Версия 4.5
21.07.17

API «РЕВЕРС 8000»

1 Оглавление

2 Общие положения

В данном документе рассматривается программный интерфейс к СКУД «РЕВЕРС 8000», позволяющий решить следующие задачи:

1. Получать информацию о точках доступа, охранных и пожарных ШС, разделах;
2. Получать информацию о пространственных зонах;
3. Получать информацию о расписаниях на точках доступа;
4. Создавать, изменять, удалять пользователей системы;
5. Создавать, изменять, удалять записи в дереве подразделений;
6. Создавать, изменять, удалять записи в справочнике должностей;
7. Выдавать пользователю карту доступа с правами по доступу. Разрешать и запрещать карту. Удалять карту.
8. Назначать пользователю пин-код. Изменять пин-код, удалять пин-код.
9. Получать список типов праздничных дней. Создавать, изменять, удалять записи в справочнике типов праздничных дней.
10. Получать календарь праздников. Создавать, изменять, удалять записи в календаре праздников.
11. Получать в реальном времени события, вырабатываемые системой.
12. Получать события по запросу из журнала событий системы.

Вопросы первоначальной синхронизации данных остаются за рамками данного документа. Т.е. если внешняя система, основанная на данном API, вводится в действие на объекте, ранее оборудованном СКУД «РЕВЕРС 8000», то необходимо будет произвести первоначальное получение внешней системой данных от СКУД «РЕВЕРС 8000». Данные вопросы предлагается отнести к категории вопросов внедрения.

3 Особенности взаимодействия

Предлагается весь обмен осуществлять по протоколу TCP. При этом на стороне СКУД будет находиться TCP-сервер (далее – **Сервер или СКУД**), а со стороны внешнего ПО - клиент (далее – **Клиент или ВС - внешняя система**).
Данные предлагается передавать в формате JSON.
Перед началом обмена необходимо предусмотреть процедуру аутентификации.
Сервер реализуется для ОС MS Windows (как сервис) и для OS Linux (как демон).

4 Аутентификация

Предлагается реализовать аутентификацию по ассиметричному ключу и кодовой фразе с помощью протокола TLS v1, что также сразу решает проблему с шифрованием трафика.

Процесс получения доступа происходит следующим образом:

1. Каждый из участников обмена генерирует пару ключ-сертификат с помощью утилиты openssl. Также возможен случай, когда сервер генерирует обе пары и передаёт одну клиенту.
2. Клиент с сервером обмениваются сертификатами по любому доступному каналу связи (e-mail, флешка, и т.д.).
3. Клиент пытается подключиться к серверу через tls-сокет, используя свой сертификат в качестве локального, серверный – в качестве удалённого.
4. При невозможности авторизации одного из участников возникает ошибка tls-handshake, и соединение обрывается. Если авторизация успешна, дальнейший обмен происходит по зашифрованному tls-каналу.

Данный подход позволяет серверу в любой момент отзывать клиентские сертификаты (например, если есть подозрение, что он был скомпрометирован), а также обеспечивает надёжную криптографическую защиту.

Для генерации надо использовать опции:

-new rsa:2048 – алгоритм RSA с длиной ключа 2048 бит;

-sha256 – использовать SHA-256 а не SHA-1 для hash.

Что касается длительности действия для x509, то для тестовых целей можно поставить 3650 (10 лет).

При использовании TLS, нам не нужна внутренняя аутентификация, поэтому после соединения можно сразу посылать сообщения.

5 Формат команд

Каждая команда представляет собой JSON-объект, который передается в виде последовательности байт, перед которой идет ее длина (4 байта, младшими байтами вперед).

Все объекты передаются в кодировке UTF-8.

Формат команды:

{
“Command”: наименование команды,
“Id”: уникальный идентификатор команды,
“Version”: версия команды,
… - здесь какие-то другие данные, в зависимости от наименования команды
}

Версия команды обозначается целым числом. Она изменяется, если в формат передачи данной команды вносятся изменения. По-умолчанию для всех команд установим версию 1.

6 Команды

Примечание: все идентификатора, передаваемые от внешней системы (**ВС**) в **СКУД**, для СКУД являются внешними идентификаторами, и сохраняются в полях *_EXT_ID соответствующих таблиц.

6.1 +Пинг

Команда нужна для предотвращения разрыва TCP-соединения по таймауту. Команда отправляется **Сервером** раз в N секунд.

Формат команды от **Сервера**:

{
“Command”: “ping”,
“Id”: …,
“Version”: версия команды
}

Ответ **Клиента**:

{
“Command”: “ping”,
“Id”: тот же, что передан сервером
“Version”: версия команды
}

6.2 +Передача списка пространственных зон

Информирование **ВС** о пространственных зонах, имеющихся в **СКУД**. Необходим, поскольку информация о ТД ссылается на Id зон из данного списка.

Изначальный запрос на получение списка должен исходить от **ВС**.

В ответ СКУД передает список зон. Для каждой пространственной зоны передаются id, номер и наименование. Зона номер "0" считается территорией вне объекта, контролируемого системой.

Формат запроса от **ВС**:

{
“Command”: “zonelist”,
“Id”: …,
“Version”: версия команды
}

Ответ от **СКУД**:

{
“Command”: “zonelist”,
“Id”: тот же, что передан сервером,
“Version”: версия команды,
“Data”: [
{
“Id”: идентификатор зоны 0,
“Number”: номер зоны 0 (0),
“Name”: наименование зоны 0
},
…,
{
“Id”: идентификатор зоны N,
“Number”: номер зоны N (N),
“Name”: наименование зоны N

```
                }
        ]
}
```

6.3 +Оповещение об изменении списка пространственных зон

Информирование **ВС** об изменениях в пространственных зонах, имеющихся в **СКУД**. Выполняется по инициативе **СКУД**. Формат запроса от **ВС**:

```
{
“Command”: “zonedata”,
“Id”: …,
“Version”: версия команды,
“Data”: {
        “Id”: идентификатор зоны,
        “Number”: номер зоны,
        “Name”: наименование зоны,
        “Action”: действие (0, 1, 2)
        }
}
```

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

6.4 +Передача списка ТД

Команда для передачи списка ТД необходима для того, чтобы на стороне **ВС** можно было получить актуальную информацию по событиям **СКУД**, а также для задания прав пользователям по ТД.

Изначальный запрос на получение ТД должен исходить от ВС.

В ответ СКУД передает список ТД. Для каждой ТД передаются id, наименование, идентификатор пространственной зоны, из которой ведет точка доступа, и идентификатор пространственной зоны, в которую ведет точка доступа.

Формат запроса от **ВС**:

```
{
“Command”: “aplist”,
“Id”: …,
“Version”: версия команды
}
```

Ответ от **СКУД**:

```
{
“Command”: “aplist”,
“Id”: тот же, что передан сервером,
“Version”: версия команды,
“Data”: [
                {
```

“Id”: идентификатор ТД 1,
“Name”: наименование ТД 1,
“FromZone”: идентификатор пространственной зоны, ИЗ которой ведет ТД 1,
“ToZone”: идентификатор пространственной зоны, В которую ведет ТД 1
},
…,
{
“Id”: идентификатор ТД N,
“Name”: наименование ТД N,
“FromZone”: идентификатор пространственной зоны, ИЗ которой ведет ТД N,
“ToZone”: идентификатор пространственной зоны, В которую ведет ТД N
}
]
}

6.5 +Оповещение об изменении ТД

Команда отправляется из СКУД в ВС тогда, когда происходит изменение, добавление или удаление ТД.

Формат команды от **СКУД**:

{
“Command”: “apdata”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“Id”: идентификатор ТД,
“Name”: наименование ТД,
“FromZone”: идентификатор пространственной зоны, ИЗ которой ведет ТД,
“ToZone”: идентификатор пространственной зоны, В которую ведет ТД,
“Action”: действие с ТД
}
}

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

6.6 +Передача списка охранных ШС

Необходима для того, чтобы на стороне **ВС** можно было получить актуальную информацию по событиям **СКУД**.

Изначальный запрос должен исходить от **ВС**.

Формат запроса от **ВС**:

{
“Command”: “glooplist”, //guard loop
“Id”: …,
“Version”: версия команды
}

Ответ от **СКУД**:

{
“Command”: “glooplist”,
“Id”: тот же, что передан сервером,
“Version”: версия команды,
“Data”: [
{
“Id”: идентификатор 1,
“Type”: тип 1,
“Name”: наименование 1,
},
…,
{
“Id”: идентификатор N,
“Type”: тип N,
“Name”: наименование N,
}
]
}

Типы:
1,2,3 - типы шлейфов «РЕВЕРС» в соответствии с таблицей LoopTypes;
101600 - охранный шлейф «ОРИОН» («Болид»)

6.7 +Оповещение об изменении списка охранных ШС

Команда отправляется из СКУД в ВС тогда, когда происходит изменение, добавление или удаление охранного ШС.

Формат команды от **СКУД**:

{
“Command”: “gloopdata”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“Id”: идентификатор,
“Name”: наименование,

“Action”: действие
}
}

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

6.8 +Передача списка пожарных ШС

Необходима для того, чтобы на стороне **ВС** можно было получить актуальную информацию по событиям **СКУД**.

Изначальный запрос должен исходить от **ВС**.

