- «**loc_name**»* – локализованное название процессного действия;
- «**action_id**» – ID действия.

  Присутствует, если действие надо отличать от «startProcess», «finishAssignment», и «cancelProcess» для кастомной обработки.
- «**remove_card_from_app_folder**»* – необходимость сокрытия карточки из папки действия после выполнения процессного действия;
- «**success**»* – стиль(цвет) кнопки процессного действия в МП.

  Возможные значения:

  → «true» – зелёный;

  → «false» – красный;

  → «null» – синий.
- «**pre_conditional_handlers**»* – ID скриптов, которые необходимо выполнить перед выполнением процессного действия (см. более подробное описание в п.п. 9.1);
- «**notification_action**» – уведомление, отображающееся после выполнения процессного действия.

Описание формата «TransitionNotificationResponse» представлено в п.п. 16.49.

- «**confirm_dialog**» – диалоговое окно подтверждающее процессное действие.

  Описание формата «TransitionDialogResponse» представлено в п.п. 16.47.

- «**form**» – процессная форма с полями которая заполняется перед выполнением процессного действия.

  Описание формата «TransitionFormResponse» представлено в п.п. 16.48.

**Важно!**

Параметры «confirm_dialog», «notification_action» и «form» являются взаимоисключающими.

Присутствие того или иного параметра определяется следующим образом:

- «form» – используются, если необходимо внести дополнительную информацию перед выполнением самого действия (например, заполнить поле «Комментарий»);
- «confirm_dialog» – используются, когда от пользователя требуется подтверждение выполнения действия (например, «Вы действительно хотите выполнить действие?» и варианты «Да» и «Нет»);
- «notification_action» – используются, когда никаких дополнительных действий от пользователя не требуется, в таком случае действие выполнится сразу и после будет отображено уведомление (например, «Действие выполнено»).

## 16.36. PropertyPermissionsRequest

Формат:

```
{
      name: string,
      editable: boolean,
      visible: boolean,
      required: boolean,
      items: Nulable<CompositionPropertyPermissionsRequest[]>,
}
```

Описание параметров:

- «**name**»* – имя поля карточки (например, «comment»);

- «**editable**»* – возможность редактирования значения поля карточки;
- «**visible**»* – видимость поля карточки;
- «**required**» – обязательность заполнения значения поля карточки;
- «**items**» – информация видимости/редактируемости параметров значений поля карточки с типом «composition».

  Описание формата «CompositionPropertyPermissionsRequest» представлено в п.п. 16.18.

## 16.37. PropertyPermissionsResponse

Формат:

```
{
      name: string,
      editable: boolean,
      visible: boolean,
      required: boolean,
      new_item_properties:    Nullable<PropertyPermissionsResponse
[]>,
      items: Nullable<CompositionPropertyPermissionsResponse []>,
}
```

Описание параметров:

- «**name**»* – имя поля карточки (например, «comment»);
- «**editable**»* – возможность редактирования значения поля карточки;
- «**visible**»* – видимость поля карточки;
- «**required**» – обязательность заполнения значения поля карточки;
- «**new_item_properties**» – информация видимости/редактируемости параметров нового значения поля карточки с типом «composition».

  Описание формата «PropertyPermissionsResponse» представлено в п.п. 16.37.
- «**items**» – информация видимости/редактируемости параметров значений поля карточки с типом «composition».

  Описание формата «CompositionPropertyPermissionsResponse» представлено в п.п. 16.19.

## 16.38. PropertyValueRequest

Формат:

```
{
      id: string,
      properties: CardPropertyRequest[].
}
```

Описание параметров:

- «**id**» – ID значения composition поля.
- «**properties**» – массив объектов, представляющий собой информацию значений «composition» значения поля.

Описание формата «CardPropertyRequest» представлено в п.п. 16.13.

## 16.39. PropertyValueResponse

Формат:

```
{
      id: string,
      description: string,
      title: string,
      properties: CardPropertyResponse [],
}
```

Описание параметров:

- «**id**»* – ID объекта значения поля.
- «**description**»* – текстовое представление значения ссылочного поля, поддерживающее html-разметку и используемое при отображении значения.

  Исключением являются справочники пользователей и сотрудников, в отображении значений которых данное поле игнорируется и используется информация, взятая из справочников.
- «**title**» – заголовок значения, отображаемый при проваливании в просмотр/редактирование табличного значения.
- «**properties**» – массив полей, отображаемых при проваливании в просмотр/редактирование табличного значения.

