



# Электроника

системы управления  
безопасностью

150001, Россия, Ярославль,  
ул. Большая Федоровская, 75  
Тел./факс: +7 (4852) 66-00-15  
эл. почта: [info@electronika.ru](mailto:info@electronika.ru)  
[www.electronika.ru](http://www.electronika.ru)

## ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ELECTRONIKA SECURITY MANAGER

Документация разработчика

Техническое задание на разработку драйвера интеграции ПО Sigur

11.2019

## Оглавление

<b>1</b>	<b>Введение .....</b>	<b>5</b>
1.1	Назначение .....	5
1.2	Предмет .....	5
<b>2</b>	<b>Характеристика объекта автоматизации.....</b>	<b>6</b>
2.1	Характеристика СКУД Sigur .....	6
2.2	Программный комплекс ESM .....	7
2.2.1	Основные задачи ESM.....	7
2.2.2	Подсистема драйверов интеграции .....	8
<b>3</b>	<b>Функциональные требования .....</b>	<b>9</b>
3.1	Драйвер интеграции Sigur должен обеспечивать: .....	9
3.2	Требования к модели данных драйвера .....	10
3.3	Требования к использованию существующих данных ESM.....	11
3.4	Требования к реализации логики обработки данных.....	12
<b>4</b>	<b>Общие требования .....</b>	<b>13</b>
4.1	Требования к масштабируемости.....	13
4.2	Требования к надежности .....	13
4.3	Требования к производительности .....	13
4.4	Требования к средствам разработки и производства.....	13
4.5	Критерии удовлетворения требований.....	13
<b>5</b>	<b>История изменений документа .....</b>	<b>14</b>
<b>6</b>	<b>Приложение 1 - Общее описание интеграции внешних систем .....</b>	<b>15</b>
6.1	Введение .....	15
6.1.1	Программный комплекс ESM .....	15
6.1.2	Основные задачи ESM.....	15
6.1.3	Подсистема драйверов интеграции .....	15
6.2	Функциональные требования .....	16
6.2.1	Драйвер интеграции должен обеспечивать: .....	16
6.2.2	Требования к модели данных драйвера .....	17
6.2.3	Требования к использованию существующих данных ESM.....	18

---

6.2.4	Требования к реализации логики обработки данных от внешней системы.....	19
6.3	Общие требования .....	20
6.3.1	Требования к масштабируемости.....	20
6.3.2	Требования к надежности .....	20
6.3.3	Требования к производительности .....	20
6.3.4	Требования к средствам разработки и производства.....	20
6.3.5	Критерии удовлетворения требований.....	21
7	<b>Приложение 2 - Описание протокола взаимодействия sServer с Ports .....</b>	<b>27</b>
7.1	Формат сообщений драйвера .....	27
8	<b>Приложение 3 - Описание интерфейса интеграции ПО Sigur .....</b>	<b>30</b>
8.1	Текстовый протокол Sigur .....	30
8.2	Структура БД Sirug .....	31
9	<b>Приложение 4 - Декларативное описание объектов драйвера .....</b>	<b>32</b>
9.1	Общие сведения.....	32
9.2	Создание оборудования .....	32
9.3	Описание форматов файлов.....	32
9.3.1	__init__.py .....	32
9.3.2	<имя оборудования> *.py .....	33
9.4	<b>Приложение 1. Параметры системных действий .....</b>	<b>37</b>
9.4.1	init(self).....	37
9.4.2	finit(self).....	37
9.4.3	onEvent(self, event=None, portsEvent=None) .....	37
9.4.4	preCreate(cls, createAttrs=None, targetObj=None, mainObj=None, info=None) .....	37
9.4.5	postCreate(self, ignore_create_objects=False).....	38
9.4.6	preDelete(self, deleteAsLink=False) .....	38
9.4.7	postDelete(cls, **kwargs) .....	38
9.4.8	preChangeParent(self, newParent=None, addr=None, info=None) .....	38
9.4.9	postChangeParent(self, oldParent=None, info=None).....	38
9.4.10	preAction (Атрибут) .....	38
9.4.11	postAction (Атрибут) .....	38
9.4.12	preAction (Ссылка) .....	38
9.4.13	postAction (Ссылка).....	38

9.5	Приложение 2. Типы редакторов значений атрибутов .....	39
9.5.1	int .....	39
9.5.2	date .....	39
9.5.3	UnixDate .....	39
9.5.4	UnixDateTime .....	39
9.5.5	time .....	39
9.5.6	datetime .....	39
9.5.7	enum .....	39
9.5.8	treeSelect .....	39
9.5.9	dynamicTreeSelect .....	40
9.5.10	checkbox .....	40
9.5.11	fileMapdx .....	40
9.5.12	fileiconico .....	40
9.5.13	multiSelect .....	40
10	Приложение 5 - Описание сущностей Общих объектов .....	41
11	Приложение 6 - Описание объектной модели ядра .....	42
11.1	dev_except module .....	42
11.2	dev_obj_model module .....	45
11.3	dev_obj_model_element module .....	48
11.4	equipment_interface module .....	64
11.5	equipment_module module .....	67
11.6	primitive module .....	70
12	Приложение 7 - Список событий драйвера ПО Sigur .....	71

## 1 ВВЕДЕНИЕ

ПО ESM в рамках обеспечения безопасности объектов, выступает интеграционной платформой для объединения подсистем безопасности в единый комплекс. В качестве одной из подсистем безопасности может выступать система контроля и управления доступом (СКУД) функционирующая на основе ПО [Sigur](#). Цель разработки состоит в доработке программного комплекса по добавлению драйвера (модуля) интеграции ПО Sigur (далее Sigur).

### 1.1 Назначение

Данное техническое задание содержит требования к разработке модуля взаимодействия ESM с ПО Sigur. В результате разработки модуль интеграции (далее Драйвер) должен обеспечить необходимый функционал для выполнения мониторинга и управления СКУД под управлением Sigur из программного обеспечения ESM

.

### 1.2 Предмет

Предметом данного документа являются требования к разработке Драйвера интеграции ПО Sigur. При реализации интеграции следует основываться на том, что Драйвер взаимодействует с ПО Sigur по единому каналу и протоколу взаимодействия, а аппаратный комплекс СКУД представляет собой упрощенную структуру точек доступа и объектов доступа. При разработке Драйвера интеграции не рассматривается мониторинг и управлением третьих систем, которые могут быть подключены к Sigur, например, системы видеонаблюдения и охранной сигнализации.

.

## 2 ХАРАКТЕРИСТИКА ОБЪЕКТА АВТОМАТИЗАЦИИ

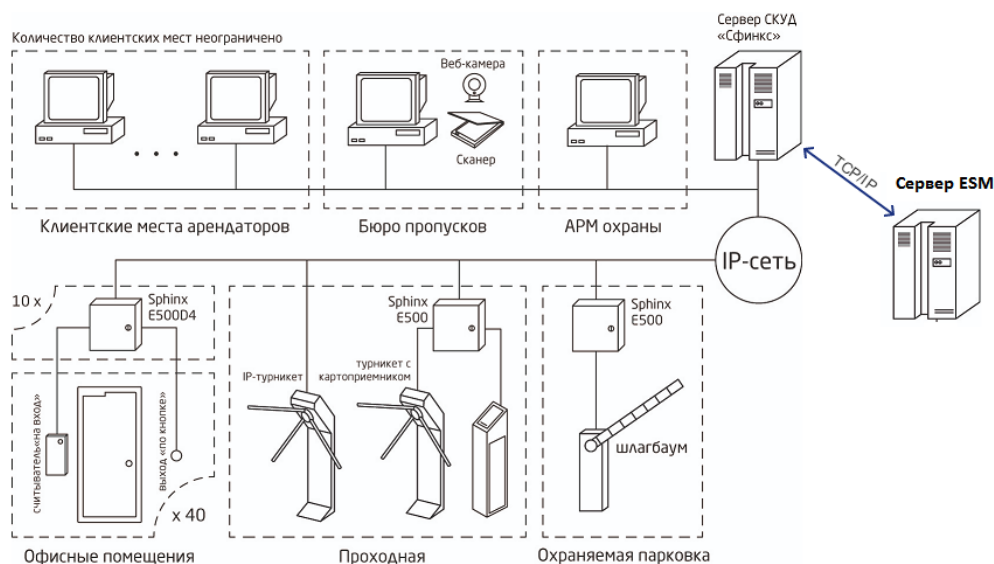
### 2.1 Характеристика СКУД Sigur

Продукция SIGUR применяется для обеспечения безопасности объектов любого масштаба и назначения.

СКУД Sigur обеспечивает следующие варианты построения систем контроля доступа:

- Бизнес-центры и административные здания
- Промышленные предприятия
- Гостиницы, отели, hostels
- Территориально-распределенные объекты
- Крупные учебные заведения
- и другие

Типовая структура СКУД выглядит следующим образом:



Система Sigur состоит из следующих компонентов:

- Сервер системы – компьютер под управлением операционной системы Windows или Linux Debian с установленным программным обеспечением СКУД «Sigur». К серверу максимально подключаются до 16 преобразователей интерфейсов USB – RS485 «Sigur Connect» и неограниченное количество контроллеров серии E (с Ethernet интерфейсом).
- Клиентское место системы – рабочее место пользователя системы, которое можно запустить на любом компьютере, связанном с главным сервером системы по протоколу TCP/IP или непосредственно на сервере.
- Преобразователь интерфейсов USB – RS485 «Sigur Connect» – электронный модуль в пластиковом корпусе. Преобразователь обеспечивает согласование порта USB компьютера с

линией связи RS485.

- Линия связи – соединяет компьютер с контроллерами системы. Интерфейс связи – RS485. К каждой линии можно подключить до 255 контроллеров. Возможно использование повторителей, увеличивающих максимальную длину линии в два или четыре раза.
- Контроллер – электронное устройство, представляющее собой микропроцессорную плату высокой степени интеграции в металлическом корпусе. Контроллер подключается к линии связи RS485 или Ethernet, считывателям, датчикам и к исполнительным устройствам.
- Исполнительные устройства – турникеты, ворота, шлагбаумы или двери, оборудованные электромагнитными или электромеханическими замками. Контроллер управляет исполнительными устройствами и получает информацию об их состоянии с помощью своих выходов и входов.
- Считыватели – электронные устройства, предназначенные для ввода запоминаемого кода с клавиатуры либо считывания кодовой информации с ключей (идентификаторов) системы.
- Ключ – уникальный признак объекта доступа (сотрудника, автомобиля, посетителя). Как правило – код электронной карты.
- Объект доступа – сотрудник, посетитель или автомобиль, действия которых регламентируются правилами разграничения доступа.
- Контрольный считыватель – используется для оперативного поиска сотрудников в базе данных системы и для быстрого ввода кода нового пропуска в систему.

❗ На данной схеме также обозначен программный комплекс ESM для определения точки взаимодействия со СКУД ESM.

## 2.2 Программный комплекс ESM

Программное обеспечение ESM используется для построения комплексных систем безопасности от локального до федерального масштаба. ПО позволяет решить весь список задач по обеспечению инженерно-технической безопасности объекта и настройке системы управления режимом с выведением ситуации под контроль руководства.

### 2.2.1 Основные задачи ESM

Задачи, решаемые ПО в качестве основной интеграционной платформы системы безопасности:

1. Конфигурация подключаемого оборудования;
2. Управление базами данных;
3. Сбор, обработки и анализ информации;
4. Обеспечение пользовательских интерфейсов;

5. Защита информации и разделение доступа к ней;
6. Управление подсистемами безопасности;
7. Интеграция с информационными системами управления.

### 2.2.2 Подсистема драйверов интеграции

Подсистема драйверов интеграции предназначена для обеспечения функционала взаимодействия со сторонними системами или приборами.

Описание общих технических решений по разработке драйверов интеграции ESM приведено в [Приложение 1 - Общее описание интеграции внешних систем](#).

Согласно указанному техническому решению данная разработка включает реализацию Драйвера из двух частей:

- логическая часть, описывающая модель данных интеграции и обработку потока данных;
- подключаемая часть, реализующая взаимодействие с Sigur и преобразование нотации протокола в/из структуру JSON принятую в ESM. Описание структур приведено в [Приложение 2 - Описание протокола взаимодействия sServer с Ports](#).



### 3 ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

#### 3.1 Драйвер интеграции Sigur должен обеспечивать:

1. Установление и удержание сеанса связи, генерацию нотификаций об изменении состояния сеанса связи с ПО Sigur. Описание интерфейса и протокола взаимодействия с ПО Sigur приведено в [Приложение 3 - Описание интерфейса интеграции ПО Sigur](#).
2. После успешной установки связи Драйвер должен опросить состояния Точек доступа, и сгенерировать нотификации с полученными состоянием для каждой точки доступа. При потере связи с ПО Sigur, драйвер должен сгенерировать нотификации о потере связи с каждой точкой доступа.
3. После успешной установки связи драйвер должен подписаться в ПО Sigur на получение сообщений о событиях и состояниях по установленному запросу протокола обмена. Для всех полученных сообщений Драйвер должен сгенерировать нотификации в формате JSON, где Точка доступа должна выступать в роли directObject, Объект доступа - в роли subject. Перечень необходимых событий приведен в [Приложение 7 - Список событий драйвера ПО Sigur](#).
4. Надежную обработку последовательности сообщений ПО Sigur и генерацию нотификаций в ESM. Для этого:
  - а. передавать значение временной метки, предусмотренной в объекте Драйвер Sigur, в подключаемую часть с целью загрузки истории события с момента последнего подключения;
  - б. в логической части отбрасывать сообщения, которые старше значения временной метки.
  - с. обновлять значение временной метки при генерации нотификации в ESM.
5. Автоматизированный (по команде) импорт списка объектов доступа (сотрудниках). Процедура импорта должна создавать/редактировать в БД ESM частные лица, пропуска. При импорте необходимо предусмотреть возможность обновления. Для вновь создаваемых Субъектов и Пропусков в ESM, требуется создать именной Уровень доступа, связать его с Уровнем доступа, для нового Уровня доступа необходимо создать объект Право доступа, и назначить данное Право пропуску ESM. Если при импорте производится обновление уже существующего Пропуска, то обновлять (редактировать) существующий уровень доступа не требуется, а также не требуется создавать новый Идентификатор (common\_idcode), и назначать его Пропуску.
6. Автоматизированный (по команде) импорт списка Точек доступа. Предусмотреть возможность обновления существующего списка Точек доступа, при повторном импорте. Для вновь создаваемых точек доступа в процедуре импорта требуется обеспечить автоматическое создание объектов Право доступа

(common\_commonright), и установку ссылки из Права доступа на новую точку доступа.