Формат запроса от **ВС**:

```
{
“Command”: “flooplist”,          //fire loop
“Id”: …,
“Version”: версия команды
}
```

Ответ от **СКУД**:

```
{
“Command”: “flooplist”,
“Id”: тот же, что передан сервером,
“Version”: версия команды,
“Data”: [
                    {
                    “Id”: идентификатор  1,
                    “Type”: тип 1,
                    “Name”: наименование 1,
                    },
                    …,
                    {
                    “Id”: идентификатор N,
                    “Type”: тип N,
                    “Name”: наименование N,
                    }
          ]
}
```

Тип введен на тот случай, если мы не будем ограничиваться только пожарной сигнализацией «Орион», и если надо будет сообщить расширенные данные о ШС. Пожарный ШС «Орион» имеет тип «101500».

6.9 +Оповещение об изменении списка пожарных ШС

Команда отправляется из **СКУД** в **ВС** тогда, когда происходит изменение, добавление или удаление пожарного ШС.

Формат команды от **СКУД**:

```
{
“Command”: “floopdata”,
“Id”: …,
“Version”: версия команды,
“Data”: {
                    “Id”: идентификатор,
                    “Name”: наименование,
```

“Action”: действие
}
}

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

6.10 +Передача списка входов/выходов

Необходима для того, чтобы на стороне **ВС** можно было получить актуальную информацию по событиям **СКУД**.

Изначальный запрос должен исходить от **ВС**.

Формат запроса от **ВС**:

{
“Command”: “iolist”
“Id”: …,
“Version”: версия команды
}

Ответ от **СКУД**:

{
“Command”: “iolist”,
“Id”: тот же, что передан сервером,
“Version”: версия команды,
“Data”: [
{
“Id”: идентификатор 1,
“Type”: тип (0 - вход, 1 - выход),
“Name”: наименование 1,
},
…,
{
“Id”: идентификатор N,
“Type”: тип (0 - вход, 1 - выход),
“Name”: наименование N,
}
]
}

6.11 +Оповещение об изменении списка входов/выходов

Команда отправляется из **СКУД** в **ВС** тогда, когда происходит изменение, добавление или удаление логического входа или выхода.

Формат команды от **СКУД**:

{
“Command”: “iodata”,

“Id”: …,
“Version”: версия команды,
“Data”: {
“Id”: идентификатор,
“Type”: тип (0 - вход, 1 - выход),
“Name”: наименование,
“Action”: действие
}
}

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

6.12 +Передача списка разделов

Необходима для того, чтобы на стороне **ВС** можно было получить актуальную информацию по событиям **СКУД**.

Изначальный запрос должен исходить от **ВС**.

Формат запроса от **ВС**:

{
“Command”: “sectionlist”,
“Id”: …,
“Version”: версия команды
}

Ответ от **СКУД**:

{
“Command”: “sectionlist”,
“Id”: тот же, что передан сервером,
“Version”: версия команды,
“Data”: [
{
“Id”: идентификатор 1,
“Name”: наименование 1,
},
…,
{
“Id”: идентификатор N,
“Name”: наименование N,
}
]
}

6.13 +Оповещение об изменении списка разделов

Команда отправляется из СКУД в ВС тогда, когда происходит изменение, добавление или удаление раздела.

Формат команды от **СКУД**:

{
“Command”: “sectiondata”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“Id”: идентификатор,
“Name”: наименование,

“Action”: действие
}
}

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

6.14 +Передача списка шаблонов прав

Команда передачи списка шаблонов прав необходима для того, чтобы на стороне ВС оператор мог выбрать, какой шаблон прав задать карте.

Изначальный запрос на получение шаблонов должен исходить от ВС.

В ответ СКУД передает данные о шаблонах прав. Для каждого шаблона передается id и наименование.

Формат запроса от **ВС**:

{
“Command”: “rightslist”,
“Id”: …,
“Version”: версия команды
}

Ответ от **СКУД**:

{
“Command”: “rightslist”,
“Id”: тот же, что передан клиентом,
“Version”: версия команды,
“Data”: [
{
“Id”: идентификатор шаблона прав 1,
“Name”: наименование шаблона прав 1

```
                },
                …,
                {
                “Id”: идентификатор шаблона прав N,
                “Name”: наименование шаблона прав N
                }
        ]
}
```

6.15 +Оповещение об изменении шаблона прав

Команда отправляется из СКУД в ВС тогда, когда происходит изменение, добавление или удаление шаблона прав.

Формат команды от **СКУД**:

```
{
“Command”: “rightsdata”,
“Id”: …,
“Version”: версия команды,
“Data”: {
                “Id”: идентификатор шаблона прав,
                “Name”: наименование шаблона прав,
                “Action”: действие с шаблоном прав
        }
}
```

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

В данном случае при удалении происходит полное удаление из БД, а не отправка в архив. Это означает, что сервер API, получив оповещение об удалении такого Id, не сможет ничего прочитать из БД. В этом случае Name будет “”. Это является дополнительным указанием на то, что данные удалены. Ориентируясь на переданный Id, клиент может удалить данные в своих списках, полученных по команде 6.14.

6.16 +Запрос расписаний на ТД

Команда отправляется от **ВС** к **СКУД** и предназначена для получения списка расписаний для данной ТД (на самом деле, списка названий расписаний для сопоставления их с номерами расписаний). Список расписаний используется при задании прав карты (при способе задания прав без использования шаблонов, определенных в СКУД).

В ответ СКУД должен сдать список расписаний для данной ТД.

Формат запроса от **ВС**:

```
{
“Command”: “timetablelist”,
“Id”: …,
“Version”: версия команды,
“ApId”: Id точки доступа
```

}

Ответ от **СКУД**:

{
“Command”: “timetablelist”,
“Id”: тот же, что передан ВС,
“Version”: версия команды,
“ApId”: Id точки доступа,
“Data”: [
{
“Id”: идентификатор расписания, // необходим для поддержки команды оповещения
“Number”: номер расписания (1),
“Name”: наименование расписания 1,
“Type”: тип расписания 1 (0 - недельное, 1 - сменное),
“Days”: длина расписания 1 в днях (без учета праздников, 7 для недельного)
“BeginDate”: дата начала расписания 1 - строка в формате “dd.mm.yyyy”), актуально, если сменное),
},
…,
{
“Id”: идентификатор расписания,
“Number”: номер расписания (N),
“Name”: наименование расписания N,
“Type”: тип расписания N (0 - недельное, 1 - сменное),
“Days”: длина расписания N в днях (без учета праздников, 7 для недельного)
“BeginDate”: дата начала расписания N - строка в формате “dd.mm.yyyy”), актуально, если сменное),
}
]
}

6.17 +Оповещение об изменении расписаний на ТД

Информирование **ВС** об изменениях в списке расписаний, имеющихся в **СКУД**. Выполняется по инициативе **СКУД**. Оповещение проходит по конкретному расписанию, связано с удалением расписания, изменением его номера и/или наименования:

{
“Command”: “timetabledata”,
“Id”: …,
“Version”: версия команды,
“Data”: {
//“ApId”: идентификатор ТД, - вообще говоря, это лишние данные, достаточно Id расписания - удалил! - номер и наименование для контроля - что изменилось
“Id”: идентификатор расписания,
“Number”: номер расписания,
“Name”: наименование расписания,
“Type”: тип расписания (0 - недельное, 1 - сменное),
“Days”: длина расписания в днях (без учета праздников, 7 для недельного)
“BeginDate”: дата начала расписания - строка в формате “dd.mm.yyyy”), актуально, если сменное),
“Action”: действие (1)

```
        }
}
```

Возможные значения поля “Action”:

- Данные изменены: 1.

Обратите внимание - только изменение! Количество расписаний для ТД всегда фиксировано!

6.18 +Оповещение об изменении состава расписания на конкретной ТД*

Информирование **ВС** об изменениях в конкретном расписании. Выполняется по инициативе **СКУД**. Не связано с удалением расписания, изменением его номера и/или наименования, его типа (недельное, сменное) и даты начала - в этом случае см.п. 6.17. Связано с изменением типа расписания и содержимого расписания. В этой связи нет необходимости в поле Action, т.к. данное оповещение всегда связано с изменением.

В этой связи выделенное красным удаляем - Id точки доступа просто не нужен, т.к. может быть получен из запроса расписаний для ТД, а остальные данные логично отнести к изменению параметров расписания. Идентификатор расписания выносим из списка в отдельную пару. Тогда Data - как обычно, массив.