  Описание формата «CardPropertyResponse» представлено в п.п. 16.14.

Значения полей имеют данный формат в случае, если тип поля является типом справочника или таблицы.

**Пример:**

Если тип поля «sec$User», то формат значения поля карточки будет «PropertyValueResponse», где «id» это ID пользователя в справочнике.

## 16.40. RolesPropertyRequest

Формат:

```
{
      name: string,
      old_roles: OldRolesValueRequest [],
      new_roles: NewRolesValueRequest [],
      removed_roles: string [],
}
```

Описание параметров:

- «**name**»* – имя поля карточки (поля назначения ролей);
- «**old_roles**»* – массив объектов, представляющий информацию ролей карточки «roles» из «CardResponse».

  Описание формата «OldRolesValueRequest» представлено в п.п. 16.31.
- «**new_roles**»* – массив объектов, представляющий информацию ролей карточки после внесения изменений пользователем.

  Описание формата «NewRolesValueRequest» представлено в п.п. 16.30.
- «**removed_roles**» * – массив ID ролей карточки, которые были исключены при внесении изменений пользователем.

## 16.41. SearchCardResponse

Формат:

```
{
      card: CardResponse,
      hits: SearchHitResponse [],
}
```

Описание параметров:

- «**card**»* – объект карточки.

  Описание формата «CardResponse» представлено в п.п. 16.16.

- «**hits**»* – массив описаний признаков, по которым карточка подходит под условие поиска (на данный момент не используется в МП).

  Описание формата «SearchHitResponse» представлено в п.п. 16.42.

## 16.42. SearchHitResponse

Формат:

```
{
    property_caption: string,
    pharse_text: string,
}
```

Описание параметров:

- «**property_caption**»* – имя параметра в значении которого было найдено соответствие;
- «**pharse_text**»* – текст соответствия.

## 16.43. SignPropertyRequest

Формат:

```
{
    name: string,
    certificate: string,
    signed_properties: string [],
    attachments_signs: { [id: string]: string },
    card_sign: string,
}
```

Описание параметров:

- «**name**»* – имя поля (в данном случае «sign_data);
- «**certificate**»* – сертификат, которым были подписаны хэши карточки;
- «**signed_properties**»* – массив имен полей карточки, которые были учтены при формировании хэша полей карточки который был подписан;

- «**attachments_signs**»* – объект, представляющий «map», в котором «ключ» это ID вложения, а значение подписанный хэш в формате «base64» этого вложения;
- «**card_sign**» – подписанный хэш полей карточки в формате «base64».

  Если параметр «sign_card» не равен «true», то данный параметр не используется.

## 16.44. StartProcessParams

Формат:

```
{
      handler_id: string,
      action_name: string,
      process_id: string,
      form_commit_data:            (FormPropertyRequest            |
RolesPropertyRequest | SignPropertyRequest) [],
}
```

Описание параметров:

- «**handler_id**»* – ID обработчика процессного действия на стороне сервера.

  Возможные значения:

  → «id» из TransitionFormResponse;

  → «handler_id» из TransitionDialogResponse;

  → «handler_id» из TransitionNotificationResponse.
- «**action_name**»* – имя процессного действия name из «ProcessTransitionResponse» запускаемого процесса;
- «**process_id**»* – ID метаданных запускаемого процесса;
- «**form_commit_data**»* – значения полей формы запуска процесса.

  Может быть пустым массивом, если форма в назначении отсутствует и вместо неё отображается диалоговое окно или нотификация.

  Описание формата «FormPropertyRequest» представлено в п.п. 16.25.

  Описание формата «RolesPropertyRequest» представлено в п.п. 16.40.

  Описание формата «SignPropertyRequest» представлено в п.п. 16.43.

## 16.45. StartResolutionProcessParams

Формат:

```
{
      handler_id: string,
      assignment_id: string,
      action_name: string,
      process_id: string,
      form_commit_data:            (FormPropertyRequest            |
RolesPropertyRequest | SignPropertyRequest) [],
}
```

Описание параметров:

- «**handler_id**»* – ID обработчика процессного действия на стороне сервера.

  Возможные значения:

  - «id» из TransitionFormResponse;
  - «handler_id» из TransitionDialogResponse;
  - «handler_id» из TransitionNotificationResponse.