7. При обработке команд активации/деактивации/блокировки/разблокировке Пропуска в ESM, выполнять соответствующие запросы в ПО Sigur. Для поддержки данного функционала, необходимо реализовать в драйвере
8. Выполнение команд изменения режима работы точек доступа, по установленным запросам протокола обмена с ПО Sigur.

### 3.2 Требования к модели данных драйвера

При разработке Драйвера в логической модели необходимо описать сущности определяющие БД интеграции. Правила описания приведены в [Приложение 4 - Декларативное описание объектов драйвера](#).

Драйвер должен содержать следующие объекты:

**Драйвер Sigur** - объект описывающий подключаемую часть драйвера и его параметры:

- адрес и порт Ports - строка, например 127.0.0.1,8001 - задает адрес соединения с подключаемой частью Драйвера;
- адрес сервера Sigur, порт связи, логин и пароль - параметры подключения к ПО Sigur;
- флаг включения/выключения Драйвера - определяет активность драйвера;
- временная метка последнего полученного сообщения - хранит отметку последнего обработанного сообщения ПО Sigur. Данная метка должна быть передана в подключаемую часть для получения истории события за время отсутствия подключения. Обработка сообщения заключается в разборе данного сообщения, формирования и отправки в очередь сервера ESM соответствующей нотификации;
- константа \_\_obsobjtype равная строке "dev" для поддержки логики работы ESM.

**Точка доступа** - объект описывающий элемент системы Sigur обеспечивающий преграждение и доступ пользователей СКУД. Параметры точки доступа:

- наименование (Description) - описание точки доступа в ПО Sigur;
- внешний идентификатор (external\_id) - идентификатор точки доступа в ПО Sigur;
- константа \_\_obsobjtype равная строке "reader".

**Режим** - объект описывающий набор интервалов доступа Sigur, должен использовать для назначения основного режима работы объекта доступа Sigur. Данный объект должен содержать атрибуты Sigur:

- Название(Description) - строка скопированная из ПО Sigur;
- Внешний идентификатор (external\_id) - идентификатор в ПО Sigur.

**Уровень доступа** - объект задающий список точек доступа назначаемый объекту доступа (пользователю) Sigur и его Основной режим. Данный объект должен быть

связан с объектом "Права" драйвера "Общие объекты". При активации пропуска с таким правом, в Sigur должен быть создан/обновлен объект доступа и ему должны быть назначены точки доступа, входящие в объект "Уровень доступа", а также назначен режим работы установленный в объекту уровень доступа. Атрибуты объекта Уровень доступа:

- Список точек доступа (ap\_list) - задает набор объектов "Точка доступа" Sigur.
- Основной режим (mode) - ссылка на объект Режим.

### 3.3 Требования к использованию существующих данных ESM

Для работы с объектами доступа Sigur драйвер должен использовать существующие сущности ESM драйвера "Общие объекты" (common\_protocol\_config), который содержит объекты являющиеся источником данных для Sigur. Такими источниками являются:

**Организация** (common\_organization) - является источником данных для поля НадОтдел Sigur.

**Подразделение** (common\_department) - является источником данных для поля Отдел Sigur.

**Субъект**(common\_subject) и его ребенок **Частное лицо** (common\_person) - является источником данных для объекта доступа Sigur "Сотрудник".

**Права доступа** (common\_commonright) - является источником данных для объекта доступа Sigur "Сотрудник", а именно Уровень доступа Драйвера Sigur.

В системе ESM объект common\_commonright является контейнером ссылок на частные уровни доступа драйверов интеграции подсистем. При разработке Драйвера интеграции, для определения назначаемого Уровня доступа в СКУД, необходимо использовать данный контейнер.

**Пропуск** (common\_permit) - является источником данных для объекта доступа Sigur "Сотрудник", а именно номер пропуска, PIN, срок действия. Объект "Пропуск" в ESM является базовым объектов для определения экспортируемых данных в интегрируемую систему. Пропуск имеет:

- ссылку на сущность Субъект (common\_subject), определяющий частное лицо - Объект доступа Sigur;
- ссылку на сущность идентификатор (common\_idcode), определяющий код карты - номер пропуска объекта доступа Sigur;
- связь с Правом доступа (common\_commonright), определяющие набор точек доступа назначаемых объекта доступа в Sigur;
- атрибуты начала действия и окончания действия, определяющие срок действия пропуска в Sigur;
- атрибут PIN код, определяющий PIN код объекта доступа в Sigur.

При изменении состояния пропуска (активирован/деактивирован/заблокирован), Драйвер должен обеспечивать выполнение автоматических запросов на изменение состояния объекта доступа ПО Sigur. Описание сущностей приведено в [Приложение 5 - Описание сущностей Общих объектов](#).

### 3.4 Требования к реализации логики обработки данных

Для реализации логики работы драйвера в модели интеграции ESM принято использование специальных функций - action объявленных в xml файле описания объекта Драйвера. В функциях action производится вызовы команд передаваемых в подключаемую часть Драйвера под управлением модуля Ports.

При разработке драйвера необходимо реализовать следующие функции action:

- Для точек доступа - функции изменения режима работы: включить нормальный режим, заблокировать, разблокировать. При передаче команд в ПО Sigur Драйвер должен использовать действительные идентификаторы, хранящиеся в атрибуте external\_id.
- Для объекта драйвер Sigur - команды импорта конфигурации: импортировать точки доступа, импортировать объекты доступа, запрос на импорт режимов. При импорте данных Драйвер для своих объектов (точка доступа, режим) должен записывать полученные идентификаторы Sigur в поле external\_id.
- Для объекта драйвер Sigur - статические функции для изменения объектов доступа: активировать пропуск, деактивировать пропуск, заблокировать пропуск. В данные функции, в качестве параметра, будет передан объект Пропуска ESM. Описание объекта Пропуск и других сущностей драйвера Общие объекта приведено в [Приложение 5 - Описание сущностей Общих объектов](#).

Функция "Активировать пропуск", должна отправить команду на создание нового объекта доступа, если атрибут внешний идентификатор объекта Пропуск не содержит идентификатор Sigur. В результате создания нового объекта доступа в Sigur, драйвер должен получить идентификатор нового объекта доступа (Сотрудника) и записать его в активированный пропуск. Если пропуск уже имеет внешний идентификатор, то драйвер должен отправить команду на редактирование объекта доступа.

Функция "Деактивировать пропуск", должна отправить в Sigur команду редактирования объекта доступа (Сотрудник), с параметром удаления/очистки кода карты (номера пропуска).

Функция "Заблокировать пропуск", должна отправить в Sigur команду редактирования объекта доступа (Сотрудник), с установкой недействительного срока действия объекта доступа (Сотрудника).

## 4 ОБЩИЕ ТРЕБОВАНИЯ

- Используемая ОС – Разработка должна быть кросс-платформенной.
- Подключаемая часть драйвер должна быть реализован в виде модуля Python, которая будет Ports. Требования к интерфейсу библиотеки приведены в приложении 2.
- Драйвер должен иметь вести журнал своей работы в лог файле. В лог файл следует помещать, сообщения о изменении состояния связи, о выполнении команд, об ошибках в работе драйвера.
- При наличии связи с ПО Sigus должен сразу передавать команду. Если связи нет, то драйвер должен вернуть результат о неуспешности выполнения команды с признаком об отсутствии связи.

### 4.1 Требования к масштабируемости

- Драйвер должен взаимодействовать с одной программной системой Sigur.
- Количество точек доступа неограниченно.

### 4.2 Требования к надежности

Драйвер должен работать в режиме 24x7.

### 4.3 Требования к производительности

Драйвер должен обеспечивать передачу сообщений о событиях и состояниях, передачу команд со скоростью не менее 400 сообщений в секунду.

### 4.4 Требования к средствам разработки и производства


Драйвер должен быть разработан библиотека на языке Python 3.7, команды и сообщения передаются в виде JSON нотации. Интерфейс библиотеки описан в Приложении 3.

Исполнитель должен согласовывать с Заказчиком использование необходимых библиотек Python.

### 4.5 Критерии удовлетворения требований

Критерием удовлетворения бизнес требований является успешная работа драйвера в течение 3 месяцев.

## 5 ИСТОРИЯ ИЗМЕНЕНИЙ ДОКУМЕНТА

ФИО	Описание	Дата	Версия
Алексеев А.В.	Первая версия ТЗ	10.05.2018	v1.0
Алексеев А.В.	Актуализация приложений. Поправка версии Python	 21.12.2019	v1.1

## **6 ПРИЛОЖЕНИЕ 1 - ОБЩЕЕ ОПИСАНИЕ ИНТЕГРАЦИИ ВНЕШНИХ СИСТЕМ**

### **6.1 Введение**

ПО ESM в рамках обеспечения безопасности объектов, выступает интеграционной платформой для объединения подсистем безопасности в единый комплекс. В качестве одной из подсистем безопасности может выступать система контроля и управления доступом (СКУД), далее называемая внешней системой.

#### **6.1.1 Программный комплекс ESM**

Программное обеспечение ESM используется для построения комплексных систем безопасности от локального до федерального масштаба. ПО позволяет решить весь список задач по обеспечению инженерно-технической безопасности объекта и настройке системы управления режимом с выведением ситуации под контроль руководства.

#### **6.1.2 Основные задачи ESM**

Задачи, решаемые ПО в качестве основной интеграционной платформы системы безопасности:

1. Конфигурация подключаемого оборудования;
2. Управление базами данных;
3. Сбор, обработки и анализ информации;
4. Обеспечение пользовательских интерфейсов;
5. Защита информации и разделение доступа к ней;
6. Управление подсистемами безопасности;
7. Интеграция с информационными системами управления.

#### **6.1.3 Подсистема драйверов интеграции**

Подсистема драйверов интеграции предназначена для обеспечения функционала взаимодействия со сторонними системами или приборами.

Описание общих технических решений по разработке драйверов интеграции ESM приведено в Описании интеграции внешних систем.

Согласно указанному техническому решению данная разработка включает реализацию Драйвера из двух частей:

- логическая часть (protocol в sServer), описывающая модель данных интеграции и обработку потока данных;
- подключаемая часть (Ports), реализующая взаимодействие с внешней системой и преобразование нотации протокола в/из структуру JSON принятую в ESM.

Описание структур приведено в Описание протокола взаимодействия sServer и Ports.

- дополнения редактора лицензий ESM - ESM\_HaspMemoryEditor.editor.py
- дополнения справки по лицензиям - sServer.cmd.getLicenseHtml.py, описание лицензии взять из документа с оценкой интеграции.

## 6.2 Функциональные требования

### 6.2.1 Драйвер интеграции должен обеспечивать:

1. Установление и удержание сеанса связи, генерацию нотификаций об изменении состояния сеанса связи с внешней системой. После успешной установки связи Драйвер должен опросить состояния Точек доступа, и сгенерировать нотификации с полученными состоянием для каждой точки доступа. При потере связи с внешней системой, драйвер должен сгенерировать нотификации о потере связи с каждой точкой доступа.
2. После успешной установки связи драйвер должен получать события от внешней системы. Для всех полученных сообщений Драйвер должен сгенерировать нотификации в формате JSON, где Точка доступа должна выступать в роли directObject, Объект доступа - в роли subject. Перечень необходимых событий определяется возможностями конкретной внешней системой.
3. Синхронизировать время со внешней системой (либо, при невозможности устанавливать время во внешней системе, корректировать время в исходящих нотификациях).
4. Автоматизированный (по команде) импорт списка объектов доступа (сотрудниках, транспорте, материальных ценностях). Процедура импорта должна создавать/редактировать в БД ESM субъекты и их пропуска. При импорте необходимо предусмотреть возможность обновления. Для вновь создаваемых Субъектов и Пропусков в ESM, требуется создать именной Уровень доступа, связать его с Уровнем доступа, для нового Уровня доступа необходимо создать объект Право доступа, и назначить данное Право пропуску ESM. Если про импорте производится обновление уже существующего Пропуска, то обновлять (редактировать) существующий уровень доступа не требуется, а также не требуется создавать новый Идентификатор (common\_idcode), и назначать его Пропуску.

#### При наличии технической возможности:

- a. производить импорт списка объектов доступа при изменении оных (добавлении/удалении/редактировании) во внешней системе;
- b. импортировать еще и дерево организаций, временные зоны и расписания и прочие опциональные вещи из внешней системы.