Возвращаются всегда все дни расписания.

```
{
“Command”: “timetableedit”,
“Id”: …,
“TtId”:  идентификатор расписания,
“Version”: версия команды,
//“ApId”: идентификатор ТД,
/“Number”: номер расписания,
//“Name”: наименование расписания,
//“Type”: тип расписания (0 - недельное, 1 - сменное),
//“Days”: длина расписания в днях (без учета праздников, 7 для недельного)
//“BeginDate”: дата начала расписания - строка в формате “dd.mm.yyyy”), (актуально, если
//сменное),
“Data”: [
        {
                “Id”: идентификатор дня в расписании,
                “DayNumber”: номер дня в расписании (1-..., 0-праздник) (1, для недельного это
                понедельник, 7 - воскресенье),
                “HolidayType”: тип праздничного дня (в соответствии с таблицей типов, для каждого
                расписания присутствуют все типы),
                “Begin1”: начало интервала 1 (строка в формате “hh:mm”),
                “End1”: конец интервала 1,
                “Begin2”: начало интервала 2,
                “End2”: конец интервала 2,
                “Begin3”: начало интервала 3,
                “End3”: конец интервала 3,
                “Begin4”: начало интервала 4,
                “End4”: конец интервала 4
        }
        ,…,
        {
```

“Id”: идентификатор дня в расписании,
“DayNumber”: номер дня в расписании (1-..., 0-праздник) (Days),
“HolidayType”: тип праздничного дня (в соответствии с таблицей типов, для каждого расписания присутствуют все типы),
“Begin1”: начало интервала 1 (строка в формате “hh:mm”),
“End1”: конец интервала 1,
“Begin2”: начало интервала 2,
“End2”: конец интервала 2,
“Begin3”: начало интервала 3,
“End3”: конец интервала 3,
“Begin4”: начало интервала 4,
“End4”: конец интервала 4
}
]
}

Пояснения: если день -праздник (DayNumber = 0), то тип праздника определяется по полю HolidayType (1, 2 и т.д.).

6.19 +Редактирование расписания на ТД*

Выполняется по инициативе **ВС**. В **СКУД** передается измененное в **ВС** расписание.

{
“Command”: “edittimetable”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“ApId”: идентификатор ТД,
// Удалил идентификатор расписания, ВС не обязан его знать при изменении/добавлении
“Number”: номер расписания,
“Name”: наименование расписания,
“Type”: тип расписания (0 - недельное, 1 - сменное),
“Days”: длина расписания в днях (без учета праздников, 7 для недельного)
“BeginDate”: дата начала расписания - строка в формате “dd.mm.yyyy”), (актуально, если сменное),
“Data”: [
{
“DayNumber”: номер дня в расписании (1-..., 0-праздник) (1, для недельного это понедельник, 7 - воскресенье),
“HolidayType”: тип праздничного дня (в соответствии с таблицей типов, для каждого расписания присутствуют все типы),
“Begin1”: начало интервала 1 (строка в формате “hh:mm”),
“End1”: конец интервала 1,
“Begin2”: начало интервала 2,
“End2”: конец интервала 2,
“Begin3”: начало интервала 3,
“End3”: конец интервала 3,
“Begin4”: начало интервала 4,
“End4”: конец интервала 4
},
…,
{
“DayNumber”: номер дня в расписании (1-..., 0-праздник) (Days),

“HolidayType”: тип праздничного дня (в соответствии с таблицей типов, для каждого расписания присутствуют все типы),
“Begin1”: начало интервала 1 (строка в формате “hh:mm”),
“End1”: конец интервала 1,
“Begin2”: начало интервала 2,
“End2”: конец интервала 2,
“Begin3”: начало интервала 3,
“End3”: конец интервала 3,
“Begin4”: начало интервала 4,
“End4”: конец интервала 4
}
]
}
}

Ответ от **СКУД**:

{
“Command”: “edittimetable”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}

Примечание: после выполнения операции требуется загрузка контроллера. Нет смысла делать это сразу после выполнения данной операции автоматически, поскольку операция достаточно длительная, и при этом весьма вероятно, что ВС захочет еще что-то сделать с контроллером, например, поменять еще одно расписание (выполнить еще одну такую же команду). Поэтому добавлена явная команда «загрузить контроллер», причем загрузка может быть выполнена по Id точки доступа.

6.20 +Запрос справочника типов праздничных дней

Команда отправляется от **ВС** к **СКУД** и предназначена для получения справочника типов праздничных дней. По-умолчанию в системе есть один тип - "Праздничный день", он установлен как неудаляемый. Допустимо создание собственных типов - например, "Предпраздничный день", "Послепраздничный день" и т.п. Смысл - для каждого типа дня в расписании предусматриваются свои правила прохода через ТД.

В ответ СКУД должен сдать список типов.

Формат запроса от **ВС**:

{
“Command”: “holidaytypes”,
“Id”: …,
“Version”: версия команды,
}

Ответ от **СКУД**:

{

“Command”: “ holidaytypes ”,
“Id”: тот же, что передан ВС,
“Version”: версия команды,
“Data”: [
{
“Id”: Id типа праздничного дня (номер типа),
“Name”: наименование типа праздничного для,
“Default”: 1-предопределенный и удалению не подлежит, 0-введен пользователем
},
…,
{
“Id”: Id типа праздничного дня,
“Name”: наименование типа праздничного для,
“Default”: 1-предопределенный и удалению не подлежит, 0-введен пользователем
}
]
}

6.21 +Редактирование справочника типов праздничных дней*

Команда отправляется от **ВС** к **СКУД**.

Формат запроса от **ВС**:

{
“Command”: “editholidaytypes”,
“Id”: …,
“Version”: версия команды,
“Data”:{
“Id”: Id типа праздничного дня (номер типа),
“Name”: наименование типа праздничного для,
“Action”: действие: 0,1 или 2
}
}

Значения Action:

- Данные следует добавить: 0.
- Данные следует изменить: 1.
- Данные следует удалить: 2.

Ответ от **СКУД**:

{
“Command”: “ editholidaytypes ”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}

Примечание: Будет выдана ошибка, если при изменении или удалении указать несуществующий Id типа. При добавлении Id формируется автоматически, поэтому указывать можно какой угодно - на реальное добавление это не влияет - поэтому рекомендуется указывать 0, чтобы самим разработчикам клиента было понятно, что передается фиктивный Id. Узнать, что получилось после добавления, можно, выполнив запрос справочника типов праздничных дней.

6.22 +Оповещение об изменении справочника типа праздничных дней

Команда отправляется из **СКУД** в **ВС** тогда, когда происходит изменение, добавление или удаление типов праздничных дней.

Формат команды от **СКУД**:

{
“Command”: “holidaytypesdata”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“Id”: Id типа праздничного дня (номер типа),
“Name”: наименование типа праздничного для,
“Action”: 0,1 или 2
}
}

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

В данном случае при удалении происходит полное удаление из БД, а не отправка в архив. Это означает, что сервер API, получив оповещение об удалении такого Id, не сможет ничего прочитать из БД. В этом случае Name будет “”. Это является дополнительным указанием на то, что данные удалены. Ориентируясь на переданный Id, клиент может удалить данные в своих списках, полученных по команде 6.20.

6.23 +Запрос списка праздничных дней (календаря праздников)

Команда отправляется от **ВС** к **СКУД** и предназначена для получения списка праздников. Для каждого праздника указывается его тип.

В ответ СКУД должен сдать список праздничных дней.

Формат запроса от **ВС**:

{
“Command”: “holidays”,
“Id”: …,
“Version”: версия команды,
}

Ответ от **СКУД**:

{
“Command”: “ holidays”,
“Id”: тот же, что передан ВС,
“Version”: версия команды,
“Data”: [
{
“Id”: Id праздничного дня,
“Type”: Id типа праздничного дня (см. выше список типов праздников),

“Date”: дата праздника в формате «дд.мм»
},
…,
{
“Id”: Id праздничного дня,
“Type”: Id типа праздничного дня (см. выше список типов праздников),
“Date”: дата праздника в формате «дд.мм»

}
]
}

6.24 +Редактирование списка праздничных дней (календаря праздников)*

Команда отправляется от **ВС** к **СКУД** и предназначена для изменения календаря праздников.

Формат запроса от **ВС**:

{
“Command”: “editholidays”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“Type”: Id типа праздничного дня,
“Date”: дата праздника в формате «дд.мм»,
“Action”: 0,1 или 2
}

}

Ответ от **СКУД**:

Значения Action:

- Данные следует добавить: 0.
- Данные следует изменить: 1.
- Данные следует удалить: 2.

Ответ от **СКУД**:

{
“Command”: “ editholidays ”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}

Примечание: Оперирование идет по дате, а не по Id, соответственно, несколько меняется логика по отношению к другим командам. Добавление и изменение - в данном случае одно и то же - в ответ на обе эти команды будет добавлен/изменен праздничный день. Удаление - если день не был праздничным, то все равно успех, ибо по результату успех - он не праздничный. Ошибка будет выдана, если при добавлении и изменении указать несуществующий Id типа. Ошибка будет выдана, если указать невалидную дату (например, «30.02»).

Примечание 2: После изменения календаря следует выполнить загрузку всех контроллеров. С целью оптимизации это не делается автоматически после выполнения каждой команды, ВС следует позаботиться о подаче данной команды явно.

6.25 +Оповещение об изменении списка праздничных дней (календаря праздников)

Команда отправляется из **СКУД** в **ВС** тогда, когда происходит изменение, добавление или удаление календаря праздничных дней.

Формат команды от **СКУД**:

{
“Command”: “holidaysdata”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“Id”: Id праздничного дня,
“Type”: Id типа праздничного дня (см. выше список типов праздников),
“Date”: дата праздника в формате «дд.мм»
“Action”: 0,1 или 2
}
}

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

В данном случае при удалении происходит полное удаление из БД, а не отправка в архив. Это означает, что сервер API, получив оповещение об удалении такого Id, не сможет ничего прочитать из БД. В этом случае тип будет 0, строка даты будет содержать «00.00». Это является дополнительным указанием на то, что данные удалены. Ориентируясь на переданный Id, клиент может удалить данные в своих списках, полученных по команде 6.24.