- «**assignment_id**»* – ID назначения процессного действия из «CardActionResponse»;
- «**action_name**»* – имя процессного действия «name» из «ProcessTransitionResponse» запускаемого процесса;
- «**process_id**»* – ID метаданных запускаемого процесса;
- «**form_commit_data**»* – значения полей формы запуска процесса.

  Может быть пустым массивом, если форма в назначении отсутствует и вместо неё отображается диалоговое окно или нотификация.

  Описание формата «FormPropertyRequest» представлено в п.п. 16.25.

  Описание формата «RolesPropertyRequest» представлено в п.п. 16.40.

  Описание формата «SignPropertyRequest» представлено в п.п. 16.43.

## 16.46. TabPermisionsResponse

Формат:

```
{
      name: string,
      editable: string,
```

```
    visible: string,
}
```

Описание параметров:

- «**name**»* – имя вкладки карточки (например, «Details»);
- «**editable**»* – возможность редактирования вкладки.

  Однако, если указать «false», а в каком-либо properties указать «true», то будет считаться, что вкладка редактируемая и для редактирования доступно одно указанное поле.
- «**visible**»* – видимость вкладки.

## 16.47. TransitionDialogResponse

Формат данных:

```
{
    handler_id: string,
    title: string,
    body: string,
    cancel_caption: string,
    ok_caption: string,
}
```

Описание параметров:

- «**handler_id**»* – ID обработчика процессного действия на стороне сервера, который необходим для указания в запросе на выполнение процессного действия;
- «**title**»* – заголовок диалогового окна;
- «**body**»* – сообщение диалогового окна;
- «**cancel_caption**»* – кнопка отмены диалогового окна;
- «**ok_caption**»* – кнопка подтверждения диалогового окна.

## 16.48. TransitionFormResponse

Формат данных:

```
{
    id: string,
    fields: FormFieldResponse [],
```

```
    signature_settings: FormSignatureSettingsResponse,
}
```

Описание параметров:

- «**id**»* – ID процессной формы, а также ID обработчика процессного действия на стороне сервера, который необходим для указания в запросе на выполнение процессного действия;
- «**fields**»* – метаинформация о формате полей процессной формы;
- «**signature_settings**» – информация о возможности выполнения подписи КриптоПро в процессном действии.

  Описание формата «FormSignatureSettingsResponse» представлено в п.п. 16.26.

## 16.49. TransitionNotificationResponse

Формат данных:

```
{
    handler_id: string,
    message: string,
}
```

Описание параметров:

- «**handler_id**»* – ID обработчика процессного действия на стороне сервера, который необходим для указания в запросе на выполнение процессного действия;
- «**message**»* – сообщение, которое будет выведено пользователю после выполнения процессного действия.

## 16.50. UnattachedCardDiffResponse

Формат данных:

```
{
    updated_card_ids: string [],
    removed_card_ids: string [],
}
```

Описание параметров:

- «**updated_card_ids**»* – массив ID карточек, которые не привязаны к папкам действий, не имеют пометку «Важное», но были указаны в теле запроса как необходимые для МП и были изменены;

- «**removed_card_ids**»* – массив ID карточек, которые были указаны в теле запроса как необходимые для МП и были удалены.

## Термины и сокращения

| Сокращение | Описание |
|---|---|
| МП | Мобильное приложение |
| ОС | Операционная система |
| Система | СЭД ТЕЗИС, система ТЕЗИС |
| API | Application programming interface – Программный интерфейс приложения |
| Поле с типом «composition» | «Composition» означает наличие вложенных properties в значения данного поля |
| Basic-авторизация | Базовая авторизация |
| Bearer-авторизация | Авторизация токена |
| Callback-функция | Функция обратного вызова |
| ID | Identifier – Идентификатор |
| IMEI | International Mobile Equipment Identity – Международный идентификатор мобильного оборудования |
| JSON | JavaScript Object Notation – Текстовый формат обмена данными, основанный на JavaScript |
| REST | REpresentational State Transfer – Архитектурный стиль взаимодействия компонентов распределённого приложения в сети |
| URL | Uniform Resource Locator – Адрес ресурса в сети Интернет |

ТЕЗИС

Система управления документами и задачами

Версия 5.22

Руководство по API

Дата выхода: Июль 2024 года

Головной офис:

443090, Россия, г. Самара, ул. Гастелло, д. 43а

Телефон: +7 (846) 273 94 89

+7 (846) 273 94 90

Сайт: https://haulmont.ru

Почта: info@haulmont.com

Информация о региональных офисах и торговых представительствах размещена на официальном сайте компании.