5. Автоматизированный (по команде) импорт списка Точек доступа. Предусмотреть возможность обновления существующего списка Точек доступа, при повторном импорте. Для вновь создаваемых точек доступа в процедуре импорта требуется обеспечить автоматическое создание объектов Право доступа (common\_commonright), и установку ссылки из Права доступа на новую точку доступа.
6. При обработке команд активации/деактивации/блокировки/разблокировке Пропуска в ESM, выполнять соответствующие запросы во внешней системе. Для поддержки данного функционала, необходимо реализовать в драйвере выполнение команд изменения режима работы точек доступа во внешней системе.

Данный список требований может меняться в зависимости от возможностей внешней системы.

### 6.2.2 Требования к модели данных драйвера

При разработке Драйвера в логической модели необходимо описать сущности определяющие БД интеграции. Правила описания приведены в [Декларативное описание объектов оборудования](#).

В общем случае драйвер может содержать следующие объекты:

**Драйвер** - объект описывающий подключаемую часть драйвера и его параметры:

- адрес и порт Ports - строка, например 127.0.0.1,8001 - задает адрес соединения с подключаемой частью Драйвера;
- адрес сервера внешней системы, порт связи, логин и пароль - параметры подключения к внешней системе;
- флаг включения/выключения Драйвера - определяет активность драйвера;
- константа \_\_obsobjtype равная строке "dev" для поддержки логики работы ESM.

Обязательные Action-ы объекта:

- activate (параметр permit - объект пропуска ESM) - активация пропуска, драйвер должен отправить команду на создание нового объекта доступа, если атрибут внешний идентификатор объекта Пропуск не содержит идентификатор во внешней системе (поле subsystems\_ids). В результате создания нового объекта доступа во внешней системе, драйвер должен получить идентификатор нового объекта доступа (Сотрудника) и записать его в активированный пропуск. Если пропуск уже имеет внешний идентификатор, то драйвер должен отправить команду на редактирование объекта доступа
- deactivate (параметр permit - объект пропуска ESM) - деактивация пропуска, драйвер должен отправить во внешнюю систему команду редактирования объекта доступа (Сотрудник), с параметром удаления/очистки кода карты (номера пропуска)

- block (параметр permit - объект пропуска ESM) - блокировка пропуска, драйвер должен отправить во внешнюю систему команду редактирования объекта доступа (Сотрудник), с установкой недействительного срока действия объекта доступа (Сотрудника)
- deblock (параметр permit - объект пропуска ESM) - разблокировка пропуска

**Точка доступа** - объект описывающий элемент внешней системы обеспечивающий преграждение и доступ пользователей СКУД. Параметры точки доступа:

- наименование (Description) - описание точки доступа;
- внешний идентификатор (external\_id) - идентификатор точки доступа во внешней системе;
- константа \_\_objtype равная строке "reader".

Обязательные Action-ы объекта:

- readerDeblock - разблокировать считыватель
- readerBlock - блокировать считыватель
- readerArm - поставить считыватель на охрану (параметр Delay - задержка перед постановкой)
- readerDesarm - снять считыватель с охраны (параметр Delay - задержка перед снятием)
- readerAcceptAlarm - принять тревогу считывателя

**Уровень доступа** - объект задающий список точек доступа назначаемый объекту доступа (пользователю). Данный объект должен быть связан с объектом "Права" драйвера "Общие объекты" (или являться им). При активации пропуска с таким правом, во внешней системе должен быть создан/обновлен объект доступа и ему должны быть назначены точки доступа, входящие в объект "Уровень доступа", а также назначен режим работы установленный в объекту уровень доступа. Атрибуты объекта Уровень доступа:

- наименование (Description) - описание точки доступа;
- список точек доступа (ap\_list) - задает набор объектов "Точка доступа".

Описанные выше типы - общий случай, в интегрируемой системе так же могут быть и другие типы объектов. В этом случае их тоже необходимо добавить в модель и реализовать получение событий/состояний по ним и выполнение у них действий.

### 6.2.3 Требования к использованию существующих данных ESM

Для работы с объектами доступа драйвер должен использовать существующие сущности ESM драйвера "Общие объекты" (common\_protocol\_config). Минимальный набор используемых общих объектов:

**Организация** (common\_organization)

**Подразделение** (common\_department)

**Должность** (common\_post)

**Субъект**(common\_subject) к которому обязательно должен быть привязан один объект одного из перечисленных типов:

- **Частное лицо** (common\_person)
- **Материальная ценность** (common\_thing)
- **Транспортное средство** (common\_mobile)

**Права доступа** (common\_commonright)

В системе ESM объект common\_commonright является контейнером ссылок на частные уровни доступа драйверов интеграции подсистем. При разработке Драйвера интеграции, для определения назначаемого Уровня доступа в СКУД, необходимо использовать данный контейнер.

**Пропуск** (common\_permit) - является источником данных для объекта доступа во внешней системе, а именно номер пропуска, PIN, срок действия. Объект "Пропуск" в ESM является базовым объектом для определения экспортируемых данных в интегрируемую систему. Пропуск имеет:

- ссылку на сущность Субъект (common\_subject), определяющий частное лицо;
- ссылку на сущность идентификатор (common\_idcode), определяющий код карты;
- связь с Правом доступа (common\_commonright);
- атрибуты начала действия и окончания действия, определяющие срок действия пропуска во внешней системе;
- атрибут PIN код, определяющий PIN код объекта доступа во внешней системе.

При изменении состояния пропуска (активирован/деактивирован/заблокирован), Драйвер должен обеспечивать выполнение автоматических запросов на изменение состояния объекта доступа во внешней системе. Описание сущностей приведено в Описании сущностей Общих объектов.

#### 6.2.4 Требования к реализации логики обработки данных от внешней системы

Внешняя система должна отдавать состояния точек доступа (либо же присылать события об изменении их состояний), а так же данные о действиях с пропусками и проходах. Требования к обработке этих данных описывается следующими пунктами:

- Драйвер должен на выходе передавать и события, и состояния (если внешняя система чего то из этого не имеет - генерировать на уровне драйвера)
- При получении информации об изменении данных о пропусках/субъектах пропусков/дереве организаций/правах доступа из внешней системы драйвер должен изменить соответствующую информацию в модели данных ESM
- Описание формата уведомлений на выходе драйвера есть в документе "Модель события"
- Можно менять состояние объекта без генерации события с помощью уведомления с кодом 1000 (нужно при первоначальном получении состояний устройств после запуска драйвера, например)

- При первом подключении к внешней системе прислать состояния по всем точкам доступа (при наличии возможности)

Более подробно про события и состояния можно прочитать тут [Правила преобразования событий и состояний](#)

### 6.3 Общие требования

- Вся обработка данных внешней системы преимущественно должна быть в подключаемой части драйвера (в логической части в идеале должны быть только формирование команд, поиск объектов в таблицах для формирования уведомлений, и некие действия при подключении/отключении подключаемой части)
- Требования к протоколу передачи данных между логической частью драйвера и подключаемой содержатся в описании протокола взаимодействия sServer и Ports
- Драйвер должен иметь вести журнал своей работы в лог файле. В лог файл следует помещать, сообщения о изменении состояния связи, о выполнении команд, об ошибках в работе драйвера
- При наличии связи с внешней системой должен сразу передавать команду. Если связи нет, то драйвер должен вернуть результат о неуспешности выполнения команды с признаком об отсутствии связи
- Код подключаемой части должен быть покрыт тестами
- Должны быть функциональные тесты, показывающие степень работоспособности подключаемой части драйвера с другими версиями внешней системы.

#### 6.3.1 Требования к масштабируемости

- Драйвер должен взаимодействовать с одной точкой входа во внешнюю систему (COM порт, REST API, динамическая библиотека или что-то еще)
- Количество наблюдаемых шлейфов сигнализации и разделов неограниченно.

#### 6.3.2 Требования к надежности

Драйвер должен работать в режиме 24x7.

#### 6.3.3 Требования к производительности

Драйвер должен обеспечивать передачу сообщений о событиях и состояниях, передачу команд со скоростью не менее 400 сообщений в секунду.

#### 6.3.4 Требования к средствам разработки и производства

Серверная часть должна быть разработана на языке Python 3.7 и я должна являться частью модуля sServer.

Подключаемая часть может быть разработана с использованием готового транспортного слоя в виде модуля Ports на языках:

- C#
- Python 3.7

Если очевидно целесообразнее разрабатывать подключаемую часть на другом языке - необходимо согласовать это с Заказчиком.

Исполнитель должен согласовывать с Заказчиком использование необходимых библиотек.

### 6.3.5 Критерии удовлетворения требований

Перед приемкой работ код драйвера должен пройти ревью у специалистов Заказчика.

Критерием удовлетворения бизнес требований является успешная работа драйвера в течение 3 месяцев.

Если из интегрируемой системы мы получаем события - то такой же набор событий должен быть создан и в ESM, тогда необходимо только приводить их нужному виду, описанному тут Модель события.

Если же драйвер может только мониторить изменения состояния объектов (как, например, в случае с мониторингом значения тегов через OPC DA), тогда драйвер должен искусственно генерировать события на каждое изменение состояния объекта. Например, предположим что тег zone отображает состояние зоны и тогда при изменении значения с 0 на 1 нужно сгенерировать событие "Зона поставлена на охрану", а с 1 на 0 - "Зона снята с охраны".

В любом событии, а так же с помощью системного события с кодом 1000 можно передавать состояния объектов, указанных в разделе statements события. Состояния представляют из себя словарь states с ключами:

- Часто используемые:
  - CON - подключен
  - ACC - тревога принята
  - ARM - на охране
  - ALRM - тревога
  - ACT - активен
  - TRB - неисправность
- Редко используемые:
  - SGN - сигнал (для ключниц сигнал - ячейке чужой ключ)
  - ERR - ошибка
  - KEY - ключ сдан
  - FAST - часто моргает

- SLOW - медленно моргает
- BLCK - заблокирован
- ATTN - внимание

Перечисленные выше флаги могут принимать значения 1 и 0 и использоваться с объектами любых типов объектов мониторинга.

Полученные от интегрируемой системы события и состояния необходимо логично интерпретировать в набор вышеперечисленных флагов и послать в события ESM.

Так же есть особый флаг CONF, который используется только с типом объектов мониторинга cam, является словарем с ключами ip, port, login, password, storage, type, name. Этот флаг необходим для отображения камер с помощью ESMCamModule и импорта архива модулем sArchiveImporter.

- monitor (разновидность: Монитор), predefined actions:
  - addShow - Добавить камеру на монитор, параметры:
    - cam (тип str, значение по умолчанию 0) - ID камеры
    - control (тип str, значение по умолчанию 0) - 1 - разрешено управление
  - removeAll - Очистить монитор
- reader (разновидность: Считыватель), predefined actions:
  - test - Тестовое действие
  - readerArm - Поставить считыватель на охрану, параметры:
    - delay (тип int, значение по умолчанию 0) - Задержка
  - readerBlock - Блокировать считыватель
  - readerDesarm - Снять считыватель с охраны, параметры:
    - delay (тип int, значение по умолчанию 0) - Задержка
  - readerDeblock - Разблокировать считыватель
  - readerAcceptAlarm - Принять тревогу считывателя
- system (разновидность: Система), predefined actions:
  - deactivatePermit - Деактивировать пропуск, параметры:
    - permitId (тип str, значение по умолчанию ) - Номер пропуска
- dev (разновидность: Переменная), predefined actions:
  - set value - Установить значение переменной, параметры:
    - value (тип int, значение по умолчанию 0) - Значение
- output (разновидность: Замок), predefined actions:
  - outputOn - Запитать замок, параметры:
    - delay (тип int, значение по умолчанию 0) - Задержка
  - outputOff - Распитать замок, параметры:
    - delay (тип int, значение по умолчанию 0) - Задержка
  - outputFast - Заморгать быстро, параметры:

- delay (тип int, значение по умолчанию 0) - delay
- nparam (тип str, значение по умолчанию ololo) - Параметр
- some\_param (тип int, значение по умолчанию 13) - Какой-то параметр
- outputSlow - Заморгать медленно, параметры:
  - delay (тип int, значение по умолчанию 0) - Задержка
- outputFast2 - test outputFast2
- startReload - Прогрузить выход в PCE
- outputRestore - Восстановить выход
- input (разновидность: Кнопка выход), предопределенные действия:
  - inputBlock - Блокировать кнопку выхода
  - inputDeblock - Разблокировать кнопку выхода
- input (разновидность: ОШС), предопределенные действия:
  - inputArm - Поставить ОШС на охрану
  - inputDesarm - Снять ОШС с охраны
- dev (разновидность: Видеосервер), предопределенные действия:
  - restartAll - Перезагрузить все модули
- group (разновидность: Офис), предопределенные действия:
  - upShutters - Поднять группу рольставень
  - armPartFloor - Поставить офис на охрану
  - downShutters - Опустить группу рольставень
  - desarmPartFloor - Снять офис с охраны
  - acceptPartFloorAlarm - Принять тревогу офиса
- group (разновидность: Точка доступа), предопределенные действия:
  - armAccessPoint - Поставить точку доступа на охрану, параметры:
    - delay (тип int, значение по умолчанию 0) - Задержка
  - desarmAccessPoint - Снять точку доступа с охраны, параметры:
    - delay (тип int, значение по умолчанию 0) - Задержка
- group (разновидность: Этаж), предопределенные действия:
  - armFloor - Поставить этаж на охрану
  - desarmFloor - Снять этаж с охраны
  - acceptFloorAlarm - Принять тревогу этажа
- group (разновидность: Здание), предопределенные действия:
  - armBuilding - Поставить здание на охрану
  - desarmBuilding - Снять здание с охраны
  - acceptBuildingAlarm - Принять тревогу у здания
- group (разновидность: Раздел ОШС), предопределенные действия:
  - armPart - Поставить раздел ошс 16 на охрану
  - desarmPart - Снять раздел ошс 16 с охраны