6.26 +Запрос списка должностей

Команда отправляется от **ВС** к **СКУД** и предназначена для верификации списка должностей. Например, таким образом ВС может узнать, что справочник должностей в СКУД не пуст, и, следовательно, требуется первоначальная синхронизация данных от СКУД к ВС (выходит за рамки данного документа).

В ответ СКУД должен сдать список должностей. Для каждой должности передаются id,ext_id (идентификатор записи в ВС) и наименование.

Формат запроса от **ВС**:

{
“Command”: “joblist”,
“Id”: …,
“Version”: версия команды

}

Ответ от **СКУД**:

{
“Command”: “joblist”,
“Id”: тот же, что передан сервером,
“Version”: версия команды,
“Data”: [
{
“Id”: идентификатор должности 1,
“Ext_Id”: идентификатор должности 1 в ВС,
“Name”: наименование должности 1
},
…,
{
“Id”: идентификатор должности N,
“Ext_Id”: идентификатор должности N в ВС,
“Name”: наименование должности N
}
]
}

6.27 +Редактирование должности

Отправляется от ВС в СКУД при добавлении, изменении или удалении должности. Направление передачи именно такое, т.к. предполагается, что редактирование происходит на стороне ВС.

Формат команды от **ВС**:

{
“Command”: “jobedit”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“Id”: идентификатор должности в ВС (для СКУД это - EXT_ID),
“Name”: наименование должности,
“Action”: действие
}
}

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

Примечание: если приходит запрос на изменение данных, а в СКУД такой должности нет (при нормальной синхронизации такая ситуация невозможна), она автоматически создается.

Примечание 1: при удалении можно передавать только Id и Action.

Примечание 2: если приходит запрос на добавление должности с уже известным ext_id, будет выполнено редактирование данной записи.

Ответ от **СКУД**:

{
“Command”: “jobedit”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}

6.28 +Оповещение об изменении списка должностей в СКУД

Новое - изменено направление команды; теперь, как и в других случаях, это изменение от СКУД в ВС; то, что по инициативе ВС, называется ___edit (см. выше).

Вводится для полноты API, поскольку можно предположить, что не всегда изменения будут со стороны ВС, могут быть и со стороны СКУД.

Отправляется от ВС в СКУД при добавлении, изменении или удалении должности. Направление передачи именно такое, т.к. предполагается, что редактирование происходит на стороне ВС.

Формат команды от **ВС**:

{
“Command”: “jobdata”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“Id”: идентификатор должности в СКУД,
“Ext_Id”: идентификатор должности в ВС,
“Name”: наименование должности,
“Action”: действие
}
}

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

В данном случае при удалении происходит полное удаление из БД, а не отправка в архив. Это означает, что сервер API, получив оповещение об удалении такого Id, не сможет ничего прочитать из БД. В этом случае Ext_Id будет 0, Name будет “”. Ориентируясь на переданный Id, клиент может удалить данные в своих списках, полученных по команде 6.26.

6.29 +Запрос дерева подразделений

Команда отправляется от ВС к СКУД и предназначена для верификации дерева подразделений. Например, таким образом ВС может узнать, что справочник подразделений в СКУД не пуст, и, следовательно, требуется первоначальная синхронизация данных от СКУД к ВС (выходит за рамки данного документа).

В ответ СКУД должен сдать дерево подразделений. Для каждого подразделения передаются id, ext_id, наименование и id родителя. Для корневых узлов в качестве родителя передается 0 (будет необходима интеллектуальная модификация, потому что у нас передается сам себе родитель).

Формат запроса от **ВС**:

{
“Command”: “deplist”,
“Id”: …,
“Version”: версия команды
}

Ответ от **СКУД**:

{
“Command”: “deplist”,
“Id”: тот же, что передан сервером,
“Version”: версия команды,
“Data”: [
{
“Id”: идентификатор подразделения 1,
“Ext_Id”: идентификатор подразделения 1 в ВС,
“Name”: наименование подразделения 1,
“ParentId”: идентификатор родителя подразделения 1,
“Parent_Ext_Id”: идентификатор родителя подразделения 1 в ВС,

},
…,
{
“Id”: идентификатор подразделения N,
“Ext_Id”: идентификатор подразделения N в ВС,
“Name”: наименование подразделения N,
“ParentId”: идентификатор родителя подразделения N,
“Parent_Ext_Id”: идентификатор родителя подразделения N в ВС,
}
]
}

6.30 +Оповещение об изменении дерева подразделений

Новое - изменено направление команды; теперь, как и в других случаях, это изменение от СКУД в ВС; то, что по инициативе ВС, называется ___edit (см. далее).

Отправляется от СКУД в ВС при добавлении, изменении или удалении подразделения.

Формат команды от **СКУД**:

{
“Command”: “depdata”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“Id”: идентификатор подразделения (в СКУД),
“Ext_Id”: идентификатор подразделения в ВС,
“Name”: наименование подразделения,

“ParentId”: идентификатор родителя подразделения,
“Parent_Ext_Id”: идентификатор родителя подразделения 1 в ВС,
“Action”: действие
}
}

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

Замечание: удаление в данном случае происходит насовсем, без архивирования, поэтому при удалении (Action = 2) значим только Id, остальные данные обнулены.

6.31 +Редактирование подразделения

Новое

Отправляется от ВС в СКУД при добавлении, изменении или удалении подразделения. Чем вызывается соответствующее редактирование в СКУД.

Формат команды от **ВС**:

{
“Command”: “depedit”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“Id”: идентификатор подразделения в ВС,
“ParentId”: идентификатор родителя подразделения в ВС (передаем 0 для корневого узла),
“Name”: наименование подразделения,
“Action”: действие
}
}

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

Ответ от **СКУД**:

{
“Command”: “depedit”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}

Примечание: если приходит запрос на изменение данных, а в СКУД такого подразделения нет, (при нормальной синхронизации такая ситуация невозможна), возвращается ошибка. Связано это

в первую очередь с тем, что данные о родительском подразделении тоже могут содержать ошибку - непонятно, куда его надо будет добавлять. В этой связи в обоих случаях (нет подразделения, нет родительского подразделения) запрос на изменение вернет ошибку.

Примечание 1: запрос на добавление вернет ошибку, если уже есть такой id, или если нет id родителя.

Примечание 2: при удалении можно передавать только Id и Action. Вернется ошибка при отсутствии id.

6.32 +Запрос списка дополнительных свойств пользователя

Команда служит для получения списка дополнительных свойств пользователя с целью правильной их интерпретации и отображения.

Формат команды от **ВС**:

{
“Command”: “userproplist”,
“Id”: …,
“Version”: версия команды
}

Ответ **СКУД**:

{
“Command”: “userproplist”,
“Id”: тот же, что передан ВС,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее). Если код ошибки = 0, или ответ вообще не содержит записи с кодом ошибки, то в ответе содержится список дополнительных свойств пользователя:
“Data”: [
{
“Id”: ID записи в таблице «OWNER_PROP_LIST»,
“Property”: имя свойства (то, по которому ищем в таблице свойств пользователя) (поле «OPLIST_PROPERTY»),
“Text”: текст, выводимый при данном свойстве («OPLIST_TEXT»),
“Type”: тип значения свойства (0-строка, 1-число, 2 - дата) («OPLIST_TYPE»), число,
“Pos”: позиция свойства в отображаемом списке («OPLIST_POS»), число,
“Default”: значение по умолчанию, строка,
“Mask”: маска ввода, строка,
“OwType”: для какого типа пользователя задано свойство, число (1 - пользователь, 2 - посетитель)
},
{
...
}
]
}

6.33 +Оповещение об изменении списка дополнительных свойств пользователя

Новое ; как и в других случаях, это изменение от СКУД в ВС; то, что по инициативе ВС, называется edit (см. далее).

Отправляется от СКУД в ВС при добавлении, изменении или удалении подразделения.

Формат команды от **СКУД**:

{
“Command”: “userpropdata”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“Id”: ID записи в таблице «OWNER_PROP_LIST»,
“Property”: имя свойства (то, по которому ищем в таблице свойств пользователя) (поле «OPLIST_PROPERTY»),
“Text”: текст, выводимый при данном свойстве («OPLIST_TEXT»),
“Type”: тип значения свойства (0-строка, 1-число, 2 - дата) («OPLIST_TYPE»), число,
“Pos”: позиция свойства в отображаемом списке («OPLIST_POS»), число,
“Default”: значение по умолчанию, строка,
“Mask”: маска ввода, строка,
“OwType”: для какого типа пользователя задано свойство, число (1 - пользователь, 2 - посетитель),
“Action”: действие
}
}

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

Замечание: удаление в данном случае происходит насовсем, без архивирования, поэтому при удалении (Action = 2) значим только Id, остальные данные обнулены.

6.34 +Редактирование списка дополнительных свойств пользователя

При изменении записи списка важно иметь в виду, что не допускается редактирование имени свойства, поскольку оно присваивается автоматически. Соответственно, имя свойства не включено в передаваемую информацию. Для получения имени свойства следует запросить список командой “userproplist” и произвести сопоставление по Id.