- acceptPartAlarm - Принять тревогу раздела ошс 16
- group (разновидность: Помещение), predetermined actions:
  - desarmRoom - Снять помещение с охраны
  - acceptRoomAlarm - Принять тревогу помещения
- input (разновидность: Дверь), predetermined actions:
  - inputArm - Поставить дверь на охрану, параметры:
    - delay (тип int, значение по умолчанию 0) - Задержка
  - inputBlock - Блокировать дверь, параметры:
    - delay (тип int, значение по умолчанию 0) - Задержка
  - inputDesarm - Снять дверь с охраны, параметры:
    - delay (тип int, значение по умолчанию 0) - Задержка
  - inputDeblock - Разблокировать дверь, параметры:
    - delay (тип int, значение по умолчанию 0) - Задержка
  - inputAcceptAlarm - Принять тревогу двери
- output (разновидность: Исполнительное устройство), predetermined actions:
  - outputOn - Запитать выход, параметры:
    - delay (тип int, значение по умолчанию 0) - Задержка
  - outputOff - Распитать выход, параметры:
    - delay (тип int, значение по умолчанию 0) - Задержка
  - outputFast - outputFast, параметры:
    - delay (тип int, значение по умолчанию 0) - zaderjka
- cam (разновидность: Видеокамера), predetermined actions:
  - arm - Поставить на охрану
  - rec - Включить запись
  - move - move, параметры:
    - pan (тип int, значение по умолчанию 0) - По горизонтали
    - titl (тип int, значение по умолчанию 0) - По вертикали
    - zoom (тип int, значение по умолчанию 0) - Увеличение
  - rec2 - Начать запись, параметры:
    - time (тип int, значение по умолчанию 0) - Время
    - rollback (тип int, значение по умолчанию 0) - Предзапись
  - stop - stop
  - disarm - Снять с охраны
  - recStop - Выключить запись
  - goPreset - Перейти в пресет, параметры:
    - preset (тип int, значение по умолчанию 0) - Номер пресета



- setPreset - Установить пресет, параметры:
  - preset (тип int, значение по умолчанию 0) - Номер пресета
- ~showVideo - Показать видео
- clearPreset - Очистить пресет, параметры:
  - preset (тип int, значение по умолчанию 0) - Номер пресета
- startCleaning - Включить стеклоочиститель

Все упоминаемые здесь файлы можно найти в архиве **ESM\_SDK.7z**, предоставляемом по запросу.

### Разворачивание стенда ESM для разработки интеграции:

1. Установить сервер и клиент из дистрибутивов **ESM\_Server\_XXXX.Y.F\_DDK.exe** и **ESM\_Client\_XXXX.Y.F.exe**
2. Распаковать модули sServer и sAuth из папки **Modules**
3. Для их работы нужно поставить Python 2.7 и модули из папки **packages** скриптом **\_install.bat**
4. Скопировать файл settings.py из каталога в который установился сервер из дистрибутива в папки с модулями sServer и sAuth из архива
5. Для работы сервера необходимо запустить модули sServer, sAuth и sDisplay, логин и пароль по умолчанию admin
6. Можно запускать клиент и подключаться

### Общий порядок действий по добавлению новой интеграции в ESM:

1. Добавить в sServer папку *%название оборудования%\_protocol\_config* и заполнить ее xml-ками с описанием таблиц
2. Сделать миграции для добавления в БД соответствующих таблиц и кодов событий в таблицу event\_catalog (примеры миграций и инструкция по созданию новых в папке **migrations**)
3. Разработать подключаемый модуль
  - a. Если модуль пишется на Python 2.7, то следует добавить свою реализацию в существующий модуль Ports (в папке **Ports**). Для этого нужно добавить папку *%название оборудования%* в папку (по аналогии с существующими) и добавить в ports.py строки по аналогии с:

```
elif device_name == "%название оборудования%":  
    from %название оборудования%.Driver%название оборудования% import  
    Driver%название оборудования%  
    self.__driver = Driver%название оборудования%  
(self.__eventHandler, self.__stateHandler)
```
  - b. Если модуль пишется на C#, то можно использовать реализацию транспорта, да и просто подсмотреть решение из папки **Ports\_RubezhGlobal**

**Список вопросов от подрядчиков:**

1. У меня не запускается сервер с ImportError: No module named core
  - a. core передается в event\_packet снаружи. Так же есть возможность вместе с xml использовать классы для реализации action-ов (удобнее отлаживать и не надо писать код в xml), пример такого описания объектов можно найти в папке **plavnik\_protocol\_config**
2. Ошибка 'module' object has no attribute 'EventPacketException'
  - a. В event\_packet нужно определить этот класс и еще функцию decodeCommandResult, которая будет вызываться, когда портс пришлет синхронный ответ на отправленную сервером команду:

```
class EventPacketException(Exception):  
    pass  
def decodeCommandResult(core, rootElement, cmdRes, cmdName):  
    return cmdRes
```
3. Все запускается без ошибок, но в дереве устройств *%название оборудования%* почему-то не появляется
  - a. Нужно в клиенте открыть через меню панель Настроек и там на последней вкладке дать администратору права на добавленное оборудование
4. По какому принципу объектам назначать типы объектов мониторинга (константу obsobjtype)?
  - a. Выбрать наиболее подходящий из Список типов объектов мониторинга и их предопределенных действий и, при наличии технической возможности, реализовать у объекта все предопределенные в списке действия (если объект интегрируемой системы может выполнять действия, которых нет в списке предопределенных - их все равно нужно реализовать, назвав на свое усмотрение)

## 7 ПРИЛОЖЕНИЕ 2 - ОПИСАНИЕ ПРОТОКОЛА ВЗАИМОДЕЙСТВИЯ SSERVER C PORTS

### 7.1 Формат сообщений драйвера

Сообщение о событии предназначено для передачи информации от подключаемой части драйвера в серверную часть о произошедших фактах изменения состояния объектов интегрируемой системы или его параметрах. Событие описывается JSON структурой следующего формата

Event	
1	{
2	"event": {
3	"code": число, #код события
4	"obj": число, #идентификатор объекта-источника события
5	"type": строка, #тип объекта
6	"time": время, # UNIXTIME-время события
7	"params": #параметры события [
8	"<имя параметра>": "<значение>",
9	"<имя параметра>": "<значение>",
10	"<имя параметра>": "<значение>"
11	]
12	}
13	}

В массиве params должны передаваться дополнительные параметры, не определенные в основных переменных сообщения.

Состояние предназначено для передачи информации о состоянии объекта. Драйвер передает данные сообщения при выполнении команд опроса состояния объектов, например, после успешного подключения к интегрируемой системе, и/или при получении сообщения от интегрируемой системы об изменении состояния какого-либо объекта. Сообщение о состоянии описывается JSON структурой следующего формата:

State	
1	{
2	“state”: {
3	“obj”: число, #идентификатор объекта-источника события
4	“status”: строка, #код состояния
5	“time”: время, # UNIXTIME-время события
6	“params”: #параметры состояния [
7	“<имя параметра>”: “<значение>”,
8	“<имя параметра>”: “<значение>”,
9	“<имя параметра>”: “<значение>”
10	]
11	}
12	}

Команды предназначены для управления объектами интегрируемой системы, или для получения данных из интегрируемой системы. Драйвер должен обеспечивать асинхронное выполнение команд. При получении команды, драйвер должен проверить команду и вернуть ответ сразу.

Возвращаемые значения Драйвера на получение команд.

{"status" : "ok"} – команда принята успешно

{"status" : "json format error"} – не удалось десериализовать команду

{"status" : "format error"} – ошибка формата команды.

{"status" : "command not found"} – команда не найдена.

Команда должна описываться JSON структурой следующего формата:

Command	
1	{
2	“cmd”: “<имя команды>”,
3	“cmdId”: “<уникальный идентификатор команды>”, # используется для
4	отправки асинхронного ответа на команду.
5	“params”: #параметры команды [
6	“<имя параметра>”: “<значение>”,
7	“<имя параметра>”: “<значение>”,
8	“<имя параметра>”: “<значение>”
9	]
10	}

Если команда предполагает возвращение результата из интегрируемой системы, то Драйвер должен вернуть результат при помощи передачи сообщения cmdResponse.

### cmdResponse

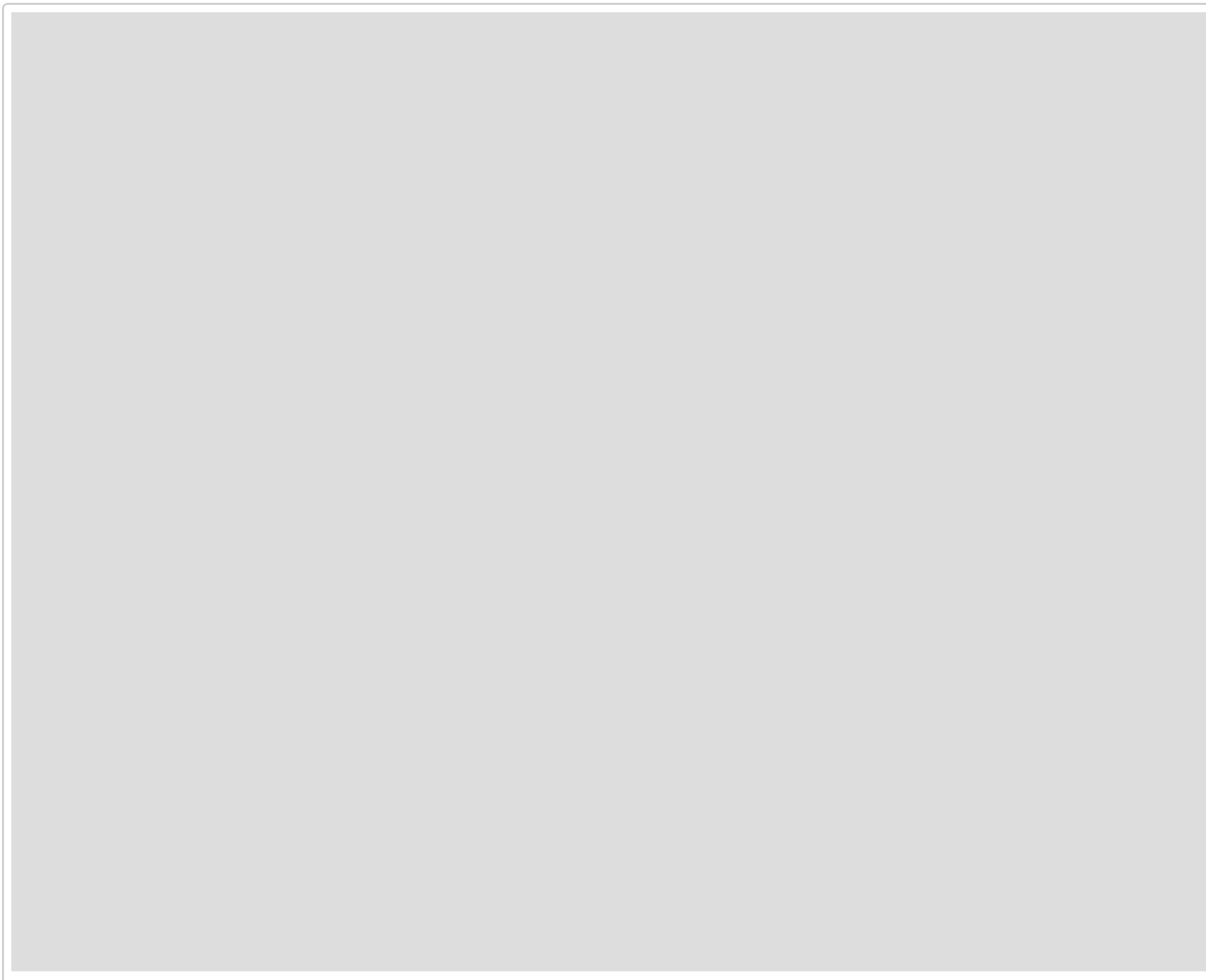
```

1  {
2      "cmdResponse": {
3          "cmd": "<имя команды>",
4          "cmdId": "<уникальный идентификатор команды>", взять из
команды, на которую производится ответ.
5          "status": "результат обработки команды"*
6          "params": #параметры ответа(данные) [
7              "имя параметра":<значение>, #число, время,
строка, время, массив
8              "имя параметра":<значение>,
9              "имя параметра":<значение>
10             ]
11         }
12     }
13 }
```

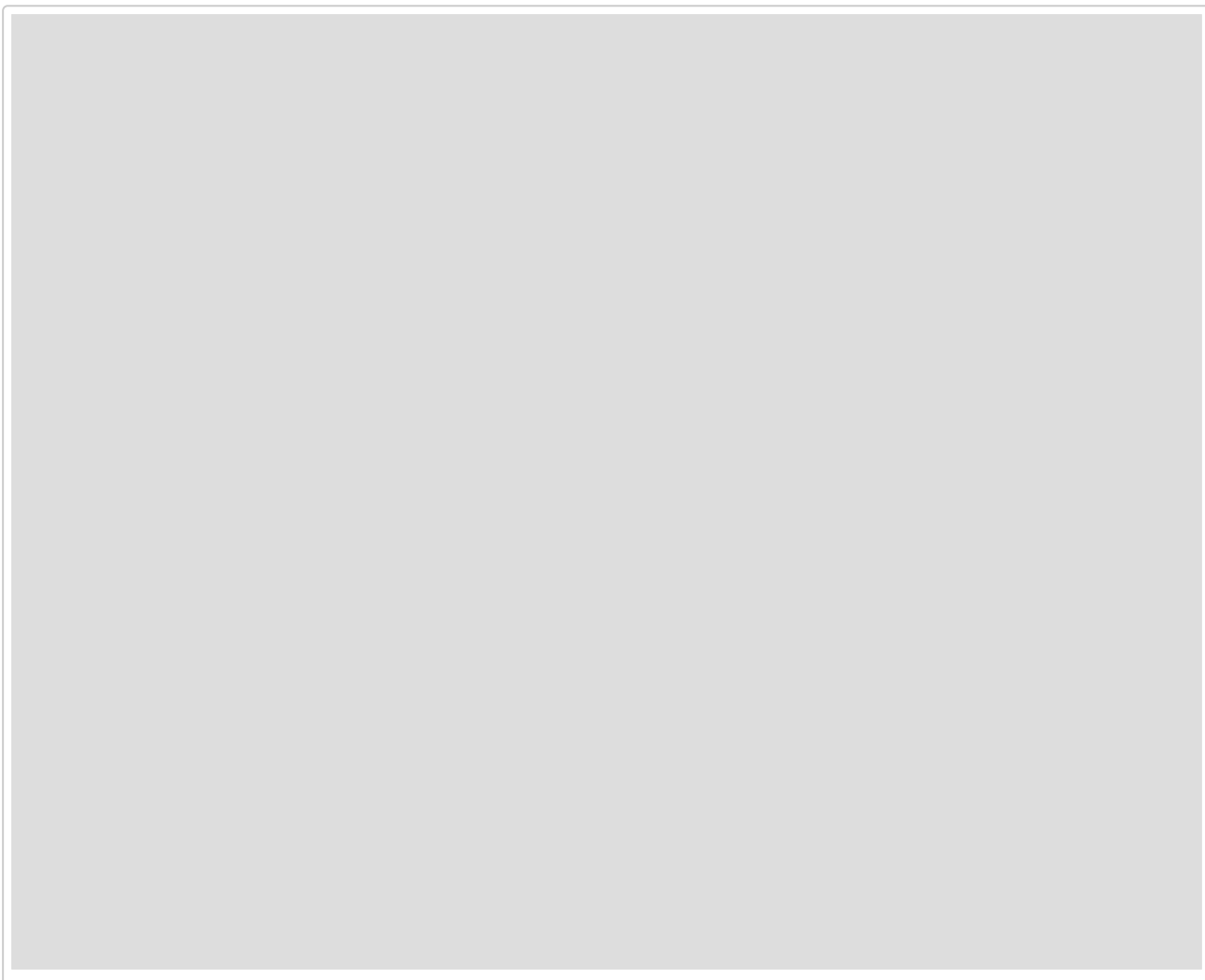
\* - поле status определяется протоколом интегрируемой системы и должно учитываться при логической обработке результата в серверной части

## 8 ПРИЛОЖЕНИЕ 3 - ОПИСАНИЕ ИНТЕРФЕЙСА ИНТЕГРАЦИИ ПО SIGUR

### 8.1 Текстовый протокол Sigur



## 8.2 Структура БД Sirug



## 9 ПРИЛОЖЕНИЕ 4 - ДЕКЛАРАТИВНОЕ ОПИСАНИЕ ОБЪЕКТОВ ДРАЙВЕРА

### 9.1

#### Общие сведения

Каждое оборудование в sServer представляется отдельной папкой с названием <имя оборудования>\_protocol\_config, где <имя оборудования> строка состоящая только из строчных латинских букв и цифр например rubej08\_protocol\_config

В каждой папке обязательно находится следующие файлы

- \_\_init\_\_.py - хранит информацию о названии оборудования и главном объекте
- event\_packet.py - хранит часть обязательного кода необходимого для работы сервера, а так же любой код необходимый для работы функционирования драйвера;
- <имя типа оборудования>\_\*.py - произвольное число python файлов по одному на каждый тип объекта данного оборудования.

### 9.2 Создание оборудования

Для создания нового оборудования нужно сделать следующие действия

1. С помощью механизма миграций добавить в таблицу equipments запись с названием создаваемого оборудования.
2. Создать в sServer папку в которой будут размещаться файлы с описанием этого оборудования по формату <имя оборудования>\_protocol\_config.
3. Создать в этой папке минимальный набор необходимых файлов (см прошлый раздел).
4. Для каждого типа оборудования создать python файл с его описанием.
5. Добавить оборудование в модуль создание лицензий ESM\_HaspMemoryEditor, в файл editor.py переменная equip\_name. Добавить оборудование в модуль licensemanager в файл licensemanager.py переменная equipments
6. Добавить описание лицензии на драйвер в sServer\getLicenseHtml.py

### 9.3 Описание форматов файлов

#### 9.3.1 \_\_init\_\_.py

Файл \_\_init\_\_.py должен содержать одну обязательную переменную и может содержать одну не обязательную.

- alias - русское название оборудование (для отображения в клиенте, обязательный параметр)
- rootDev - имя типа, который инициализирует подключение к модулю Ports (если он есть, необязательный параметр).

Пример:



```
# -*- coding: utf-8 -*-  
  
alias = 'Шины контроллеров PCE'  
rootDev = 'bus'
```

### 9.3.2 <имя оборудования>\_\*.py



#### Важно!

Типы объектов входящих в состав оборудования должны иметь названия состоящие только из строчных латинских букв и цифр, например input. Тогда файл, содержащий описание этого типа, будет называться pce\_input.py.

В процессе создания оборудования может потребоваться организовать связь многие-ко-многим между двумя типами, например между bus и command. Тогда файл будет называться pce\_bus\_command.py.

### Формат файла описания объекта оборудования

Данный файл представляет собой python класс унаследованный от protocol\_obj\_base, содержащий все действия объекта (доступные из клиента), служебные методы, а так же ссылки, атрибуты и другие необходимые переменные. Далее приведен пример того как может выглядеть такой файл.

```
from equipment.protocol_obj_base import protocol_obj_base, action_decorator,
Attribute, Link, ParentStruct
from equipment import dev_except

from pceapi.dev_types import TypesList, DevType
from pceapi.objects.address import DeviceAddr
from pceapi.commands.base import Command
import pceapi.commands.input as commands
import pceapi.commands.common as common_commands

from .pce_area import pce_area
from .pce_string import pce_string
from .pce_adgroup import pce_adgroup

from .utils import getCurrOperatorAddr

class pce_input(protocol_obj_base, alias='Вход', parent=ParentStruct('node',
'Входы')):
    OBSOBJTYPE = 'input'
    _DEV_TYPE = TypesList.Input

    def sendCommand(self, cmd: Command):
        return self.sendToPorts(cmd.getName(), cmd.toDict())

    @classmethod
    @action_decorator(alias='Прогрузить все', actionClass='static')
    def reloadAll(cls):
        res = ''
        for o in cls._core.getElements('input'):
            try:
                o.doAction('startReload')
                res = """"%s<div style="color: green;">%s '%s' %s </div>"" % (
                    res, o.getDescription(),
cls._core.getString('pceReloadSuccess'))
            except Exception as e:
                res = """"%s<div style="color: red;">%s '%s': %s</div>"" % (
                    res, cls._core.getString('pceReloadError'),
o.getDescription(), e)

        return {
            'txt' : res,
            'timeout' : 30000,
            'type' : 'info'
        }

    @action_decorator(alias='Шунтировать вход на время Delay', delay='Задержка')
    def inputBlock(self, delay: int = 0):
        return self.sendCommand(commands.InputBlock(self.getPceAddr(), delay,
getCurrOperatorAddr(self.getParent(3))))
```

```
def __getObsObjType(self, field):  
    return self.OBSOBJTYPE  
  
description = Attribute(alias='Описание', fieldType=str)  
name = Link('Строка имени', target=pce_string, postAction=__startReload)  
obsobjtype = Attribute(alias='Тип объекта мониторинга', index=100,  
    fget=__getObsObjType, readOnly=True,  
    storeInDb=False)
```

### Описание названия класса

Название класса совпадает с названием файла и пишется строчными буквами. Класс должен быть обязательно унаследован от класса `protocol_obj_base`. Далее могут идти параметры передаваемые мета классу для создания типа.

**alias** - определяет русское название отношения родства, которое будет отображаться в клиенте ESM. Обязательный параметр.

**parent** - информация о родительском оборудовании (если оно есть). Значение должно быть объектом класса `ParentStruct`. Конструктор данного класса принимает следующие параметры

- **typeName** - английское название родительского типа оборудования, например 'node'
- **alias** - определяет русское название отношения родства, которое будет отображаться в клиенте ESM.
- **addr** - определяет автогенерацию адреса ребенка у родителя. Если ген отсутствует то пользователь обязан сам указать адрес при привешивании.

**archive** - вместо удаления объекта он помечается как удаленный выставлением 1 в колонке `flags` в БД

**uniqueBridge** - используется только для объектов мостовых связей (многие ко многим). Если выставлен то будет проверяться уникальность связи при создании

**hidden** - оборудование является внутренним и не экспортируется в клиент

### Описание атрибутов

Атрибуты типа объекта описываются как `property` класса. Для этого используется класс `Attribute`. Описание всех принимаемых значений конструктором `attribute` можно прочитать в исходном коде.

Список возможных значений `editorType` можно посмотреть в приложении 2.

Атрибуты типа объекта описываются как property класса. Для этого используется класс Link. Описание всех принимаемых значений конструктором attribute можно прочитать в исходном коде.

Системные столбцы такие как uniid, flags, devparent, devaddr описывать не нужно.

### Описание индексов

Индексы описываются как обычные public методы класса отмеченные специальным декоратором index\_decorator. Пример:

```
@index_decorator('globalID')
def globalIDKey(self):
    if self.getParent() is None or self.id == '':
        return None
    else:
        return '%s.%s' % (self.getParent().getUniID(), self.id)
```

Метод расчета индекса должен вернуть значение, которое потом будет использоваться для поиска. index\_decorator принимает один обязательные параметр name - имя индекса, которое потом будет использоваться для поиска.

### Описание публичных действий

Данные действия будут доступны для вызова из интерфейса клиента esm, или из типа объекта мониторинга.

Действия описываются как обычные public методы класса отмеченные специальным декоратором action\_decorator. Пример таких действий есть в начале описания. Все параметры методов-действий должны обязательно иметь значение по умолчанию и тип. На основании этого типа будет выбираться редактор для ввода значения в интерфейсе клиента ESM. action\_decorator принимает следующие параметры:

**alias** - русское название действия (для отображения в клиенте)

**actionClass** - не обязательный параметр, который может принимать следующие значения; **static** - действие будет доступно не у конкретного объекта, а у типа целиком; **fastCall** - при выполнении действия не будет выводиться запрос на подтверждение оператору

Далее должно идти русское название каждого параметра (см. пример)

### Системные методы

Есть 9 действий с фиксированными названиями - init, finit и onEvent, preCreate, postCreate, preDelete, postDelete, preChangeParent, postChangeParent. Все действия объявлены в базовом классе и могут быть переопределены по необходимости. Сигнатуру методов можно посмотреть в коде и в Приложении 1.

- `init` вызывается сразу после загрузки объекта из БД и обычно используется для инициализации подключения к `ports`.
- `finit` вызывается при выключении сервера и обычно используется для деинициализации подключения к `ports`
- `onEvent` вызывается каждый раз когда от `port` приходит новое событие
- `preCreate` статическое действие которое вызовется перед созданием объекта данного типа. Если будет вызвано исключение то объект не будет создан
- `postCreate` вызывается после создания объекта
- `preDelete` вызывается перед удалением объекта. Если будет вызвано исключение то объект не будет удален
- `preChangeParent` вызывает перед изменением родителя у объекта
- `postChangeParent` вызывается после изменения родителя у объекта

### Отправка команд

Для отправки команды в модуль `ports` необходимо вызвать метод `sendToPorts`. Он принимает 2 параметра - имя команды и `json` с данными которые будут отправлены.

## 9.4 Приложение 1. Параметры системных действий

### 9.4.1 `init(self)`

Параметров нет

### 9.4.2 `finit(self)`

Параметров нет

### 9.4.3 `onEvent(self, event=None, portsEvent=None)`

Принимает 2 параметра `event` и `portsEvent`. Одновременно значение может иметь только один. `portsEvent` используется для системных событий `Ports`. Для события подключения будет такое значение `{'type': 'connect', 'time': time.time()}`. Для события отключения будет такое значение `{'type': 'disconnect', 'time': time.time()}`.

Для событий отправляемых оборудованием будет параметр `event`. Значение параметра будет то что отправил `Ports`

### 9.4.4 `preCreate(cls, createAttrs=None, targetObj=None, mainObj=None, info=None)`

- `createAttrs` - словарь с атрибутами, которые были переданы в запрос `createEquipDev` или в метод `createElement` при создании объекта
- `targetObj` - используется только при создании мостовой связи. Является объектом с которым устанавливается связь
- `mainObj` - используется только при создании мостовой связи. Является объектом который устанавливает связь

- info - ???

#### 9.4.5 postCreate(self, ignore\_create\_objects=False)

ignore\_create\_objects - ???

#### 9.4.6 preDelete(self, deleteAsLink=False)

deleteAsLink - принимает значение True при удалении объектов мостовых связей

#### 9.4.7 postDelete(cls, \*\*kwargs)

Не известно что там, реально никогда не использовалось

#### 9.4.8 preChangeParent(self, newParent=None, addr=None, info=None)

- newParent - объекта класса BaseElement. Объект который хочет стать родителем текущего объекта. Или None если у объекта убирается родитель.
- addr - адрес который будет присвоен ребенку
- info - ???