Формат команды от **ВС**:

{
“Command”: userpropedit”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“Id”: ID записи в таблице «OWNER_PROP_LIST»,
“Text”: текст, выводимый при данном свойстве («OPLIST_TEXT»),
“Type”: тип значения свойства (0-строка, 1-число, 2 - дата) («OPLIST_TYPE»), число,

“Pos”: позиция свойства в отображаемом списке («OPLIST_POS»), число,
“Default”: значение по умолчанию, строка,
“Mask”: маска ввода, строка,
“OwType”: для какого типа пользователя задано свойство, число (1 - пользователь, 2 - посетитель),
“Action”: действие
}
}

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

Ответ от **СКУД**:

{
“Command”: “ userpropedit”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}

Примечание: если приходит запрос на изменение данных, а в СКУД такого описания свойства нет, выдается ошибка.

Примечание 2: поля “Pos”, “Default”, “Mask” не являются обязательными. Для удаления достаточно передать только “Id” записи.

Примечание 3: для добавления (“Action”: действие = 0) следует передавать Id = 0. В противном случае будет сгенерирована ошибка. На самом деле Id будет сформировано автоматически в процессе добавления в БД. Поскольку не предполагалось ранее, что список дополнительных свойств может быть задан извне (в **ВС** согласно терминологии данного документа), то в таблице дополнительных свойств пользователя просто нет поля для Id из внешней системы. Поэтому идентифицировать добавленное придется по свойству “text”, перечитав данные командой “userproplist”.

6.35 +Запрос списка пользователей

Команда служит для получения ВС списка пользователей от СКУД с целью верификации. Поскольку списки пользователей, как правило, большие, при получении списка выдается только самая необходимая информация. Для получения подробной информации о пользователе служит команда “ userinfo ” - запрос информации о пользователе.

{
“Command”: “userlist”,
“Id”: …,
“Version”: версия команды,
}

Ответ **СКУД**:

{
“Command”: “userlist”,
“Id”: тот же, что передан клиентом,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее). Если код ошибки = 0, или ответ вообще не содержит записи с кодом ошибки, то в ответе содержится полная информация о пользователе:
“Data”: [
{
“Id”: ID пользователя в СКУД,
“GUID”: GUID пользователя - строка, вместе с фигурными скобками (как хранится в БД),
“ExtId”: ID во внешней системе
“Archive”: флаг архивности пользователя (0 - не архивен, 1 - архивен)
“Type”: тип пользователя (1-пользователь, 2-посетитель),
“LastName”: фамилия,
“FirstName”: имя,
“SecondName”: отчество,
“DepId”: id подразделения,
“JobId”: id должности,
“TabNum”: табельный номер (это строка!), // добавил, т.к. это - основные данные
},
{
...
}
]
}

6.36 +Запрос информации о пользователе

Команда служит для получения полной информации о пользователе.

Формат команды от **ВС**:

{
“Command”: “userinfo”,
“Id”: …,
“Version”: версия команды,
“UserExtId”: ID пользователя в ВС, о котором надо получить информацию (ExtID) или “UserId” - ID пользователя в СКУД, НО не одновременно. Сначала Сервер API проверит наличие пары с “UserExtId”, и, только если ее нет, с UserId
}

Ответ **СКУД**:

{
“Command”: “userinfo”,
“Id”: тот же, что передан клиентом,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее). Если код ошибки = 0, или ответ вообще не содержит записи с кодом ошибки, то в ответе содержится полная информация о пользователе:
“Data”: {“Id”: ID пользователя в СКУД,
“GUID”: GUID пользователя - строка, вместе с фигурными скобками (как хранится в БД),

“ExtId”: ID во внешней системе (сейчас передается как число, может быть изменено до строки)
“Archive”: флаг архивности пользователя (0 - не архивен, 1 - архивен)
“Type”: тип пользователя (1-пользователь, 2-посетитель),
“LastName”: фамилия,
“FirstName”: имя,
“SecondName”: отчество,
“DepId”: id подразделения,
“JobId”: id должности,
“TabNum”: табельный номер (это строка!),
“PhotoHash”: 8-байтный хеш фото на основе алгоритма CityHash. Если у пользователя фото нет, данная пара не передается. В качестве хеша используется CityHash, реализация приложена. Хеш-функция возвращает unsigned int64,
далее идут дополнительные поля в формате «ИМЯ СВОЙСТВА:ЗНАЧЕНИЕ», причем значения, независимо от типа дополнительного поля, всегда передаются строкой. При этом передаем и предопределенные, а не только назначенные пользователем свойства (выделены синим) Фото (OW_PHOTO) не передается:
“OW_WEIGHT”:”123”, // это вес пользователя, явно заданное
// разработчиками свойство
“workgroup”: “0”, // рабочая группа, , явно заданное разработчиками
// свойство
“OW_PROP_FROMLIST_1”: “123”,
“OW_PROP_FROMLIST_2”: “25.11.2016 18:51:33”,
и т.д. до конца списка дополнительных полей
}
}

Перечень дополнительных полей можно получить специальной командой (см. userproplist). Если у пользователя есть фото, и фото изменилось по сравнению с тем, что хранит клиент (изменился хеш от фото), клиент запрашивает фото (см. команду userphoto).

6.37 +Передача данных о пользователе

Отправляется от **ВС** в **СКУД** при добавлении, изменении или удалении пользователя.

От **ВС** в **СКУД** передаются следующие данные основные данные:

1. Тип пользователя
2. Фамилия
3. Имя
4. Отчество
5. Id подразделения
6. Id должности
7. Табельный номер*
8. Описание*
9. Хеш фотографии пользователя

При добавлении пользователя

Формат команды от **ВС**:

{
“Command”: “userdata”,
“Id”: …,
“Version”: версия команды,
“Data”: {

“Id”: Id пользователя в СКУД,
“ExtId”: идентификатор пользователя в ВС (сейчас передается как число, может быть изменено до строки),

НЕЛЬЗЯ добавлять по ID в СКУД, по ID в СКУД могут проходить только изменения/удаления!
Допустимо установить по ID в СКУД ID в ВС (при попытке установки уже существующего выдается ошибка БД)!,
Таким образом, если есть поле “Id”, все операции API осуществляет по нему (без этого соглашения невозможно установить ExtId по Id).
Для получения Id добавленного пользователя следует запросить список пользователей или данные пользователя по ExtId.

“Type”: тип пользователя (1-пользователь, 2-посетитель),
“LastName”: фамилия,
“FirstName”: имя,
“SecondName”: отчество,
“DepId”: id подразделения,
“JobId”: id должности,
“TabNum”: табельный номер (это строка!),
“PhotoHash”: 8-байтный хеш фото на основе алгоритма CityHash. Если у пользователя фото нет, данная пара не передается. В качестве хеша используется CityHash, реализация приложена. Хеш-функция возвращает unsigned int64,
далее идут дополнительные поля в формате «ИМЯ СВОЙСТВА:ЗНАЧЕНИЕ», причем значения, независимо от типа дополнительного поля, всегда передаются строкой. При этом передаем и предопределенные, а не только назначенные пользователем свойства (выделены синим) Фото (OW_PHOTO) не передается:
“OW_WEIGHT”:”123”, // это вес пользователя
“workgroup”: “0”, // рабочая группа
OW_PROP_FROMLIST_1: “123”,
OW_PROP_FROMLIST_2: “25.11.2016 18:51:33”,
и т.д. до конца списка дополнительных полей
“Action”: действие с пользователем
}
}

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

Ответ от **СКУД**:

{
“Command”: “userdata”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}

Примечание: если у пользователя нет фотографии (или если она была удалена), то поле “PhotoHash” передавать не надо. Таким образом, на стороне СКУД необходимо явно проверять, есть ли в JSON-объекте поле “PhotoHash”. Если нет, фото пользователя надо удалить.

Примечание 1: если из ВС пришел запрос на удаление пользователя, а у него есть активные карты, СКУД вернет код ошибки, т.к. нельзя удалять пользователя с картами.

Примечание 2: Если из ВС пришел запрос на добавление пользователя, а такой ExtId уже есть в СКУД (добавление - только по ExtId!), то пользователь не будет сохранен, и СКУД вернет код ошибки. Кроме того, теоретически может возникнуть ситуация, при которой СКУД не будет знать id подразделения или id должности. В этом случае СКУД также вернет код ошибки и не будет сохранять пользователя. Если необходимо передать пользователя без указания подразделения и должности, следует передавать соответствующее Id:0.

Примечание 3: Передавать следует все поля, исключение составляет редактирование и удаление по ExtId - в этом случае Id не передается (в противном случае будет считаться, что редактирование и удаление передаются по Id). При передаче пустых строк - в поля будут записаны пустые строки. При передачи 0 в числовое поле - если поле является указателем на справочник или идентификатором, в БД будет занесено значение NULL (ExtId, DepId, JobId). Поле типа пользователя не проверяется на равенство (1,2), для того, чтобы не менять API при расширении типизации пользователей (например, автомобили - тип 3), поэтому за правильностью значений в данном поле ВС несет полную ответственность.