#### 9.4.9 postChangeParent(self, oldParent=None, info=None)

- oldParent - объекта класса BaseElement. Объект который был родителем текущего объекта. Или None если у объекта не было родителя.
- info - ???

#### 9.4.10 preAction (Атрибут)

- value - значение, которое будет присвоено атрибуту
- field - имя изменяемого атрибута

#### 9.4.11 postAction (Атрибут)

- oldValue - значение, которое было у атрибута
- oldValues - значение всех других атрибутов, данного объекта

#### 9.4.12 preAction (Ссылка)

- targetObj - объекта класса BaseElement. Объект который будет присвоен в качестве ссылки. Или None если ссылка сбрасывается

#### 9.4.13 postAction (Ссылка)

- targetObj - объекта класса BaseElement. Объект который будет присвоен в качестве ссылки. Или объект который был присвоен, если ссылка сбрасывается

## 9.5 Приложение 2. Типы редакторов значений атрибутов

На данный момент поддерживается набор из перечисленных ниже типов. Все названия типов являются регистрозависимыми. Данные типы определяют какой редактор будет создан в интерфейсе клиента ESM для редактирования поля. Для любого другого типа будет создан QLineEdit

### 9.5.1 int

Будет создан QSpinBox с максимальным значением 1000000

### 9.5.2 date

Будет создан QDateEdit. Данный тип должен использоваться для дат хранимых в БД в столбцах типа date. В get таких атрибутов надо использовать такой код tuple(obj.getAttribute('datestart').timetuple()) иначе ничего не заработает.

### 9.5.3 UnixDate

Будет создан QDateEdit. Данный тип должен использоваться для дат хранимых в БД в виде unix date (число секунд с 01.01.1970)

### 9.5.4 UnixDateTime

Будет создан QDateTimeEdit. Данный тип должен использоваться для дат хранимых в БД в виде unixtime (число секунд с 01.01.1970)

### 9.5.5 time

Будет создан QTimeEdit. Данный тип должен использоваться для полей хранящих время в виде числа секунд с начала суток.

### 9.5.6 datetime

Похоже что тоже самое что и UnixDateTime

### 9.5.7 enum

Будет создан выпадающий список со значениями. Значения должны перечисляться в скобках через запятую в виде enum(Значение1,Значение2,Значение3). В БД данные будут представлены в виде числа где Значение1=0, Значение2=1 и так далее.

### 9.5.8 treeSelect

Будет создан выпадающий список со значениями. Значения должны вернуть действие описанное в скобках, например, treeSelect(getAPBtype). Действие должно вернуть

словарь в котором ключами будут текстовые представления, а значением ключа будет то что хранится в БД. Например:

```
result = {  
'Разрешить один свободный проход' : 1,  
'Разрешить свободный проход' : 2,  
'Установить область' : 3  
}
```

#### **9.5.9 dynamicTreeSelect**

Тоже самое что и treeSelect, но действие указанное в скобках может возвращать разный набор вариантов для выбора в зависимости от объекта, в то время как treeSelect будет предлагать одинаковый выбор для всех объектов данного типа

#### **9.5.10 checkbox**

Будет создан QCheckBox (не рекомендуется использовать для новых разработок)

#### **9.5.11 fileMapdxf**

Специализированный тип для выбора файла формата dxf (используется только в картах).

#### **9.5.12 fileiconico**

Специализированный тип для выбора файла формата ico.

#### **9.5.13 multiSelect**

Тоже самое что и treeSelect, но можно будет выбрать несколько значений. В БД будут храниться в виде строки через запятую.



## 10 ПРИЛОЖЕНИЕ 5 - ОПИСАНИЕ СУЩНОСТЕЙ ОБЩИХ ОБЪЕКТОВ

## 11 ПРИЛОЖЕНИЕ 6 - ОПИСАНИЕ ОБЪЕКТНОЙ МОДЕЛИ ЯДРА

### 11.1 dev\_except module

*exception* dev\_except.BridgeElementNotFound(*linkName*, *typeName*='?', *id*='?',  
*targType*='?', *targId*='?', *ownerDescr*=None, *arrDescr*=None)

Bases: dev\_except.ElementNotFound

*exception* dev\_except.BridgeMustBeUnique(*linkAddr*)

Bases: dev\_except.EquipmentException

*exception* dev\_except.ChildNotFound(*addr*, *typeName*='?', *id*='?')

Bases: dev\_except.ElementNotFound

*exception* dev\_except.CommandExecError(*txt*=u'u041eu0448u0438u0431u043au0430  
u0432u044bu043fu043eu043bu043du0435u043du0438u044f  
u043au043eu043cu0430u043du0434u044b')

Bases: dev\_except.EquipmentException

*exception* dev\_except.CommandExecError2(*txt*=u'u041eu0448u0438u0431u043au0430  
u0432u044bu043fu043eu043bu043du0435u043du0438u044f  
u043au043eu043cu0430u043du0434u044b')

Bases: dev\_except.EquipmentException

*exception* dev\_except.DifferentParent

Bases: dev\_except.EquipmentException

*exception* dev\_except.ElementNotFound(*typeName*='?', *id*='?')

Bases: dev\_except.EquipmentException

element()

elementStr()

*exception* dev\_except.EquipmentException

Bases: exceptions.Exception

*exception* dev\_except.HandlerCompileError(*txt*=u'u041eu0448u0438u0431u043au0430  
u043au043eu043cu043fu0438u043bu044fu0446u0438u0438')

Bases: `dev_except.EquipmentException`

*exception* `dev_except.LinkedElementNotFound(linkName, typeName='?', id='?')`

Bases: `dev_except.ElementNotFound`

*exception* `dev_except.NeedAnswers(form)`

Bases: `dev_except.EquipmentException`

`getForm()`

*exception*

`dev_except.NoEditableField(txt=u'u041du0435u0440u0435u0434u0430u043au0442u0438u0440u0443u0435u043cu043eu0435 u043fu043eu043bu0435')`

Bases: `dev_except.EquipmentException`

*exception* `dev_except.ParentNotFound(typeName='?', id='?')`

Bases: `dev_except.ElementNotFound`

*exception* `dev_except.PortNotFound(txt=u'u041fu043eu0440u0442 u043du0435u043du0430u0439u0434u0435u043d')`

Bases: `dev_except.EquipmentException`

*exception* `dev_except.PostActionFail(txt=u'post action fail')`

Bases: `dev_except.EquipmentException`

*exception*

`dev_except.TerminateAction(txt=u'u0414u0435u0439u0441u0442u0432u0438u0435u043eu0442u043cu0435u043du0435u043du043e')`

Bases: `dev_except.EquipmentException`

*exception*

`dev_except.UndefinedAction(txt=u'u041du0435u0438u0437u0432u0435u0441u0442u043du043eu0435 u0434u0435u0439u0441u0442u0432u0438u0435')`

Bases: `dev_except.EquipmentException`

*exception* `dev_except.UndefinedAttrField(attrName, obj=None)`

Bases: `dev_except.EquipmentException`

*exception* `dev_except.UndefinedExtEquipCore`

Bases: `dev_except.EquipmentException`

*exception* `dev_except.UndefinedLinkField(linkName, obj=None)`

Bases: `dev_except.EquipmentException`

*exception* `dev_except.UndefinedType(typeName='?')`

Bases: `dev_except.EquipmentException`

*exception* `dev_except.WrongChildType(txt=u'wrong child type')`

Bases: `dev_except.EquipmentException`

*exception* `dev_except.WrongLinkTargetType(txt)`

Bases: `dev_except.EquipmentException`

## 11.2 dev\_obj\_model module

```
class dev_obj_model.Model(equipment)
```

Класс для работы с записями таблиц оборудования в объектном стиле

`calculateAttributes()`

`createElement(typeName, createAction=True, attrs=None, info=None)`

`dbCommit()`

`deleteElement(element)`

`farEquip(eqName)`

`getAttrListOfType(typeName)`

`getChildTypes(typeName)`

`getConstListOfType(typeName)`

`getCurrOperator()`

`getCurrOperatorSubject()`

`getDeviceTree(root_item)`

`getElementById(typeName, id)`

`getElements(typeName)`

`getElementsByIndex(typeName, indexName, keyValueDict, keyValue=None)`

`getFirst(typeName)`

`getInterfaceOfType(typeName)`

`getMark()`

`getMessage()`

`getNewDevice(item, dev_type, dev_id)`

`getString(string)`

`getStruct()`

`getTypesList()`

`hasChildOfType(parentType, childType)`

`loadElement(typeName, uniId, devAddr, devParent, cache, parent)`

`loadElements()`

`mutationReportAddChild(obj, addr, parentObj)`

*mutationReportBindObject(linkedObj, linkName, linkOwnerObj, pushNotif=True)*

*mutationReportCreateObject(obj)*

*mutationReportDeleteObject(obj, parentObj=None)*

*mutationReportRemoveChild(obj, parentObj)*

*mutationReportSetAttribute(obj, attrName, attrValue, attrSelectValues,*  
*pushNotif=True)*

*mutationReportSetAttributesOnCreate(notifications)*

*mutationReportUnBindObject(linkedObj, linkName, linkOwnerObj, pushNotif=True)*

*newMark()*

*printDeviceTree()*

*pushEvent(event)*

*sql(req, params=None)*

### 11.3 dev\_obj\_model\_element module

```
class dev_obj_model_element.BaseElement(unlId=None, handlers=None, addr=None,
parent=None, attrValues=None, calculateAttrs=True)
```

```
addChild(child, addr=None, skipTrig=False, info=None)
```

Привязывает к текущему объекту-родителю ребенка

Parameters:	<ul style="list-style-type: none"> <li>• <b>child</b> – объект ребенка, если у родителя нет детей такого типа бросится исключение <code>dev_except.WrongChildType</code></li> <li>• <b>addr</b> – адрес, с которым привяжется ребенок, если адрес = None - будет выброшено исключение</li> <li>• <b>skipTrig</b> – если True - ChangeParentAction-ы выполняться не будут</li> <li>• <b>info</b> – параметр для preChangeParentAction у child-a</li> </ul>
Returns:	

```
attrUpdated(attrName, pushNotif=True)
```

```
bindElement(linkName, el, ignoreInstead=False)
```

Заполняет ссылку `linkName` элементом `el` Если тип целевого объекта не совпадает с типом ссылки будет брошено исключение `dev_except.WrongLinkTargetType` Если при выполнении `postAction` возникнет ошибка будет брошено исключение `dev_except.PostActionFail`

Parameters:	<ul style="list-style-type: none"> <li>• <b>linkName</b> – имя ссылки</li> <li>• <b>el</b> – целевой объект</li> <li>• <b>ignoreInstead</b> – если истина - то даже при наличии у ссылки <code>insteadAction</code> - действие выполнено не будет, а будет присвоена ссылка</li> </ul>
Returns:	возвращает всегда True

```
bindFarObject(obj2EquipName, obj2TypeName, obj2Id, linkName)
```

Создает дальнюю ссылку (между объектами из различного оборудования)



<b>Parameters:</b>	<ul style="list-style-type: none"> <li>• <b>obj2EquipName</b> – Название оборудования целевого объекта</li> <li>• <b>obj2TypeName</b> – тип внутри оборудования целевого объекта</li> <li>• <b>obj2Id</b> – уникальный идентификатор целевого объекта</li> <li>• <b>linkName</b> – название ссылки у текущего объекта</li> </ul>
<b>Returns:</b>	

`buildCommand(cmdName, params=None, rootElement=None)`

Посылает команду в портс. Для этого команда с именем `cmdName` должна быть описана в разделе `<commands>` в xml с описанием текущего объекта. Для разбора ответа на команду будет вызвана функция `decodeCommandResult` из `event_packet`-а оборудования.

<b>Parameters:</b>	<ul style="list-style-type: none"> <li>• <b>cmdName</b> – имя команды</li> <li>• <b>params</b> – словарь с параметрами для подстановки в команду</li> <li>• <b>rootElement</b> – объект-владелец <code>portsClient</code>-а, через который будет отправлена команда. Если <code>None</code> - такой элемент будет найден функцией <code>findRoot</code>. Если все равно не будет найден - будет брошено исключение <code>dev_except.PortNotFound</code></li> </ul>
<b>Returns:</b>	ответ портса на команду

`buildPlugin(pluginClassName, **kwargs)`

`calcAttr(attrName)`

`calcAttrSelectValues(attrName)`

`calcLink(linkName)`

`calculateCache()`

`checkLink(linkName, targetObj)`

Если мостовой таблицы с названием `linkName` между текущим объектом и `targetObj` не существует - будет брошено исключение

Parameters:	<ul style="list-style-type: none"> <li>• <b>linkName</b> –</li> <li>• <b>targetObj</b> – НЕ ИСПОЛЬЗУЕТСЯ</li> </ul>
Returns:	

`checkUniqueIndex(indexName, testValueSet=None)`

Проверяет уникальность ключа `indexName` этого объекта, или такого же объекта, но с измененными полями. Альтернативные значение полей передаются через хэш `testValueSet`

`clearIndexes()`

`convertLinks()`

`createLink(linkName, targetObj, attrs=None, addr=None, additionalObj=None, info=None, checkUnique=False)`

Создает связь текущего объекта и `targetObj` через мостовую таблицу `linkName`

Parameters:	<ul style="list-style-type: none"> <li>• <b>linkName</b> –</li> <li>• <b>targetObj</b> –</li> <li>• <b>attrs</b> – словарь с атрибутами, которые будут выставлены у объекта мостовой связи</li> <li>• <b>addr</b> – адрес, с которым привяжется <code>targetObj</code> к текущему объекту</li> <li>• <b>additionalObj</b> – Массив словарей вида <code>{'linkName': имя ссылки, 'obj': объект}</code>, объекты из которого (<code>obj</code>) будут присвоены ссылкам с названием <code>linkName</code> соответственно у объекта мостовой связи !!!WTF???</li> <li>• <b>info</b> – попадет в одноименный параметр у действия <code>preCreate</code> объекта мостовой связи</li> <li>• <b>checkUnique</b> – если <code>True</code> и уже существует объект мостовой связи для текущего объекта и <code>targetObj</code> - будет брошено исключение <code>dev_except.BridgeMustBeUnique</code></li> </ul>
Returns:	

`debugGetAttrValues()`

`deleteLink(linkName, targetObj)`

Удаляет мостовую связь `linkName` с объектом `targetObj`

Parameters:	<ul style="list-style-type: none"> <li>• <b>linkName</b> –</li> <li>• <b>targetObj</b> –</li> </ul>
Returns:	

`disableEventProcessor()`

`doAction(actName, params=None)`

Выполняет действие с именем `actName` у объекта

Parameters:	<ul style="list-style-type: none"> <li>• <b>actName</b> – имя действия, должно быть описано в секции <code>&lt;actions&gt;</code> в xml описании типа текущего объекта</li> <li>• <b>params</b> – словарь с параметрами действия</li> </ul>
Returns:	переменную <code>result</code> из тела действия, либо <code>None</code>

*classmethod* `doStaticAction(actName, params=None)`

`enableEventProcessor()`

`execHandler(role, objects, event)`

*classmethod* `export(index)`

`findRoot()`

Идет вверх по родителям до тех пор, пока не найдет объект, владеющей `portsClient`-ом (имеет связь с портом)

Returns:	найденный объект или <code>None</code>
----------	--

`finit()`

`getAddr()`

Возвращает адрес, с которым объект присоединен к объекту (если родителя нет, бросает исключение)

Returns:	целое число - адрес
----------	---------------------

`getAttribute(attrName)`

Возвращает значение атрибута с именем `attrName` Если атрибута с таким именем у объекта нет - будет брошено исключение `dev_except.UndefinedAttrField`

Parameters:	attrName –
Returns:	

`getBackLinkElements(typeName, linkName)`

Поиск объектов типа `typeName`, к которым текущий объект привязан по ссылке с названием `linkName`

Parameters:	<ul style="list-style-type: none"> <li>• <b>typeName</b> –</li> <li>• <b>linkName</b> – если у объектов типа <code>typeName</code> нет ссылки с именем <code>linkName</code> - будет брошено исключение</li> </ul>
Returns:	массив с объектами, либо пустой массив

`getBackLinkElementsIter(typeName, linkName)`

Поиск объектов типа `typeName`, к которым текущий объект привязан по ссылке с названием `linkName`

Parameters:	<ul style="list-style-type: none"> <li>• <b>typeName</b> –</li> <li>• <b>linkName</b> – если у объектов типа <code>typeName</code> нет ссылки с именем <code>linkName</code> - будет брошено исключение</li> </ul>
Returns:	итератор по ключам словаря с объектами, либо итератор по пустому словарю

`getChildByAddr(typeNameList, addr)`

Возвращает привязанные объекты перечисленных в `typeNameList` типов с адресом `addr`. Если у объекта не существует детей такого типа бросится исключение. Если ребенок не будет найден, будет брошено исключение `dev_except.ChildNotFound`

Parameters:	<ul style="list-style-type: none"> <li>• <b>typeNameList</b> – может быть листом, туплом (тогда ребенок с заданным адресом будет искаться среди объектов каждого типа) либо строкой (тогда поиск будет среди объектов только этого типа). Для листа или тупла будет возвращен первый найденный ребенок.</li> <li>• <b>addr</b> –</li> </ul>
-------------	---

**Returns:**

найденный объект-ребенок

`getChildListByType(typeName)`Возвращает массив привязанных объектов-детей типа `typeName`**Parameters:****typeName** – тип привязанных объектов, если у объекта не может быть детей такого типа - бросается исключение**Returns:**

массив объектов-детей, если детей нет - пустой массив

`getConstant(constName)`Возвращает значение константы `constName`. Если константы с таким именем нет - бросается исключение.**Parameters:****constName** – имя константы**Returns:**

значение константы

`getCurrOperatorSubject()``getDescription()`Либо поле `description`, либо тип объекта склеенный с `uniID`**Returns:**

строковое название объекта

`getDevTree()``getElementArray(linkName, filter='True')`Возвращает массив детей, привязанных к объекту через мостовую связь с названием `linkName`**Parameters:**

- **linkName** – имя мостовой таблицы. Если такой связи нет, бросается исключение
- **filter** – строка с кодом, который должен вернуть `True`, чтобы целевой объект попал в итоговый массив. В области видимости есть переменная `linked`, где лежит целевой объект.

**Returns:**

массив со связанными объектами

*classmethod* getElementById(*id*)

*classmethod* getElementIndexById(*id*)

*classmethod* getElementsByIndex(*indexName*, *keyValueDict=None*, *keyValue=None*)

getEquipName()

Возвращает имя оборудования, к которому принадлежит данный объект

**Returns:**

строка

getEventRegister()

getExtData(*field*)

*classmethod* getFilteredElements(*indexName*)

getHandlers()

getJson()

getLeftFarElements(*equipFilter=None*, *typeFilter=None*, *linkNameFilter=None*)

Нужна для поиска у объекта из правой части дальней связи привязанных объектов из левой части дальней связи. Например, для связи описанной так: 'common:commonright-orioniso:levelaccess' : {

```
'levelaccess': {
    'relation': 'u'M:1', 'alias': 'locale.IsoOrionAccessLevel'
}
```

} чтобы для объекта levelaccess получить связанный объект типа common:commonright нужно вызвать

```
levelaccess.getLeftFarElements('common', 'commonright', 'levelaccess')
```

**Parameters:**

- **equipFilter** – название оборудования
- **typeFilter** – тип объекта внутри оборудования
- **linkNameFilter** – название ссылки

**Returns:**

getLink(*linkName*, *targetObj*)

Возвращает объект из мостовой таблицы, являющийся связующим между текущим объектом и targetObj. Если объекты не связаны - будет брошено исключение dev\_except.BridgeElementNotFound

<b>Parameters:</b>	<ul style="list-style-type: none"> <li>• <b>linkName</b> – имя мостовой таблицы</li> <li>• <b>targetObj</b> –</li> </ul>
<b>Returns:</b>	объект

getLinkedElement(*linkName*)

Возвращает элемент по ссылке linkName. Если элемент не найден - будет брошено исключение dev\_except.LinkedElementNotFound

<b>Parameters:</b>	<b>linkName</b> – имя ссылки
<b>Returns:</b>	объект

getParameter(*paramName*)

Возвращает значение параметра paramName. Если параметра с таким именем нет - бросается исключение. Параметры похожи на атрибуты, но существуют только во время работы сервера и не записываются в базу.

<b>Parameters:</b>	<b>paramName</b> – имя параметра
<b>Returns:</b>	значение параметра

getParent(*level*!=0)

Возвращает родителя данного объекта, если родителя нет, бросает исключение dev\_except.ParentNotFound

<b>Parameters:</b>	<b>level</b> – уровень родителя 0 - прямой родитель, 1 - родитель родителя и т.д.
<b>Returns:</b>	объект родителя

getRightFarElements(*equipFilter=None, typeFilter=None, linkNameFilter=None*)

Нужна для поиска у объекта из левой части дальней связи привязанных объектов из правой части дальней связи. Например, для связи описанной так: 'orioniso:accesspoint-common:area' : {

```

    'in_area': {
        'relation': 'u'M:1', 'alias': locale.EntryTimeZone
    }

```

} чтобы для объекта `accesspoint` получить связанный объект типа `common:area` нужно вызвать

```
accesspoint.getRightFarElements('common', 'area', 'in_area')
```

Parameters:	<ul style="list-style-type: none"> <li>• <b>equipFilter</b> – название оборудования</li> <li>• <b>typeFilter</b> – тип объекта внутри оборудования</li> <li>• <b>linkNameFilter</b> – название ссылки</li> </ul>
Returns:	

*classmethod* `getStruct()`

`getTypeName()`

Возвращает название тип объекта внутри оборудования

Returns:	
----------	--

`getUniID()`

Возвращает уникальный идентификатор объекта оборудования

Returns:	строка
----------	--------

`hasAttribute(attrName)`

Проверяет наличие у объекта атрибута с именем `constName`

Parameters:	<b>attrName</b> –
Returns:	True or False

`hasConstant(constName)`

Проверяет наличие у объекта константы с именем `constName`

Parameters:	<b>constName</b> –
Returns:	True or False

`hasEditableField(fieldName)`

Проверяет есть ли у объекта поле с именем `fieldName` и доступно ли оно для редактирования

Parameters:	<b>fieldName</b> –
-------------	--------------------



<b>Returns:</b>	True or False
-----------------	---------------

hasExtData(*field*)

hasParameter(*paramName*)

Проверяет наличие у объекта параметра с именем *constName*

<b>Parameters:</b>	<i>paramName</i> –
--------------------	--------------------

<b>Returns:</b>	True or False
-----------------	---------------

hasParent(*level*=0)

Возвращает родителя данного объекта

<b>Parameters:</b>	<i>level</i> – уровень родителя 0 - прямой родитель, 1 - родитель родителя и т.д.
--------------------	---

<b>Returns:</b>	объект родителя или None, если его нет
-----------------	--

isLinked(*linkName*, *targetObj*)

Если объект связан с *targetObj* через мостовую таблицу с названием *linkName* - возвращает адрес

<b>Parameters:</b>	<ul style="list-style-type: none"> <li>• <i>linkName</i> – имя мостовой таблицы</li> <li>• <i>targetObj</i> –</li> </ul>
--------------------	--

<b>Returns:</b>	адрес <i>targetObj</i> у текущего объекта, либо False
-----------------	---

isLinkedElement(*linkName*)

Проверяет не пуста ли ссылка с именем *linkName*

<b>Parameters:</b>	<i>linkName</i> –
--------------------	-------------------

<b>Returns:</b>	False если ссылка пуста, True если она на что то ссылается
-----------------	--

isMarked()

Проверяет, помечен ли элемент

<b>Returns:</b>	True or False
-----------------	---------------

leaveParent(*dbCorrect*=True, *skipTrig*=False)

Отвязывает объект от родителя

Parameters:	<ul style="list-style-type: none"> <li>• <b>dbCorrect</b> – записывать ли изменения в БД, лучше не трогать - приследующем запуске могут возникнуть “висячие” ссылки. Они корректируются во время загрузки сервера.</li> <li>• <b>skipTrig</b> – если True - ChangeParentAction-ы выполняться не будут</li> </ul>
Returns:	

`linkUpdated(linkName, pushNotif=True)`

`loadingCorrectIndexes()`

`mark()`

Помечает элемент. Можно использовать для своих целей, все отмеченные элементы можно потом получить функцией

Returns:

`new(createAction=True)`

`newUniId()`

`printDevTree(doc)`

`processEvent(evt)`

`pushEvent(event)`

Добавление приведенного к единому формату события от оборудования в очередь обработки сервером

Parameters: **event** – словарь вида

```
{
    'notification': {
        'code': код события,
    }, 'statement': {
        'adverbialTime': {
            'param': таймштамп время происшествия события
        }, 'adverbialMode': {
            'param': параметр события
        }, 'directObj': {
```

```

    'dev': {
        'equip': название оборудования, 'type': название типа внутри
        оборудования, 'id': уникальный идентификатор объекта
    }
}, 'indirectObj': {
    'dev': {
        'equip': название оборудования, 'type': название типа внутри
        оборудования, 'id': уникальный идентификатор объекта
    }
}, 'subject': {
    'code': код карты субъекта 'dev': {
        'equip': 'common', 'type': 'subject', 'id': уникальный идентификатор объекта
    }
}
}
}

```

**Returns:**

`removeChild(child, dbCorrect=True, skipTrig=False)`

Отвязывает от объекта ребенка `child`

**Parameters:**

- **child** – объект ребенка, который нужно отвязать
- **dbCorrect** – записывать ли изменения в БД, лучше не трогать - приследующем запуске могут возникнуть "висячие" ссылки. Они корректируются во время загрузки сервера.
- **skipTrig** – если True - ChangeParentAction-ы выполняться не будут

**Returns:**

`selfDelete(deleteAsLink=False)`

Удаление объекта

**Parameters:**

**deleteAsLink** – нужна для удаления объекта мостовой связи, передается в `delAction` параметром, используется только в Apollo и руками трогать не нужно

Returns:

`setAttr(fieldName, value)`

Устанавливает у объекта атрибуту `fieldName` значение `value`. Если атрибут не доступен для редактирования - бросается исключение `dev_except.NoEditableField` При установке атрибута вызываются функции из `postAction` и `preAction`

Parameters:	<ul style="list-style-type: none"> <li>• <b>fieldName</b> – имя атрибута</li> <li>• <b>value</b> – значение аргумента</li> </ul>
-------------	--

Returns:	
----------	--

`setAttribute(attrName, value)`

Устанавливает у объекта атрибуту `fieldName` значение `value` даже если атрибут не доступен для редактирования При установке атрибута НЕ вызываются функции из `postAction` и `preAction`

Parameters:	<ul style="list-style-type: none"> <li>• <b>fieldName</b> – имя атрибута</li> <li>• <b>value</b> – значение аргумента</li> </ul>
-------------	--

Returns:	
----------	--

`setHandlers(handlers)`

`setParameter(paramName, value)`

Устанавливает значение `value` параметра `paramName`. Параметры похожи на атрибуты, но существуют только во время работы сервера и не записываются в базу.