Требование передачи всех полей распространяется и на дополнительные поля (свойства пользователя). Редактирование дополнительных полей осуществляется по схеме «удалить все дополнительные поля, затем добавить присланные в команде». Поэтому, если в команде не будут присланы дополнительные поля, все дополнительные поля будут удалены. Такая схема применяется для того, чтобы не вводить дополнительной команды «удалить свойство пользователя». Данная логика не распространяется на фото, про фото см. «Примечание» без номера.

Примечание 4: Редактировать и удалять даем только неархивных пользователей!

6.38 +Запрос фотографии у ВС по инициативе СКУД

СКУД, получив данные об изменении пользователя, сравнивает значение хеша фотографии, хранящегося у него, со значением, полученным от **ВС**. Если значения не совпадают, **СКУД** отправляет запрос на получение обновленной фотографии пользователя.

ВС, получив запрос, отправляет Base64-закодированный массив, содержащий фото пользователя.

Формат запроса от **СКУД**:

{
“Command”: “getphoto”,
“Id”: …,
“Version”: версия команды,
“UserId”: идентификатор пользователя в **ВС**, для которого запрашиваем фото
}

Ответ от **ВС**:

{
“Command”: “getphoto”,
“Id”: тот же, что передан сервером,
“Version”: версия команды,
“UserId”: идентификатор пользователя в ВС, для которого запрашивали фото,
“Photo”: Base64-закодированный массив, содержащий фото пользователя
}

Примечание: здесь запрос идет именно по идентификатору пользователя в **ВС**, поскольку генерируется СКУД в ответ на добавление/изменение пользователя со стороны внешней системы, следовательно, в общем случае только этот идентификатор и известен (при добавлении пользователя на момент генерации запроса идентификатор пользователя в СКУД еще неизвестен ВС).

6.39 +Запрос фотографии у СКУД по инициативе ВС

Применяется, например, при фото/видео верификации, если в качестве источника данных используется БД СКУД. Если, запросив у СКУД данные о пользователе, Вы видите, что фото изменилось (Hash не совпал), или Вы не храните фото но видите, что оно есть (в данных присутствует поле хэш-суммы фотографии), то вы можете запросить фото дополнительным запросом.

СКУД, получив запрос, отправляет Base64-закодированный массив, содержащий фото пользователя.

Формат запроса от **ВС**:

{
“Command”: “get_photo”,
“Id”: …,
“Version”: версия команды,
“UserId”: идентификатор пользователя в **СКУД**, для которого запрашиваем фото
}

Ответ от **СКУД**:

{
“Command”: “get_photo”,
“Id”: тот же, что передан сервером,
“Version”: версия команды,
“UserId”: идентификатор пользователя в **СКУД**, для которого запрашивали фото,
“Photo”: Base64-закодированный массив, содержащий фото пользователя
}

Не предусматривается ответа ошибкой, если пользователь не найден, вернется пустой массив (пустая строка).

Примечание: здесь запрос идет именно по идентификатору пользователя в **СКУД**, на тот случай, если ВС только читает пользователей из СКУД (для разбора событий и фото/видео верификации).

6.40 +Оповещение об изменении пользователя

Информирование **ВС** об изменении пользователя, выполненном в **СКУД**. Выполняется по инициативе **СКУД**. Формат сообщения от **СКУД**:

{
“Command”: “useredit”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“UserId”: идентификатор пользователя в **СКУД**,
“Action”: действие (0, 1, 2)
}
}

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

6.41 +Добавление карты (с использованием шаблона прав, полученного из СКУД)

Отправляется от **ВС** в **СКУД** при добавлении карты пользователя.

Передаются следующие данные:

1. Id пользователя в ВС, для которого добавляем карту
2. Номер карты
3. Id шаблона прав
4. Начало срока действия в формате «дд.мм.гггг чч:мм:сс»
5. Конец срока действия в формате «дд.мм.гггг чч:мм:сс»
6. Признак активности

Подробнее про признак активности:

- 0 – карта будет запрещена, даже если по сроку действия ей можно ходить.
- 1 – карта будет загружена тогда, когда наступит начало ее срока действия и будет запрещена по окончании этого срока.

Формат команды от **ВС**:

{
“Command”: “addcard”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“Id”: идентификатор пользователя в ВС,
“CardNum”: номер карты,
“TemplateId”: id шаблона прав,
“StartDate”: начало срока действия в формате «дд.мм.гггг чч:мм:сс»
“EndDate”: конец срока действия в формате «дд.мм.гггг чч:мм:сс»
“Action”: признак активности
}
}

Ответ от **СКУД**:

{
“Command”: “addcard”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}

6.42 +Редактирование карты (с использованием шаблона прав, полученного из СКУД)

Отправляется от **ВС** в **СКУД** при редактировании карты пользователя.

Передаются следующие данные:

1. Номер карты (НЕ МЕНЯЕТСЯ, используется для поиска карты)
2. Id шаблона прав
3. Начало срока действия в формате «дд.мм.гггг чч:мм:сс»
4. Конец срока действия в формате «дд.мм.гггг чч:мм:сс»
5. Признак активности

Подробнее про признак активности:

- 0 – карта будет запрещена, даже если по сроку действия ей можно ходить.
- 1 – карта будет загружена тогда, когда наступит начало ее срока действия и будет запрещена по окончании этого срока.

Формат команды от **ВС**:

{
“Command”: “editcard”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“Id”: идентификатор пользователя,
“CardNum”: номер карты,
“TemplateId”: id шаблона прав,
“StartDate”: начало срока действия в формате «дд.мм.гггг чч:мм:сс»
“EndDate”: конец срока действия в формате «дд.мм.гггг чч:мм:сс»
“Action”: признак активности
}
}

Ответ от **СКУД**:

{
“Command”: “editcard”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}

6.43 +Удаление карты

Отправляется от ВС в СКУД при удалении карты пользователя.

Формат команды от **ВС**:

{
“Command”: “delcard”,
“Id”: …,
“Version”: версия команды,
“CardNum”: номер карты,
}

Ответ от **СКУД**:

{
“Command”: “delcard”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}

6.44 +Загрузка/запрещение карты

Отправляется от **ВС** в **СКУД** при загрузке или запрещении карты пользователя.

Передаются следующие данные:

1. Номер карты (НЕ МЕНЯЕТСЯ, используется для поиска карты)
2. Признак активности

Подробнее про признак активности:

- 0 – карта будет запрещена, даже если по сроку действия ей можно ходить.
- 1 – карта будет загружена тогда, когда наступит начало ее срока действия и будет запрещена по окончании этого срока.

Формат команды от **ВС**:

{
“Command”: “loadcard”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“CardNum”: номер карты,
“Action”: признак активности
}
}

Ответ от **СКУД**:

{
“Command”: “loadcard”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)

}

Примечание: возврат кода ошибки 0 в данном случае будет означать, что действие с картой добавлено в очередь на выполнение. Реальный статус карты будет получен в другой команде.

6.45 +Передача статуса карты

Команда отправляется от **СКУД** к **ВС** при изменении статусов одной или нескольких карт.

Формат команды от **СКУД**:

```
{
“Command”: “cardstate”,
“Id”: …,
“Version”: версия команды,
“Data”: [
                {
                        “CardNum”: номер карты 1,
                        “State”: статус карты 1
                },
                …,
                {
                        “CardNum”: номер карты N,
                        “State”: статус карты N
                }
        ]
}
```

Возможные статусы карт:

- Не загружена = -1.
- Загружена = 0.
- Ошибка при загрузке = 1.
- Запрещена по времени = 2.
- Ошибка запрещения по времени = 3.
- Выполняется операция = 4.
- Запрещена оператором = 5.
- Ошибка запрещения оператором = 6.
- Запрещена по неактивности = 7.
- Ошибка запрещения по неактивности = 8.
- Запрещена, т.к. пользователь в отпуске = 9.
- Ошибка запрета для пользователя в отпуске = 10.

Статусы (7-10) зарезервированы, но сейчас в СКУД не используются.

6.46 +Запрос статуса карт

Отправляется от **ВС** к **СКУД**. Необходима для синхронизации статусов карт.

Передается список номеров карт или номер карты 0 (для единообразия также в виде массива из одного элемента) для синхронизации статусов всех карт.

Формат команды от **ВС**:

{
“Command”: “cardstatelist”,
“Id”: …,
“Version”: версия команды,
“CardNum”: [0] или [номер карты 1, …, номер карты N]
}

Ответ от **СКУД**:

{
“Command”: “cardstatelist”,
“Id”: тот же, что передан сервером,
“Version”: версия команды,
“Data”: [
{
“CardNum”: номер карты 1,
“State”: статус карты 1
},
…,
{
“CardNum”: номер карты N,
“State”: статус карты N
}
]
}

Возможные статусы карт смотри выше.

Примечание: если часть номеров карт, переданных ВС, неизвестна СКУД, то для этих карт ответ не придет, т.е. их просто не будет в ответной команде.

! ВАЖНОЕ ЗАМЕЧАНИЕ: В качестве альтернативного варианта задания прав карте можно использовать список прав карты на точках доступа. При этом в **ВС** формируются собственные способы хранения прав, никак не привязанные к шаблонам **СКУД**. В **СКУД** от **ВС** передается набор прав на ТД (точках доступа). Список ТД запрашиваются у **СКУД**, поскольку является конфигурационной информацией.