Parameters:	<ul style="list-style-type: none"> <li>• <b>paramName</b> – имя параметра</li> <li>• <b>value</b> – новое значение параметра</li> </ul>
-------------	---

Returns:	значение параметра
----------	--------------------

`show()`

`unbindElement(linkName, dbCorrect=True, ignoreInstead=False, skipActions=False)`

Очищает ссылку `linkName` Если при выполнения `postAction` возникнет ошибка будет брошено исключение `dev_except.PostActionFail`

Parameters:	<ul style="list-style-type: none"> <li>• <b>linkName</b> –</li> <li>• <b>dbCorrect</b> – если False - то изменения в базу записаны не будут. Нужно для архивных объектов, лучше не трогать</li> <li>• <b>ignoreInstead</b> – если истина - то даже при наличии у ссылки insteadAction - действие выполнено не будет, а будет очищена ссылка</li> <li>• <b>skipActions</b> – если True - то post и pre actions выполняться не будут</li> </ul>
Returns:	всегда True

`unBindFarObject(linkName)`

Очищает дальнюю ссылку linkName

Parameters:	linkName –
Returns:	

`walk(actName, childType=None, params=None)`

Перебор детей у объекта и выполнение у них действия

Parameters:	<ul style="list-style-type: none"> <li>• <b>actName</b> – имя выполняемого действия</li> <li>• <b>childType</b> – тип детей, у которых будет выполнено действие, если None - то у всех типов детей</li> <li>• <b>params</b> – параметры, передаваемые в действие</li> </ul>
Returns:	

`walkArray(actName, linkName, params=None, filter=None)`

Перебор объектов, связанных с текущим объектом мостовой связью с именем linkName и выполнение у них действия actName

Parameters:	<ul style="list-style-type: none"> <li>• <b>actName</b> –</li> <li>• <b>linkName</b> –</li> <li>• <b>params</b> – передаваемые в действие параметры</li> <li>• <b>filter</b> – фильтр, аналогичный фильтру функции getElementArray</li> </ul>
Returns:	

`walkBackLink(actName, typeName, linkName, params=None)`

Перебор объектов типа `typeName`, к которым текущий объект привязан по ссылке с названием `linkName` и выполнение у них действия `actName` с параметрами `params`

Parameters:	<ul style="list-style-type: none"> <li>• <b>actName</b> –</li> <li>• <b>typeName</b> –</li> <li>• <b>linkName</b> –</li> <li>• <b>params</b> –</li> </ul>
Returns:	

`walkLeftFarElements(actName, params=None, equipFilter=None, typeFilter=None)`

Перебор найденных объектов из левой части дальней ссылки и выполнение у них action-a

Parameters:	<ul style="list-style-type: none"> <li>• <b>actName</b> – имя действия</li> <li>• <b>params</b> – параметры, передаваемые в действие</li> <li>• <b>equipFilter</b> –</li> <li>• <b>typeFilter</b> –</li> </ul>
Returns:	

`walkRightFarElements(actName, params=None, equipFilter=None, typeFilter=None)`

Перебор найденных объектов из правой части дальней ссылки и выполнение у них action-a

Parameters:	<ul style="list-style-type: none"><li>• <b>actName</b> – имя действия</li><li>• <b>params</b> – параметры, передаваемые в действие</li><li>• <b>equipFilter</b> –</li><li>• <b>typeFilter</b> –</li></ul>
Returns:	

```
class dev_obj_model_element.EmptyEventPacket  
    next()
```

## 11.4 equipment\_interface module

```
class equipment_interface.CollectionOfTypes(eq, typeName)
    delEl(index)
    getCount()
    getDeviceIdList()
    getEl(index)
    getElById(id)
    getElementIndex(id)
    length()
    new(attrib)
    show()

class equipment_interface.EquipmentInterface(**kwargs)
    Bases: object
```



Предоставляет программный интерфейс модуля оборудования.

`addChild(**kwargs)`

`bind(**kwargs)`

`calculateAttributes(**kwargs)`

*classmethod* `connectFarObjects(**kwargs)`

`createLink(**kwargs)`

`debugPrintDeviceTree(**kwargs)`

`debug_getEquipModel(**kwargs)`

`debug_getEquipment(**kwargs)`

`delAllBridges(**kwargs)`

`delAllElements(**kwargs)`

`delElement(**kwargs)`

`deleteLink(**kwargs)`

*classmethod* `disconnectFarObjects(**kwargs)`

*classmethod* `disconnectLeftFarObjects(**kwargs)`

*classmethod* `disconnectRightFarObjects(**kwargs)`

`doAction(**kwargs)`

`execAction(**kwargs)`

`execSysAction(**kwargs)`

`export(**kwargs)`

`finitAllDev(**kwargs)`

`getAttributeDescription(**kwargs)`

`getCollectionInterface(**kwargs)`

`getCollectionNameList(**kwargs)`

`getDescription(**kwargs)`

`getDeviceActionList(**kwargs)`

`getDeviceAttributes(**kwargs)`

`getDeviceIdListOfType(**kwargs)`  
`getDeviceListOfType(**kwargs)`  
`getDeviceParent(**kwargs)`  
`getElementById(**kwargs)`  
`getElements(**kwargs)`  
`getElementsByIndex(**kwargs)`  
*classmethod* `getInstance(**kwargs)`  
*classmethod* `getInstanceNameList(**kwargs)`  
`getLastMessage(**kwargs)`  
*classmethod* `getLeftFarObjectsList(**kwargs)`  
`getName(**kwargs)`  
`getObjectHandlers(**kwargs)`  
*classmethod* `getRightFarObjectsList(**kwargs)`  
*classmethod* `getStruct(**kwargs)`  
`getTypeNameList(**kwargs)`  
*classmethod* `init(**kwargs)`  
`initAllDev(**kwargs)`  
`installTp(**kwargs)`  
`load(**kwargs)`  
`newElement(**kwargs)`  
`reinstall(**kwargs)`  
`removeChild(**kwargs)`  
`setAttr(**kwargs)`  
`setHandlers(**kwargs)`  
`show(**kwargs)`  
`unBind(**kwargs)`

## 11.5 equipment\_module module

```
class equipment_module.EquipmentModule(**kwargs)
```

Bases: object

*classmethod* addEventProcessor(\*\*kwargs)

calculateAttributes(\*\*kwargs)

*classmethod* checkActPermitsCount(\*\*kwargs)

*classmethod* connectFarObjects(\*\*kwargs)

dbConn = None

*classmethod* disconnectFarObjects(\*\*kwargs)

*classmethod* disconnectLeftFarObjects(\*\*kwargs)

*classmethod* disconnectRightFarObjects(\*\*kwargs)

*classmethod* eventsIterator(\*\*kwargs)

*classmethod* farRuleWalk(\*\*kwargs)

*classmethod* getClientLicenceLimitations(\*\*kwargs)

*classmethod* getCurrIncident(\*\*kwargs)

*classmethod* getCurrOperator(\*\*kwargs)

getElementById(\*\*kwargs)

*classmethod* getEventProcessors(\*\*kwargs)

*classmethod* getInstance(\*\*kwargs)

*classmethod* getLeftFarObjectsList(\*\*kwargs)

*classmethod* getMark(\*\*kwargs)

getName(\*\*kwargs)

*classmethod* getRightFarObjectsList(\*\*kwargs)

*classmethod* getStruct(\*\*kwargs)

*classmethod* init(\*\*kwargs)

load(\*\*kwargs)

*classmethod* newMark(\*\*kwargs)

nextAutoIncrement(\*\*kwargs)

*classmethod* pushToEventQueue(\*\*kwargs)

*classmethod* removeEventProcessor(\*\*kwargs)

*classmethod* setCurrIncident(\*\*kwargs)

*classmethod* setCurrOperator(\*\*kwargs)

*static* setDBConn(\*args, \*\*kwargs)

## 11.6 primitive module

`primitive.bitSet(boolValuesArray)`

`primitive.cast(typeName, value)`

`primitive.hexFormat(val, len)`

`primitive.length(val)`

`primitive.swap(val)`

`primitive.swap4(val)`

`primitive.swapBytes(val, bytes)`

## 12 ПРИЛОЖЕНИЕ 7 - СПИСОК СОБЫТИЙ ДРАЙВЕРА ПО SIGUR

В таблице приведен перечень событий ПО Sigur.

код ESM	код в Sigur	Название события	использовани е ID1 ("apId")	использовани е ID2 ("empId")	Примечание
9501	-	Потеря связи драйвера с ПО Sigur	id объекта драйвер ESM	не определено	события драйвера ESM
9502	-	Восстановление связи драйвера с ПО Sigur	id объекта драйвер ESM	не определено	события драйвера ESM
9503	-	Ошибка авторизации драйвера с ПО Sigur	id объекта драйвер ESM	не определено	события драйвера ESM
9504	0	(тестовое событие)	не определено	не определено	
9505	1	Зарегистрирован взлом.	ID точки доступа	ID объекта доступа	
9506	2	Зарегистрирован проход в разблокированном режиме.	ID точки доступа	ID объекта доступа	
9507	3	Зарегистрирован проход, санкционированный с кнопки.	ID точки доступа	ID объекта доступа	
9508	4	Зарегистрирован проход.	ID точки доступа	ID объекта доступа	
9509	5	Зарегистрирован проход при открытой двери.	ID точки доступа	ID объекта доступа	
9510	6	Зарегистрирован проезд по путевому листу.	ID точки доступа	ID объекта доступа	
9511	7	Доступ запрещен.	ID точки доступа	ID объекта доступа	
9512	8	Доступ запрещен. Введен неверный PIN-код.	ID точки доступа	ID объекта доступа	
9513	9	Доступ запрещен. Контроллер не готов.	ID точки доступа	ID объекта доступа	
9514	10	Доступ запрещен. Неизвестный код пропуска.	ID точки доступа	ID объекта доступа	

код ESM	код в Sigur	Название события	использовани е ID1 ("apId")	использовани е ID2 ("empId")	Примечание
9515	11	Доступ запрещен. Режим не позволяет проход.	ID точки доступа	ID объекта доступа	
9516	12	Доступ запрещен. Нет допуска на точку доступа.	ID точки доступа	ID объекта доступа	
9517	13	Доступ запрещен. Нет допуска в это время.	ID точки доступа	ID объекта доступа	
9518	14	Доступ запрещен. Повторный проход.	ID точки доступа	ID объекта доступа	
9519	15	Доступ запрещен. Срок действия ключа истек.	ID точки доступа	ID объекта доступа	
9520	16	Пожарная тревога! Произведена аварийная разблокировка.	ID точки доступа	не используется	
9521	17	Пожарная тревога завершена.	ID точки доступа	не используется	
9522	18	Корпус контроллера открыт.	ID точки доступа	не используется	
9523	19	Корпус контроллера закрыт.	ID точки доступа	не используется	
9524	20	Связь с точкой доступа потеряна.	ID точки доступа	не используется	
9525	21	Связь с точкой доступа восстановлена.	ID точки доступа	не используется	
9526	22	Закрытие ворот.	ID точки доступа	ID объекта доступа	
9527	23	Открытие ворот.	ID точки доступа	ID объекта доступа	
9528	24	Доступ разрешен.	ID точки доступа	ID объекта доступа	
9529	25	Удержание двери в открытом состоянии начато.	ID точки доступа	не используется	



код ESM	код в Sigur	Название события	использовани е ID1 ("apId")	использовани е ID2 ("empId")	Примечание
9530	26	Истек таймаут двойной идентификации.	ID точки доступа	ID объекта доступа	
9531	27	(не используется)	ID точки доступа	ID объекта доступа	
9532	28	Переход на работу от сети (восстановление питания)	ID точки доступа	не используется	
9533	29	Переход на работу от аккумулятора (потеря сетевого питания)	ID точки доступа	не используется	
9534	30	Установка режима точки доступа "Нормальный"	ID точки доступа	не используется	
9535	31	Установка режима точки доступа "Заблокированный"	ID точки доступа	не используется	
9536	32	Установка режима точки доступа "Разблокированный"	ID точки доступа	не используется	
9537	33	Переход охранного шлейфа в статус "не активен" (снятие шлейфа)	ID шлейфа	не используется	не требуется обрабатывать
9538	34	Переход охранного шлейфа в статус "активен" (взятие шлейфа)	ID шлейфа	не используется	не требуется обрабатывать
9539	35	Переход охранного шлейфа в статус "тревога" (тревога шлейфа)	ID шлейфа	не используется	не требуется обрабатывать
9540	36	Переход точки доступа в состояние "закрыта" (закрытие двери)	ID точки доступа	не используется	
9541	37	Переход точки доступа в состояние "открыта" (открытие двери)	ID точки доступа	не используется	
9542	38	Удержание двери в открытом состоянии закончено.	ID точки доступа	не используется	

код ESM	код в Sigur	Название события	использовани е ID1 ("apId")	использовани е ID2 ("empId")	Примечание
9543	39	Начало ожидания санкции охраны	ID точки доступа	не используется	
9544	40	Окончание ожидания санкции охраны	ID точки доступа	не используется	
9545	41	Отказ от доступа	ID точки доступа	ID объекта доступа	
9546	42	Ожидание сопровождающего	ID точки доступа	ID объекта доступа	
9547	43	Ожидание ввода PIN кода	ID точки доступа	ID объекта доступа	
9548	44	Ожидание алкотеста	ID точки доступа	ID объекта доступа	