Вероятно, данный способ задания прав предпочтительнее для **ВС**, поскольку позволяет абстрагироваться от структур данных **СКУД**. Нижеследующие команды посвящены указанному способу задания/редактирования прав.

6.47 +Добавление карты (с указанием списка прав по ТД)

Отправляется от **ВС** в **СКУД** при добавлении карты пользователя.

Передаются следующие данные:

1. Id пользователя в ВС, для которого добавляем карту
2. Номер карты
3. Начало срока действия в формате «дд.мм.гггг чч:мм:сс»
4. Конец срока действия в формате «дд.мм.гггг чч:мм:сс»
5. Признак активности
6. Список прав на ТД

Подробнее про признак активности:

- 0 – карта будет запрещена, даже если по сроку действия ей можно ходить.
- 1 – карта будет загружена тогда, когда наступит начало ее срока действия и будет запрещена по окончании этого срока.

Формат команды от **ВС**:

```
{
“Command”: “add_card”,
“Id”: …,
“Version”: версия команды,
“Data”: {
                “Id”: идентификатор пользователя в ВС,
                “CardNum”: номер карты,
                “StartDate”: начало срока действия в формате «дд.мм.гггг чч:мм:сс»
                “EndDate”: конец срока действия в формате «дд.мм.гггг чч:мм:сс»
                “Action”: признак активности,
                “Rights”: [
                                {
                                        “ApId”: Id точки доступа (ТД),
                                        “RightType”: код типа прав по данной ТД (см. ниже),
                                        “TimetableNumber”: номер расписания на данной ТД (1-6, 0 - без контроля по времени)
                                },
                                {
                                ...
                                }
                        ]
        }
}
```

Где “Right” - список прав для каждой ТД, на которой доступ разрешен. Соответственно, на других ТД доступ будет запрещен.

Допустимы следующие типы прав:

- Доступ запрещен - код 15 (правильнее будет просто не давать никаких прав карте на данной ТД, так что этот код лучше не использовать);
- Обычный - код 0;
- Ответственный - код 1 (обычный + права на смену режима);
- Разовый - код 4 (обычный + карта будет изъята в картоприемник);

Ответ от **СКУД**:

```
{
“Command”: “add_card”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}
```

6.48 +Редактирование карты (с указанием списка прав по ТД)

Отправляется от **ВС** в **СКУД** при редактировании карты пользователя.

Передаются следующие данные:

1. Номер карты (НЕ МЕНЯЕТСЯ, используется для поиска карты)
2. Начало срока действия в формате «дд.мм.гггг чч:мм:сс»
3. Конец срока действия в формате «дд.мм.гггг чч:мм:сс»
4. Признак активности
5. Список прав на ТД

Подробнее про признак активности:

- 0 – карта будет запрещена, даже если по сроку действия ей можно ходить.
- 1 – карта будет загружена тогда, когда наступит начало ее срока действия и будет запрещена по окончании этого срока.

Формат команды от **ВС**:

```
{
“Command”: “edit_card”,
“Id”: …,
“Version”: версия команды,
“Data”: {
                “Id”: идентификатор пользователя,
                “CardNum”: номер карты,
                “StartDate”: начало срока действия в формате «дд.мм.гггг чч:мм:сс»,
                “EndDate”: конец срока действия в формате «дд.мм.гггг чч:мм:сс»,
                “Action”: признак активности,
                “Rights”: [
                                {
                                        “ApId”: Id точки доступа (ТД),
                                        “RightType”: код типа прав по данной ТД (см. команду выше),
                                        “TimetableNumber”: номер расписания на данной ТД (1-6, 0 - без контроля по времени)
                                },
                                {
                                ...
                                }
                        ]
        }
}
```

Ответ от **СКУД**:

```
{
“Command”: “edit_card”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}
```

6.49 +Редактирование пин-кода пользователя

Отправляется от **ВС** в **СКУД** при редактировании пин-кода пользователя.

Передаются следующие данные:

1. Id пользователя в **ВС**
2. Пин-код
3. Действие

Формат команды от **ВС**:

```
{
“Command”: “pincode”,
“Id”: …,
“Version”: версия команды,
“Data”: {
                “Id”: идентификатор пользователя в ВС,
                “Pincode”: строка, содержит до 8 литер-цифр(0-9),
                “Action”: действие
        }
}
```

Возможные значения поля “Action”:

- Данные добавлены: 0.
- Данные изменены: 1.
- Данные удалены: 2.

Примечание: У пользователя предусмотрен единственный пин-код. Таким образом, добавление и редактирование эквивалентны. Вновь добавленный /измененный пин-код получит статус «ошибка загрузки», что приведет к его загрузке в аппаратуру системным роботом. Если у пользователя уже есть пин-код, то изменить пин-код можно только тогда, когда он не загружен или запрещен (см. далее команды загрузки пин-кода).

Примечание 2:При удалении собственно пин-код можно не указывать (поскольку он может быть только 1), проверка на соответствие удаляемого пин-кода передаваемому осуществлена не будет. Удалять можно только не загруженный/запрещенный пин-код.

Ответ от **СКУД**:

```
{
“Command”: “pincode”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}
```

6.50 +Загрузка/запрещение пин-кода

Отправляется от **ВС** в **СКУД** при загрузке или запрещении пин-кода пользователя.

Передаются следующие данные:

Признак активности:

- 0 – пин-код будет запрещен.
- 1 – пин-код будет загружен.

Формат команды от **ВС**:

```
{
“Command”: “loadpincode”,
“Id”: …,
“Version”: версия команды,
“Data”: {
                “Id”: идентификатор пользователя в ВС,
                “Action”: признак активности
        }
}
```

Ответ от **СКУД**:

```
{
“Command”: “loadpincode”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}
```

Примечание: возврат кода ошибки 0 в данном случае будет означать, что действие с пин-кодом добавлено в очередь на выполнение. Реальный статус пин-кода будет получен в другой команде.

6.51 +Запрос статуса пин-кода пользователя

Отправляется от **ВС** к **СКУД**. Необходима для отслеживания процесса загрузки пин-кода.

Формат команды от **ВС**:

```
{
“Command”: “pincodestate”,
“Id”: …,
“Version”: версия команды,
“Data”: {
                “Id”: идентификатор пользователя в ВС
        }
}
```

Ответ от **СКУД**:

```
{
“Command”: “pincodestate”,
“Id”: тот же, что передан сервером,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее),
“State”: статус пин-кода
}
```

Возможные статусы карт смотри выше (такие же, как у карты). Если есть ошибка (ErrCode <> 0), статус пин-кода в ответе не содержится.

6.52 Добавление биометрического признака карте пользователя

<пока не реализован>

Отправляется от **ВС** в **СКУД** при добавлении биометрических данных пользователя.

Передаются следующие данные:

- Id пользователя;
- Номер карты пользователя;
- Тип биометрических данных (0 - шаблон отпечатка пальца “Biosmart”, 1 - шаблон отпечатка пальца “ZKTeco”);
- Собственно шаблон отпечатка - Base64-закодированный массив, содержащий шаблон биометрических данных.

Замечания:
- Для одной карты пользователя предполагается наличие не более 5 биометрических признаков одного типа (ограничения интерфейса программы «Пропуска» СКУД «РЕВЕРС 8000»). Попытка добавить более 5 признаков вернет ошибку.
- Действие передавать не нужно - только добавление. Поскольку для биометрического признака не предусмотрен Id из ВС, для упрощения взаимодействия считаем, что при необходимости заменить биометрический признак, удаляется весь список биометрических признаков данного типа для данной карты, а затем добавляются нужные признаки.

Формат команды от **ВС**:

{
“Command”: “addbiometry”,
“Id”: …,
“Version”: версия команды,
“Data”: {
“Id”: идентификатор пользователя,
“CardNum”: номер карты,
”Type”: тип биометрических данных (0,1),
“BioData”: Base64-закодированный массив, содержащий шаблон биометрических данных
}
}

Ответ от **СКУД**:

{
“Command”: “addbiometry”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)

6.53 Очистка биометрических признаков данного типа карте пользователя

<пока не реализован>

Отправляется от **ВС** в **СКУД** при добавлении биометрических данных пользователя.

Передаются следующие данные:

- Id пользователя;
- Номер карты пользователя (-1 - удаляем для всех карт данного пользователя);
- Тип биометрических данных (0 - шаблон отпечатка пальца “Biosmart”, 1 - шаблон отпечатка пальца “ZKTeco”) (-1 - удаляем для всех типов биометрии);

Формат команды от **ВС**:

```
{
“Command”: “clearbiometry”,
“Id”: …,
“Version”: версия команды,
“Data”: {
                “Id”: идентификатор пользователя,
                “CardNum”: номер карты (или -1, см. выше),
                ”Type”: тип биометрических данных (0,1) (или -1, см. выше)
        }
}
```

Ответ от **СКУД**:

```
{
“Command”: “clearbiometry”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}
```

6.54 +Загрузка контроллера

Команда от ВС к СКУД. Команду следует выполнять после выполнения ряда действий (редактирование расписания, редактирование календаря праздников).

Формат команды от **ВС**:

```
{
“Command”: “loadcontroller”,
“Id”: …,
“Version”: версия команды,
“ApId”: Id точки доступа, контроллер для которой надо загрузить (0 - все контроллеры, требуется при смене календаря праздничных дней)
}
```

Ответ **СКУД**:

{
“Command”: “loadcontroller”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}
Ответ «Ok» означает, что есть такая ТД и такой контроллер. Далее я просто поставлю запрос в очередь. Можно при получении данной команды сбросить загруженность у контроллера в «ошибка загрузки», контроллер поставить в очередь загрузки (если уже есть в очереди, не ставить). Потом его неспеша грузить в потоке загрузки. По загрузке - установить статус - «загружен». Очевидно, потребуется запрос загруженности контроллера, который ВС может издавать, а может и нет - на свое усмотрение. Параметром в этом запросе можно вернуть, стоит ли контроллер на загрузке (т.е. есть ли он в списке загрузки - очевидно, что при рестарте сервера API список контроллеров к загрузке не будет сохранен, тогда можно отправить контроллер на загрузку заново).

6.55 +Запрос загруженности контроллера.

Смысл - см. команду выше.

Команда от ВС к СКУД.

Формат команды от **ВС**:

{
“Command”: “controllerstate”,
“Id”: …,
“Version”: версия команды,
“ApId”: Id точки доступа, состояние загруженности контроллера для которой надо получить
}

Ответ **СКУД**:

{
“Command”: “controllerstate”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее),
“State”: 0 или 1 - контроллер загружен (нет, да),
“InLoad”: 0 или 1 (нет, да)
}

Данные актуальны, только если нет ошибки (ErrCode = 0).

6.56 +Передача онлайн событий

Команда от **СКУД** в **ВС**. Передаются события доступа и запрета доступа (полный перечень с кодами - см. ниже). Команда необходима для того, чтобы ВС могла анализировать у себя необходимые данные.

{
“Command”: “events”,
“Id”: …,
“Version”: версия команды,
“Data”: [
{

“EvTime”: время события (строка в формате “dd.mm.yyyy hh:nn:ss”),
“EvCode ”: код события,
“EvAddr”: ID источника события (ID записи из таблицы устройств, Точек Доступа, Шлейфов, Разделов и т.п.- в зависимости от типа события - см. таблицу в приложении),
“EvUser”: ID пользователя (или 0, если событие без пользователя),
“EvCard”: номер карты или 0, если нет карты (либо карта не передается),
“EvSection”: код раздела или пустая строка
“UserInfo”:
{
}*
},
{
…
}
]
}

Событие отправляется единожды, поскольку используется протокол TCP, а он является протоколом с гарантированной доставкой, и, если с сетью все в порядке, оно будет доставлено. Можно также предусмотреть настройку - передавать только события с пользователями - тогда информационный поток от данного экземпляра сервера API может быть использован только для online фото/видео-верификации (см. ниже «Настройка передачи событий»).
*Можно также предусмотреть настройку - передавать расширенную информацию о пользователе. В этом случае сервер сразу сообщает полную информацию о пользователе, включая все дополнительные поля, причем все дополнительные поля передаются как строки по имени свойства (OPLIST_PROPERTY: OW_PROP_FROMLIST_1, OW_PROP_FROMLIST_2 и т.п.). Перечень дополнительных полей можно получить специальной командой (см. “userproplist”). Описание информации о пользователе - см. команду “userinfo”.
! ПРЕДЛАГАЕТСЯ это изъять - многоформатность потока событий затруднит их разбор. Данные о пользователе всегда можно дозапросить командой ЗАПРОС ИНФОРМАЦИИ О ПОЛЬЗОВАТЕЛЕ (userinfo). На скорости работы API это практически не скажется, даже наоборот общее быстродействие можно повысить за счет сокращения передачи данных - клиент может кэшировать данные со своей стороны, и запрашивать данные только о тех, кого он не знает.

Примечание: Для управляемой клиентом передачи журнала событий от сервера используется другая команда - getevents.

6.57 +Передача событий по запросу

Команда служит для управляемого получения событий из БД СКУД. Одновременно передается не более 20 событий. В качестве параметра указывается ID - следует сдавать события с ID, БОЛЬШЕМ указанного. Таким образом, если передавать ID последнего принятого события, получаем возможность последовательно получить на клиента содержимое журнала регистрации. Соответственно, для того, чтобы начать вычитывать события сначала, в качестве ID передать 0.

Действует настройка - передавать расширенное описание пользователя (см. выше). Также действует настройка - передавать все события, или только события с пользователями.

Формат команды от **ВС**:

{
“Command”: “getevents”,
“Id”: …,
“Version”: версия команды,
“EventId”: Id события, начиная с которого надо вычитывать события
}

Ответ **СКУД**:

{
“Command”: “getevents”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее). Если код ошибки = 0, или ответ вообще не содержит записи с кодом ошибки, то в ответе содержится блок событий:
“Data”: [
{
“EvId”: Id события,
“EvTime”: время события (строка в формате “dd.mm.yyyy hh:nn:ss”),
“EvCode ”: код события,
“EvAddr”: ID источника события (ID записи из таблицы устройств, Точек Доступа, Шлейфов, Разделов и т.п.- в зависимости от типа события - см. таблицу в приложении),
“EvUser”: ID пользователя в СКУД (или 0, если событие без пользователя),
“EvExtUser”: ID пользователя в ВС (или 0, если событие без пользователя),
“EvCard”: номер карты или 0, если нет карты (либо карта не передается),
“EvSection”: код раздела или пустая строка
“UserFullInfo”:
{
}*
},
{
…
}
]
}

Ответ содержит 20 или менее событий. Если передано менее 20 событий, значит, переданному условию (Id > переданного) удовлетворяют менее 20 записей.

! ПРЕДЛАГАЕТСЯ изъять автоматическую передачу расширенной информации о пользователе - многоформатность потока событий затруднит их разбор. Данные о пользователе всегда можно дозапросить командой ЗАПРОС ИНФОРМАЦИИ О ПОЛЬЗОВАТЕЛЕ (userinfo). На скорости работы API это практически не скажется, даже наоборот общее быстродействие можно повысить за счет сокращения передачи данных - клиент может кэшировать данные со своей стороны, и запрашивать данные только о тех, кого он не знает.

6.57.1 Передаваемые коды событий при активной настройке «Передавать только события с пользователями».

В данном разделе приводится список событий и их кодов, которые передаются от **СКУД** в **ВС**.

1. Проход (код = 1).
2. Запрет доступа – нет прав (код = 2).
3. Запрет доступа – ТД блокирована (код = 3).
4. Предъявлена карта (код = 4).
5. Отказ от прохода (код = 5).
6. Отказ в доступе – нет связи с ТД (код = 6).
7. Запрет доступа – карта запрещена (код = 7).
8. Запрет доступа – несистемная карта (код = 307).
9. Запрет доступа – зона (код = 48).
10. Проход с нарушением зональности (код = 49).
11. Запрет доступа – время (код = 50).
12. Проход с нарушением по времени (код = 51).
13. Запрет доступа – проход по одному (код = 52).
14. Запрет доступа – доступ запрещен оператором (код = 53).
15. Проход по кнопке ДУ (код = 311).
16. Отказ от прохода по кнопке ДУ (код = 312).
17. Ошибка комиссионирования (код = 400).
18. Одноразовый пропуск сдан (код = 403).
19. Запрет доступа – несистемный пин-код (код = 420).

6.58 +Настройка передачи событий

Команда от ВС к СКУД - служит для управления фильтром событий

Формат команды от **ВС**:

{
“Command”: “filterevents”,
“Id”: …,
“Version”: версия команды,
“Filter”: 0 или 1; 0 - передавать все события, 1 - установить фильтр, т.е. передавать только события с пользователями.
}

Ответ **СКУД**:

{
“Command”: “filterevents”,
“Id”: …,
“Version”: версия команды,
“ErrCode”: код ошибки (см. далее)
}

Настройка сохраняется сервером API. По-умолчанию фильтр включен.

7 Коды ошибок

- Нет ошибок (действие выполнено успешно) – код = 0.
- Не найден id родительского подразделения – код = 1.
- Неизвестный id подразделения – код = 2.
- Неизвестный id должности – код = 3.
- Пользователь с таким id уже существует – код = 4.
- Нельзя удалить пользователя, т.к. у него есть карты – код = 5.
- Нельзя удалить карту, т.к. она не запрещена – код = 6.
- Неизвестный id пользователя – код = 7.
- Неизвестный id шаблона прав – код = 8.
- Карта с таким номером уже используется – код = 9.
- Неизвестный номер карты – код = 10.
- Ошибка БД – код = 11.
- Ошибка связи с компонентами СКУД – код 12.
- Ошибка формата передачи команды от ВС - код 13.
- Неизвестный id точки доступа - код 14.
- Неверный пин-код - код 15.
- Такой id типа праздников уже существует - код 16.
- Неизвестный id типа праздников – код = 17.
- Указана невалидная дата - код = 18.
- Неизвестый Id пользователя в ВС (OW_EXT_ID) = 19.
- Неизвестный номер расписания = 20.

8 Начальная синхронизация данных

Для начальной синхронизации данных из СКУД необходимо выгрузить, как минимум, данные по пользователям, подразделениям и должностям.

Кроме того, необходимо в СКУД все собственные id пользователей, подразделений и должностей продублировать в поля *_EXT_ID, т.к. ВС будет использовать эти же идентификаторы.